PlaySlot: Learning Inverse Latent Dynamics for Controllable Object-Centric Video Prediction and Planning

Angel Villar-Corrales [1] **Sven Behnke** [1]

Abstract

Predicting future scene representations is a crucial task for enabling robots to understand and interact with the environment. However, most existing methods rely on video sequences and simulations with precise action annotations, limiting their ability to leverage the large amount of available unlabeled video data. To address this challenge, we propose *PlaySlot*, an object-centric video prediction model that infers object representations and latent actions from unlabeled video sequences. It then uses these representations to forecast future object states and video frames. PlaySlot allows to generate multiple possible futures conditioned on latent actions, which can be inferred from video dynamics, provided by a user, or generated by a learned action policy, thus enabling versatile and interpretable world modeling. Our results show that PlaySlot outperforms both stochastic and object-centric baselines for video prediction across different environments. Furthermore, we show that our inferred latent actions can be used to learn robot behaviors sample-efficiently from unlabeled video demonstrations. Videos and code are available at https://play-slot.github.io/PlaySlot/.

1. Introduction

Accurate and flexible world models are crucial for autonomous systems to reason about their surroundings, predict possible future outcomes, and plan their actions effectively. Such models require a structured representation of the world that supports generalization, robustness, and controllability, even in complex and dynamic scenarios.

Humans naturally achieve such understanding by parsing their environment into a *background* and multiple separate *objects*, which can interact with each other and can be recombined to form more complex entities (Johnson, 2018; Kahneman et al., 1992). Neural networks equipped with such compositional inductive biases have the ability to learn structured object-centric representations with desirable properties such as robustness (Bengio et al., 2013; Dittadi et al., 2022), generalization to novel compositions (Greff et al., 2020), transferability to novel tasks (Zhang et al., 2022), and sample efficiency (Mosbach et al., 2024), among others.

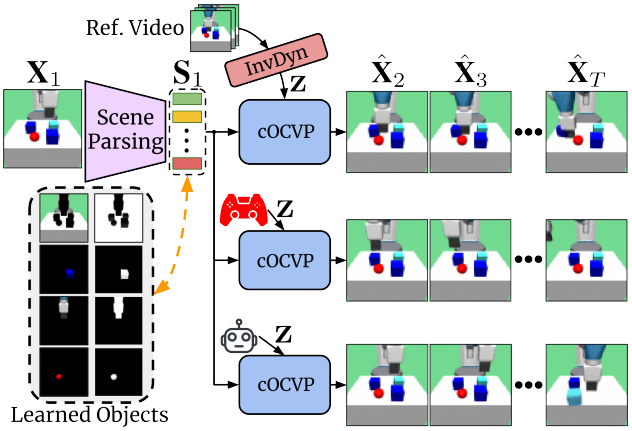


Figure 1: PlaySlot parses an image $\mathbf{X}_1$ into its object components $\mathbf{S}_1$. It then predicts multiple future object states and frames with an object-centric video prediction module (cOCVP) conditioned on latent actions $\boldsymbol{Z}$, which can be inferred from a reference video with our InvDyn module, provided as input, or generated by a learned action policy.

Building on these foundations, the field of object-centric learning has made great advances in recent years, progressing from learning object representations in simple synthetic images (Locatello et al., 2020; Burgess et al., 2019) and videos (Kipf et al., 2022; Elsayed et al., 2022), towards more complex real-world scenes (Seitzer et al., 2023; Zadaianchuk et al., 2024). Recently, the field of object-centric video prediction combines these learned object representations with forward-dynamics models and has shown great promise for multiple downstream applications such as modeling object dynamics (Villar-Corrales et al., 2023; Wu et al., 2023a) or action planning (Yoon et al., 2023;

[1]Autonomous Intelligent Systems, Computer Science Institute VI – Intelligent Systems and Robotics, Center for Robotics and the Lamarr Institute for Machine Learning and Artificial Intelligence. Correspondence to: Angel Villar-Corrales <villar@ais.uni-bonn.de>.

Mosbach et al., 2024). However, such models are currently limited to deterministic environments or rely on videos and simulations with precise action labels to forecast scene dynamics, limiting their ability to leverage unlabeled video data and serve as world models for robotic applications.

In this work, we propose *PlaySlot*, a novel method for controllable video prediction using object-centric representations. PlaySlot learns in a self-supervised manner from video to infer object representations, called slots, and latent action embeddings, which are computed using our proposed *InvDyn* module to capture the scene dynamics. PlaySlot then predicts future video frames conditioned on the inferred object slots and latent actions. At inference time, as illustrated on Fig. 1, PlaySlot parses the observed environment into a set of object slots, each of them representing a different object in the image. Then, PlaySlot forecasts future object states and frames conditioned on past object slots and latent actions, which can be inferred from a video sequence using our proposed InvDyn module, provided by a human, or generated by a learned action policy.

In our experiments, we demonstrate that PlaySlot learns a rich and semantically meaningful action space, enabling accurate video prediction while providing high levels of controllability and interpretability. We show how PlaySlot effectively captures precise robot actions and seamlessly scales to scenes with multiple moving objects or to real-world robotics data, outperforming several controllable video prediction baselines. Moreover, we show that the latent actions inferred by PlaySlot enable sample-efficient learning of robot behaviors from unlabeled demonstrations.

In summary, our contributions are as follows:

- We propose *PlaySlot* – an object-centric video prediction model that infers object representations and latent actions from unlabeled videos, and uses them to forecast future object states and video frames.
- PlaySlot outperforms several video prediction models across diverse robotic environments, while showing superior interpretability and control capabilities.
- The object representations and latent actions inferred by PlaySlot can be used to learn robot behaviors from unlabeled video demonstrations sample efficiently.

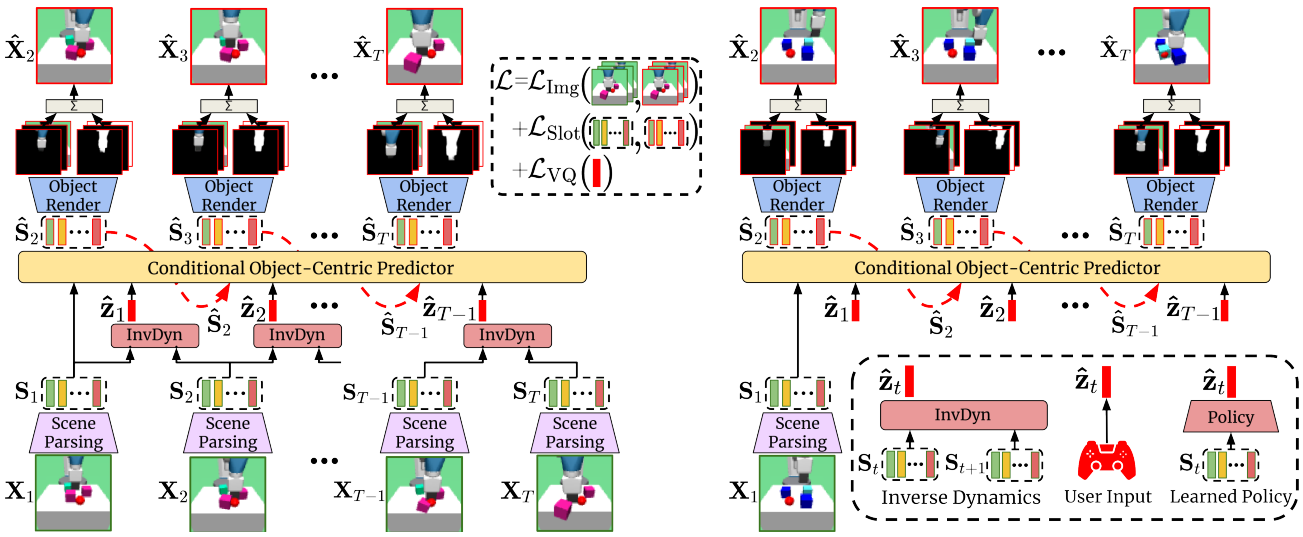
2. Related Work

Unsupervised Object-Centric Learning Object-centric representation methods aim to parse in an unsupervised manner an image or video into a set of N_S latent vectors called slots, where each of them binds to a different object in the scene (Greff et al., 2020; Locatello et al., 2019). Early slot-based methods aimed to learn object representations from synthetic images (Locatello et al., 2020; Singh et al., 2021; Biza et al., 2023) or videos (Kipf et al., 2022; Creswell et al., 2021; Singh et al., 2022) by minimizing a reconstruction objective. To learn meaningful representations from real data, recent slot-based methods leverage weak supervision (Elsayed et al., 2022; Bao et al., 2023), large pretrained transformers (Seitzer et al., 2023; Aydemir et al., 2023; Zadaianchuk et al., 2024), or diffusion models (Jiang et al., 2023; Wu et al., 2023b) These object-centric representations benefit multiple downstream tasks such as reinforcement learning for robotic manipulation (Mosbach et al., 2024; Ferraro et al., 2023) or visual-question-answering (Mamaghan et al., 2024).

Object-Centric Video Prediction Object-centric video prediction aims to model the object dynamics and interactions in a video sequence with the goal of forecasting future object states and video frames. Several methods address this task using different architectural priors, including RNNS (Zoran et al., 2021; Assouel et al., 2022), transformers (Villar-Corrales et al., 2023; Wu et al., 2023a; Daniel & Tamar, 2024; Meo et al., 2024) or state-space models (Jiang et al., 2024), attaining a remarkable prediction accuracy on synthetic datasets. Recently, some methods improve the controllability of object-centric video prediction models by conditioning the prediction process on actions (Mosbach et al., 2024) or language captions (Wang et al., 2024). However, forecasting future object states without supervision in complex environments still remains an open challenge.

Learning Latent Actions from Unlabeled Videos: Videos provide abundant information about dynamics and activities, but often lack the action labels necessary for learning behaviors from video. To address this challenge, some methods train a latent policy directly from observations by learning a discrete latent action space and sampling the actions that minimize a reconstruction error (Edwards et al., 2019; Struckmeier & Kyrki, 2023). Another group of methods, to which PlaySlot belongs, learns inverse dynamics from unlabeled videos by predicting latent actions given pairs of observations, and uses them for learning behaviors for video games and robot simulations (Ye et al., 2022; Brandfonbrener et al., 2024; Schmidt & Jiang, 2024), as pretraining for Vision-Language-Action models (Ye et al., 2024) or for learning robot policies (Cui et al., 2024).

Latent action models have also been used for conditional video prediction. The most similar method to ours is *CADDY* (Menapace et al., 2021; 2022), which learns latent actions from a collection of unlabeled videos from a single domain and uses the latent actions as conditioning signal for predicting future frames. At inference time, CADDY maps user inputs to the latent space for playable video generation. Building upon this same principle, *Genie* (Bruce et al., 2024) proposes a foundation world model



a) PlaySlot training **b)** PlaySlot inference

Figure 2: Overview of PlaySlot training and inference processes. **(a)** PlaySlot is trained given unlabeled video sequences by inferring object representations $\mathbf{S}$ and latent actions $\hat{\mathbf{z}}$, and using these representations to autoregressively forecast future video frames and object states. **(b)** PlaySlot autoregressively forecasts future frames conditioned on a single frame $\mathbf{X}_1$ and latent actions $\hat{\mathbf{z}}$, which can be inferred from observations, provided by a user, or output by a learned action policy.

for playable video generation on diverse environments. However, both CADDY and Genie operate on holistic scene representations, which are limited for tasks that require relational reasoning, often struggle to model object relationships and interactions, and require human supervision to generalize to scenes with multiple moving agents.

3. PlaySlot

We propose *PlaySlot*, a novel framework for controllable object-centric video prediction from unlabeled video sequences. Fig. 2a) illustrates the training process in PlaySlot, as well as its main four components. Namely, given T video frames $\mathbf{X}_{1:T}$, our model employs as *Scene Parsing module* that decomposes these images into object representations, called slots, $\mathbf{S}_{1:T} = (\mathbf{S}_1, ..., \mathbf{S}_T)$, where $\mathbf{S}_t = (\mathbf{s}_t^1, ..., \mathbf{s}_t^{N_\mathbf{S}}) \in \mathbb{R}^{N_\mathbf{S} \times D_\mathbf{S}}$ is the set of $D_\mathbf{S}$-dimensional object slots parsed from frame $\mathbf{X}_t$. For each consecutive pair of frames, PlaySlot employs an *Inverse Dynamics* (InvDyn) module (Sec. 3.2) in order to estimate latent action embeddings $\hat{\mathbf{z}}_t$ that encode the actions taken by the agents in the scene between every consecutive pair of frames. The *Conditional Object-Centric Predictor* (cOCVP) (Sec. 3.3) forecasts future object states conditioned on past slots and latent actions estimated by InvDyn. Finally, the *object rendering* module decodes the object slots to render object images and masks, which can be combined via a weighted sum to render video frames.

At inference time, as shown in Fig. 2b), PlaySlot autoregressively predicts multiple possible sequence continuations conditioned on the initial object slots and latent action embeddings, which can be estimated by InvDyn, provided by human, or generated by a learned action policy.

3.1. Object-Centric Representation Learning

PlaySlot employs SAVi (Kipf et al., 2022), a recursive encoder-decoder model with a structured bottleneck of $N_\mathbf{S}$ permutation-equivariant object slots, to parse a sequence of video frames $\mathbf{X}_{1:T}$ into their object components $\mathbf{S}_{1:T}, \mathbf{S}_t \in \mathbb{R}^{N_\mathbf{S} \times D_\mathbf{S}}$. The slots $\mathbf{S}_0$ are sampled from a learned distribution and recursively refined to bind to the objects in the video frames. At time step t, SAVi encodes the corresponding input into feature maps $\mathbf{h} \in \mathbb{R}^{L \times D_h}$, which are fed to Slot Attention (Locatello et al., 2020) to iteratively refine the previous slot representations conditioned on the current features. Slot Attention performs cross-attention between the image features and slots with the attention weights normalized over the slot dimension, encouraging competition between slots so as to represent each feature location:

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

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

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

where v is a linear projection. The steps described in Equations (1) and (2) can be repeated multiple times with shared weights to iteratively refine the slots and obtain an accurate object-centric representation of the scene.

To map the object representations back to images, SAVi independently decodes each slot in $\mathbf{S}_t$ with a Spatial Broadcast Decoder (Watters et al., 2019) ($\mathcal{D}_{\text{SAVi}}$) to render an object image and mask, which can be normalized and combined via a weighted sum to render the reconstructed frame:

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

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

3.2. Learning Inverse Dynamics

In general, future frames depend not only on previous observations, but also on other variables, such as robot actions. We propose an inverse dynamics module (InvDyn) that estimates, given the object slots from two consecutive time steps, latent action embeddings $\hat{\mathbf{z}} \in \mathbb{R}^{D_\mathbf{z}}$ that encode the actions taken by the agents between such time steps:

$$\hat{\mathbf{z}}_t = \text{InvDyn}(\mathbf{S}_t, \mathbf{S}_{t+1}). \tag{5}$$

3.2.1. Action Parameterization

The parameterization of the latent actions determines the complexity of the transitions that can be modeled, as well as the degree of control that we have over the predictions. On the one hand, learning a finite set of latent actions allows for controllable video prediction while limiting the complexity of the dynamics that such actions can explain. On the other hand, continuous latent vectors can model complex transitions between frames with the drawback of less interpretability and control.

As a compromise between these two approaches, inspired by Menapace et al. (2021), we propose a hybrid approach to parameterize the latent actions $\hat{\mathbf{z}}_t$ with a discrete component $\mathbf{p}_t$ denoted as *action prototype*, which determines the high-level action taking place (e.g. move left, go up), and a continuous *action variability* $\mathbf{v}_t$, which captures non-deterministic dynamics in the environment and enables to interpolate between action prototypes. This combination allows for modeling complex frame transitions effectively.

3.2.2. InvDyn Module

We propose two variants of our inverse dynamics module. InvDyn_S processes object slots $\mathbf{S}_t$ along with an additional token `[ACT]` using a transformer encoder $f_\mathbf{z}$. It outputs a single latent action $\hat{\mathbf{z}}_t$ that captures the agent's action, making it well-suited for single-agent environments. In contrast, InvDyn_M processes each slot with a shared MLP, producing $N_\mathbf{S}$ latent action embeddings $\hat{\mathbf{Z}}_t = \{\hat{\mathbf{z}}_t^1, ..., \hat{\mathbf{z}}_t^{N_\mathbf{S}}\}$, each representing the action of a specific object in the scene. Below we explain the process for computing latent actions using InvDyn_S, which follows a similar procedure to that of InvDyn_M.

Following Menapace et al. (2021), we adopt a probabilistic formulation where InvDyn predicts the posterior distribution of scene dynamics, modeled as Gaussian:

$$\boldsymbol{\mu}_{\mathbf{d}t}, \boldsymbol{\sigma}_{\mathbf{d}t}^2 = f_\mathbf{z}(\mathbf{S}_t, \texttt{[ACT]}). \tag{6}$$

We then model the distribution of latent actions $\hat{\mathbf{z}}_t$ as the difference between the distributions of dynamics embeddings from two consecutive time steps:

$$\hat{\mathbf{z}}_t \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{z}t}, \boldsymbol{\sigma}_{\mathbf{z}t}^2) \text{ with } \begin{cases} \boldsymbol{\mu}_{\mathbf{z}t} = \boldsymbol{\mu}_{\mathbf{d}t+1} - \boldsymbol{\mu}_{\mathbf{d}t}, \\ \boldsymbol{\sigma}_{\mathbf{z}t}^2 = \boldsymbol{\sigma}_{\mathbf{d}t+1}^2 + \boldsymbol{\sigma}_{\mathbf{d}t}^2, \end{cases}, \tag{7}$$

from which we can sample the latent actions $\hat{\mathbf{z}}_t$.

To prevent the model from simply encoding the target scene into $\hat{\mathbf{z}}_t$, we regularize the latent action space by enforcing an information bottleneck. Specifically, we constrain the latent action space to be low dimensional, i.e., $D_\mathbf{z} << D_\mathbf{S}$. Furthermore, we parameterize the latent actions as the sum of a discrete action prototype $\mathbf{p}_t$ and an action variability embedding $\mathbf{v}_t$, where $\mathbf{p}_t$ is obtained by vector-quantizing the latent actions $\hat{\mathbf{z}}_t$, i.e. $\mathbf{p}_t = \text{VQ}(\hat{\mathbf{z}}_t)$. We empirically verify that the information bottleneck enforced by vector quantization achieves comparable performance to the one proposed by (Menapace et al., 2021), while requiring significantly fewer hyper-parameters.

This latent action parameterization ensures that our InvDyn module encodes only the essential dynamics, effectively capturing the agent's interaction with the scene while learning semantically meaningful action prototypes. Moreover, this hybrid factorization improves the controllability and interpretability of the prediction process while maintaining the ability to model complex scene dynamics.

3.3. Conditional Object-Centric Prediction

We employ a transformer-based (Vaswani et al., 2017) module to autoregressively predict future object slots conditioned on past object states and latent actions.

Our proposed predictor, cOCVP, is a transformer encoder with N_Pred layers. At each time step t, cOCVP takes as input all previous slots $\mathbf{S}_{1:t}$, action prototypes $\mathbf{p}_{1:t}$ and variability embeddings $\mathbf{v}_{1:t}$, all of which are first linearly projected into a shared token dimensionality. The slots are then conditioned by adding them with the corresponding projected action prototype and variability embeddings. Additionally, we incorporate sinusoidal positional encodings such that all slots from the same time step receive the same encoding, thus preserving the inherent permutation equivariance of the objects.

cOCVP forecasts the future slots $\hat{\mathbf{S}}_{t+1}$ by jointly modeling the object dynamics and interactions from the past object slots conditioned on the inferred latent actions. This process is summarized as:

$$\hat{\mathbf{S}}_{t+1} = \text{cOCVP}(f_{\text{S}}(\mathbf{S}_{1:t}) + f_{\text{p}}(\mathbf{p}_{1:t}) + f_{\text{v}}(\mathbf{v}_{1:t})), \quad (8)$$

where f_{S}, f_{p} and f_{v} are learned linear layers.

The prediction process can be initiated from the slots of a single reference frame $\mathbf{S}_1$ and the corresponding inferred latent actions $\hat{\mathbf{z}}_1$. This process is repeated autoregressively, with the predicted slots being appended to the input at each subsequent time step, allowing the generation of future object representations for a desired number of time steps T.

3.4. Learning Behaviors from Unlabeled Videos

Using a trained InvDyn module, we aim to learn a policy from unlabeled video expert demonstrations without the need for action or reward information. For this purpose, we compute with InvDyn a sequence of latent actions that explain the dynamics of the expert demonstrations, and then train a policy model f_π to regress such latent actions using the object slots from the corresponding time step.

At inference time, starting with a single observation $\mathbf{X}_1$, PlaySlot computes the corresponding object slots $\mathbf{S}_1$ and uses the policy to estimate a latent action $\hat{\mathbf{z}}_1$, which is decomposed into an action prototype $\mathbf{p}_1 = \text{VQ}(\hat{\mathbf{z}}_1)$ and variability embedding $\mathbf{v}_1 = \hat{\mathbf{z}}_1 - \mathbf{p}_1$. These representations are fed to cOCVP to forecast subsequent slots $\hat{\mathbf{S}}_2$. This process is repeated autoregressively, allowing the learned behavior to unfold within the model's latent imagination.

To map the latent actions generated by the policy f_π to the real action space, we introduce an action decoder $\mathcal{D}_{\text{a}}$. This module, implemented as a three-layer MLP, is trained to translate the latent actions inferred by InvDyn into the real-world actions using a small action-labeled dataset.

This approach shares similarities with Schmidt & Jiang (2024). However, whereas their method learns policies for simple games with a small discrete set of actions, our flexible action representation and conditional object-centric decoder enable us to learn more complex robot behaviors.

3.5. Training

We differentiate three different training stages in PlaySlot. We first train SAVi to parse video frames into object-centric representations by minimizing a reconstruction loss:

$$\mathcal{L}_{\text{SAVi}} = \sum_{t=1}^{T} ||\mathcal{D}_{\text{SAVi}}(\mathcal{E}_{\text{SAVi}}(\mathbf{X}_t)) - \mathbf{X}_t||_2^2, \quad (9)$$

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

Second, given the pretrained SAVi model, we jointly train InvDyn and cOCVP by minimizing a combined loss:

$$\mathcal{L}_{\text{PlaySlot}} = \sum_{t=2}^{T+1} \lambda_{\text{Img}}\mathcal{L}_{\text{Img}} + \lambda_{\text{Slot}}\mathcal{L}_{\text{Slot}} + \lambda_{\text{VQ}}\mathcal{L}_{\text{VQ}}, \quad (10)$$

$$\mathcal{L}_{\text{Img}} = ||\hat{\mathbf{X}}_t - \mathbf{X}_t||_2^2, \quad (11)$$

$$\mathcal{L}_{\text{Slot}} = ||\hat{\mathbf{S}}_t - \mathcal{E}_{\text{SAVi}}(\mathbf{X}_t)||_2^2, \quad (12)$$

$$\mathcal{L}_{\text{VQ}} = ||\text{sg}[\hat{\mathbf{z}}_t] - \mathbf{p}_t|| + 0.25 \cdot ||\hat{\mathbf{z}}_t - \text{sg}[\mathbf{p}_t]||, \quad (13)$$

where sg is the stop-gradient operator. $\mathcal{L}_{\text{Img}}$ measures the future frame prediction error, $\mathcal{L}_{\text{Slot}}$ aligns the predicted object slots with the actual object-centric representations, and $\mathcal{L}_{\text{VQ}}$ encourages the learning of meaningful action prototypes while regularizing the latent actions to align with their prototypes (Van Den Oord & Vinyals, 2017). We do not employ teacher forcing, enabling the predictor model to learn to handle its own imperfect predictions.

Finally, the policy model f_π and action decoder $\mathcal{D}_{\text{a}}$ are trained to regress the inferred latent actions $\hat{\mathbf{z}}$ and ground truth actions $\mathbf{a}$, respectively:

$$\mathcal{L}_{f_\pi} = \sum_{t=1}^{T} ||f_\pi(\mathbf{S}_t) - \hat{\mathbf{z}}_t||, \quad (14)$$

$$\mathcal{L}_{\mathcal{D}_{\text{a}}} = \sum_{t=1}^{T} ||\mathcal{D}_{\text{a}}(\hat{\mathbf{z}}_t) - \mathbf{a}_t||. \quad (15)$$

4. Experiments

4.1. Experimental Setup

4.1.1. Datasets

We evaluate our method on three environments with distinct characteristics. Further details are provided in Appendix C.

ButtonPress This environment, based on *MetaWorld* (Yu et al., 2020), features a Sawyer robot arm that must press a red button. This environment depicts a non-object centric task involving complex shapes and textures.

BlockPush: This environment, inspired by (Li et al., 2020), features a robot arm and a table with multiple uni-colored cubes of different colors. The robot must push the block of distinct color into the location specified by a red target. This task evaluates the capabilities of an agent to reason about object relations and to model object collisions.

GridShapes: This dataset features two simple 2D shapes moving in grid-like patterns, restricted to up, down, left, or right directions on a colored background. The shapes randomly change direction with a predefined probability, introducing stochasticity to their motion. This simple dataset

Table 1: Quantitative evaluation of several object-centric (OC) and controllable (Cont.) video prediction models. Given six seed frames, all models predict the subsequent 15 frames. PlaySlot achieves the best results in datasets that require modeling object interactions (BlockPush) or feature multiple moving objects (GridShapes), while maintaining competitive performance on ButtonPress. Best two results are highlighted in boldface and underlined, respectively.

			BlockPush			**ButtonPress**			**GridShapes$_{2Objs}$**		
Model	**OC**	**Cont.**	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SVG	✗	✓	20.96	<u>0.898</u>	0.096	**32.23**	**0.950**	**0.011**	38.10	<u>0.988</u>	<u>0.002</u>
CADDY	✗	✓	<u>21.18</u>	**0.901**	<u>0.090</u>	24.58	0.853	0.028	<u>39.16</u>	0.986	0.006
SlotFormer	✓	✗	17.22	0.729	0.134	19.52	0.762	0.111	19.74	0.795	0.149
OCVP	✓	✗	17.26	0.751	0.134	19.55	0.762	0.115	18.86	0.791	0.154
PlaySlot (Ours)	✓	✓	**21.41**	0.890	**0.066**	<u>26.03</u>	<u>0.878</u>	<u>0.025</u>	**54.09**	**0.996**	**0.001**

serves as benchmark to evaluate a model's ability to jointly predict the motion of multiple moving agents in the scene.

4.1.2. Implementation Details

All our models are implemented in PyTorch (Paszke et al., 2017) and trained on a single NVIDIA A100 GPU. PlaySlot uses SAVi (Kipf et al., 2022) with 128-dimensional object slots, as well as a convolutional encoder and spatial broadcast decoder as scene parsing and rendering modules, respectively. The conditional predictor and inverse dynamics modules are transformer encoders with four layers and a token dimension of 256. For the ButtonPress and BlockPush datasets, we use the InvDyn$_S$ variant with eight different 16-dimensional action prototypes, whereas for GridShapes we use InvDyn$_M$ with five distinct eight-dimensional action prototypes. Further implementation details are provided in Appendix B.

4.2. Video Prediction

We evaluate PlaySlot for video prediction and compare it with different baselines, including the object-centric video prediction models SlotFormer (Wu et al., 2023a) and OCVP-Seq (Villar-Corrales et al., 2023), the stochastic video prediction model SVG-LP (Denton & Fergus, 2018) and the playable video generation model CADDY (Menapace et al., 2021). For a fair comparison, all models are trained with six seed frames to predict the subsequent eight, and evaluated for 15 predictions. For CADDY, PlaySlot and SVG, we predict future frames conditioned on latent actions or vectors inferred from the ground truth sequence. Additionally, on the BlockPush and ButtonPress datasets all models are trained using sequences with random exploration policies, and evaluated on expert demonstrations.

We evaluate the quality of the predicted frames using standard metrics: PSNR, SSIM (Wang et al., 2004) and LPIPS (Zhang et al., 2018). A quantitative comparison of the methods is presented in Tab. 1. As expected, deterministic object-centric models (i.e. SlotFormer and OCVP) perform poorly, as they cannot infer the agent's actions and simply average over multiple possible futures. Our proposed method outperforms all other models on both the BlockPush and GridShapes datasets, demonstrating PlaySlot's superior ability to forecast future video frames in environments involving multiple object interactions and moving agents, respectively.

Fig. 3 depicts a qualitative comparison of the best performing methods on the ButtonPress and BlockPush datasets, respectively. On the ButtonPress dataset, as shown in Fig. 3a), all methods accurately model the motion of the robot arm. However, on the more complex BlockPush task, depicted in Fig. 3b), SVG and CADDY fail to model the object collisions, leading to blurriness and vanishing objects. In contrast, PlaySlot maintains sharp object representations and correctly models interactions between objects, leading to accurate frame predictions. Further qualitative evaluations are provided in Appendix E.

4.3. Model Analysis

Impact of Number of Moving Objects: We evaluate the performance of our method for different number of moving objects. For this purpose, we train two PlaySlot variants with the InvDyn$_S$ and InvDyn$_M$ inverse dynamics modules, respectively, and compare them with the SVG and CADDY baselines on several variants of the GridShapes dataset featuring a different number of objects, ranging from one to five moving shapes. The results are depicted in Fig. 4. CADDY and PlaySlot with InvDyn$_S$, which encode scene dynamics using a single latent action, perform strongly when jointly forecasting one or two objects, but experience a sharp drop in performance as the number of objects increases. SVG scales to multiple moving objects but encodes the dynamics of all objects into a single distribution, limiting its flexibility and control over the predictions. In contrast, PlaySlot with InvDyn$_M$ uses a latent action per object, allowing to scale seamlessly to a large number of moving agents by individually modeling the motion of each object, thus outperforming all baselines.

Action Representation: In Tab. 2 we compare the hybrid latent action representation used in PlaySlot with three different variants that use continuous latent actions, a discrete

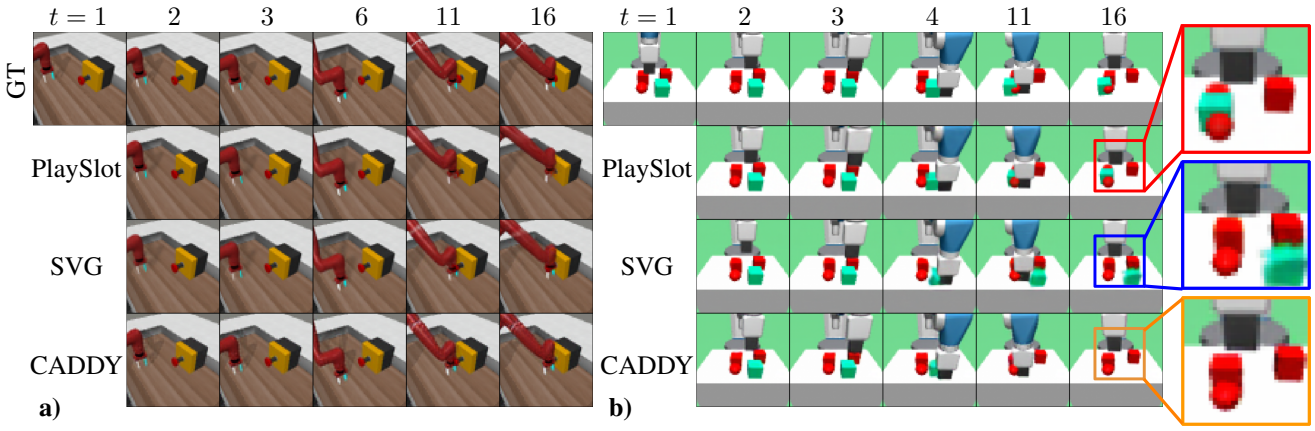


Figure 3: Qualitative comparison on (a) ButtonPress and (b) BlockPush datasets. PlaySlot accurately predicts the scene dynamics, whereas baselines fail to predict object interactions, leading to blurriness and disappearing objects.

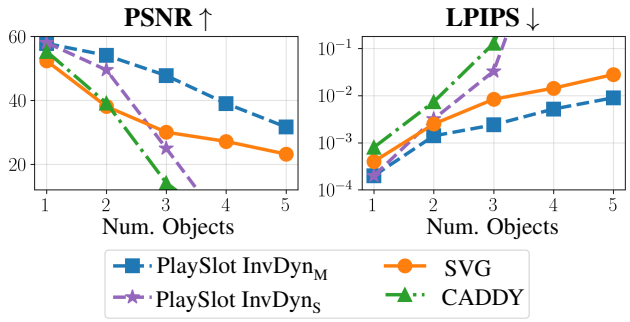


Figure 4: Quantitative results on the GridShapes dataset with different number of objects. PlaySlot outperforms the baselines, particularly for a higher number of objects.

set of latent actions, and an oracle variant with access to ground truth actions. Additionally, we evaluate the models on two different BlockPush variants, including a set with random exploration sequences, similar to the training distribution, and another set featuring expert demonstrations.

On the random exploration set, PlaySlot with a hybrid latent action performs comparably to the oracle, highlighting the ability of our InvDyn module to infer latent dynamics from unlabeled sequences. However, PlaySlot's video prediction performance drops on the expert demonstrations, where the variant using unconstrained continuous latent actions outperforms it. We attribute this to the large discrepancy between the action distribution of expert sequences and the training distribution, which challenges the generalization of the learned action prototypes.

Learned Actions: Fig. 5 depicts the effect of different action prototypes learned by PlaySlot on the BlockPush dataset. Given a single seed frame, we forecast 15 frames by repeatedly conditioning the predictor on the same action prototype. Furthermore, we visualize the predictions obtained using the latent actions inferred by our inverse dynamics module from the ground truth sequence; as well as the instance segmentation maps obtained by assigning a different color to each slot mask. PlaySlot learns to infer precise robot actions from visual observations and the physics of interacting objects, while distinctly representing each object in a different slot. Additionally, Fig. 5 shows that PlaySlot learns consistent and semantically meaningful actions, such as moving the robot towards the right (action 2), left (action 7), or upwards (action 4).

Table 2: Quantitative comparison of PlaySlot with variants using continuous and discrete latent actions, as well as an oracle model with access to the ground truth actions.

RobotDB	Model Variant	Results PSNR↑	SSIM↑	LPIPS↓
Random Exploration	PlaySlot	26.64	0.944	**0.016**
	w/ Cont. $\hat{\mathbf{z}}$	26.24	0.924	0.019
	w/ Discrete $\hat{\mathbf{z}}$	20.60	0.849	0.040
	w/ GT Actions	**27.77**	**0.955**	**0.016**
Expert Demos.	PlaySlot	21.41	0.890	0.065
	w/ Cont. $\hat{\mathbf{z}}$	22.00	0.900	0.061
	w/ Discrete $\hat{\mathbf{z}}$	18.00	0.791	0.109
	w/ GT Actions	**22.30**	**0.904**	**0.058**

Real-World Robotic Videos: We validate the applicability of PlaySlot to real-world robotic videos using the Sketchy (Cabi et al., 2020) dataset, which features a robotic gripper interacting with diverse objects. Fig. 6 shows a qualitative result of PlaySlot on Sketchy. Our model accurately infers the scene's inverse dynamics, reconstructing the ground truth sequence from a single reference frame. Furthermore, PlaySlot learns semantically meaningful and consistent action prototypes, capturing diverse behaviors such as opening the gripper (action 6), moving the robot downwards (action 2), or upwards (action 4).

4.4. Learning Behaviors from Expert Demonstrations

We evaluate the quality of PlaySlot's object-centric representations and inferred latent actions for a downstream be-

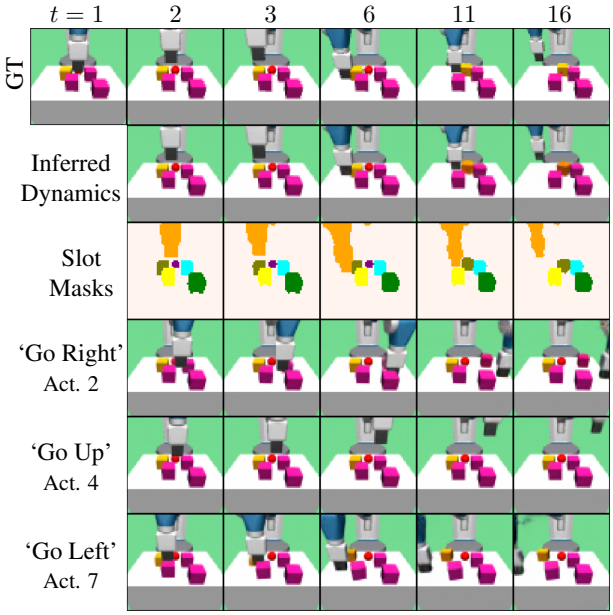


Figure 5: PlaySlot predictions given different latent actions, including inferred inverse dynamics and three action prototypes. PlaySlot learns accurate object-centric representations and semantically consistent action prototypes.

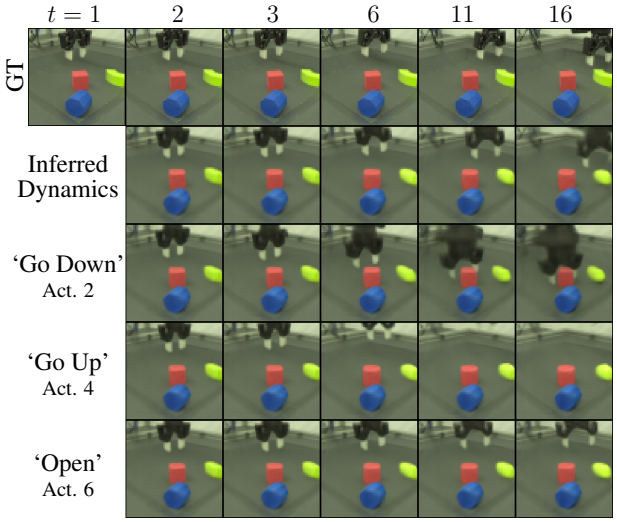


Figure 6: Qualitative results on a real-world robotics sequence. PlaySlot accurately predicts possible futures conditioned on a single reference frame and latent actions.

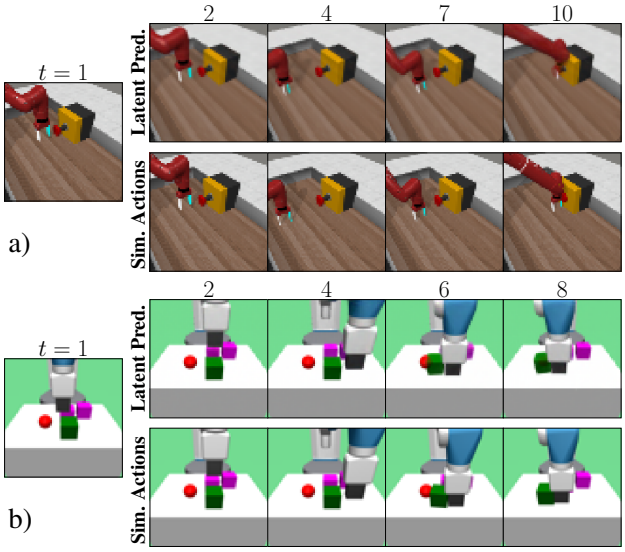


Figure 7: Predicted frames using latent actions generated by the learned policy, and sequences simulated by executing the decoded latent actions for a) ButtonPress and b) BlockPush learned behaviors.

havior learning task. To this end, we train a policy model and an action decoder, as described in Sec. 3.4, to imitate ButtonPress and BlockPush behaviors from a limited set of expert demonstrations. Fig. 7. illustrates the learned latent policies on the (a) ButtonPress and (b) BlockPush environments, respectively. The top row in each sequence (labeled *Latent Pred.*) shows predicted trajectories within PlaySlot's latent imagination, where the model, starting from a single reference frame, autoregressively generates latent actions using the policy model and predicts future scene states in the latent space. The bottom row (labeled *Sim. Actions*) depicts the simulated execution of the decoded latent actions in the corresponding environment. PlaySlot learns to solve both tasks within its latent imagination, successfully reasoning about object properties and generating a precise sequence of latent actions, which can be decoded into executable motions. Further results can be found in Appendix E.2.

5. Conclusion

We introduced PlaySlot, a novel framework for controllable object-centric video prediction. PlaySlot parses video frames into object slots, infers the scene's inverse dynamics, and predicts future object states and video frames by modeling the object dynamics and interactions, conditioned on inferred latent actions. Through extensive experiments, we demonstrated that PlaySlot learns a semantically rich and meaningful action space, allowing for accurate video frame predictions. Our method outperforms several baseline models, offering superior controllability and interpretability. Moreover, we demonstrated that the learned object representations and latent actions inferred by PlaySlot can be utilized to predict future frames with precise robot control, while also enabling the model to learn complex robot behaviors from unlabeled video demonstrations. This versatility makes PlaySlot a powerful and interpretable world model suitable for various tasks in autonomous systems.

Acknowledgment

This work was funded by grant BE 2556/16-2 (Research Unit FOR 2535 Anticipating Human Behavior) of the German Research Foundation (DFG)

Impact Statement

This paper presents work whose goal is to advance the field of machine learning by introducing PlaySlot – an object-centric video prediction model that forecasts future video frames conditioned on inferred object-centric representations and latent actions, enhancing its interpretability and controllability, as well as learning representations that can be used for action planning. This advancement is particularly relevant for domains such as robotics and autonomous systems, where understanding and predicting complex environments are essential for correct and safe operation. However, the deployment of such models in real systems requires careful consideration for ethical and safety challenges, such as biases in the training data or lack of transparency on the decision-making process.

References

Assouel, R., Castrejon, L., Courville, A., Ballas, N., and Bengio, Y. Vim: Variational independent modules for video prediction. In *Conference on Causal Learning and Reasoning*, pp. 70–89. PMLR, 2022.

Aydemir, G., Xie, W., and Guney, F. Self-supervised object-centric learning for videos. *Advances in Neural Information Processing Systems (NeurIPS)*, 36:32879–32899, 2023.

Bao, Z., Tokmakov, P., Wang, Y.-X., Gaidon, A., and Hebert, M. Object discovery from motion-guided tokens. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (2023)*, pp. 22972–22981, 2023.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(8):1798–1828, 2013.

Biza, O., Van Steenkiste, S., Sajjadi, M. S., Elsayed, G. F., Mahendran, A., and Kipf, T. Invariant slot attention: Object discovery with slot-centric reference frames. In *International Conference on Machine Learning (ICML)*, 2023.

Brandfonbrener, D., Nachum, O., and Bruna, J. Inverse dynamics pretraining learns good representations for multitask imitation. *Advances in Neural Information Processing Systems*, 36, 2024.

Bruce, J., Dennis, M. D., Edwards, A., Parker-Holder, J., Shi, Y., Hughes, E., Lai, M., Mavalankar, A., Steigerwald, R., Apps, C., et al. Genie: Generative interactive environments. In *International Conference on Machine Learning (ICML)*, 2024.

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. Monet: Unsupervised scene decomposition and representation. *arXiv:1901.11390*, 2019.

Cabi, S., Colmenarejo, S. G., Novikov, A., Konyushkova, K., Reed, S., Jeong, R., Zolna, K., Aytar, Y., Budden, D., Vecerik, M., et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. In *Robotics: Science and Systems (RSS)*, 2020.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

Creswell, A., Kabra, R., Burgess, C., and Shanahan, M. Unsupervised object-based transition models for 3D partially observable environments. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Cui, Z. J., Pan, H., Iyer, A., Haldar, S., and Pinto, L. DynaMo: In-Domain Dynamics Pretraining for Visuo-Motor Control. *arXiv preprint arXiv:2409.12192*, 2024.

Daniel, T. and Tamar, A. DDLP: Unsupervised Object-centric Video Prediction with Deep Dynamic Latent Particles. *Transactions on Machine Learning Research (TMLR)*, 2024.

Denton, E. and Fergus, R. Stochastic video generation with a learned prior. In *International Conference on Machine Learning (ICML)*, 2018.

Dittadi, A., Papa, S., De Vita, M., Schölkopf, B., Winther, O., and Locatello, F. Generalization and robustness implications in object-centric learning. In *International Conference on Machine Learning (ICML)*, 2022.

Edwards, A., Sahni, H., Schroecker, Y., and Isbell, C. Imitating latent policies from observation. In *International Conference on Machine Learning (ICML)*, 2019.

Elsayed, G. F., Mahendran, A., van Steenkiste, S., Greff, K., Mozer, M. C., and Kipf, T. SAVi++: Towards end-to-end object-centric learning from real-world videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Ferraro, S., Mazzaglia, P., Verbelen, T., and Dhoedt, B. Focus: Object-centric world models for robotics manipulation. *Advances in Neural Information Processing Systems Workshops (NeurIPSw)*, 2023.

Greff, K., Van Steenkiste, S., and Schmidhuber, J. On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*, 2020.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Jiang, J., Deng, F., Singh, G., and Ahn, S. Object-centric slot diffusion. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Jiang, J., Deng, F., Singh, G., Lee, M., and Ahn, S. SlotSSMs: Slot State Space Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Johnson, S. P. Object perception. In *Oxford Research Encyclopedia of Psychology*. 2018.

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

Kingma, D. P. and Ba, J. A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Kipf, T., Elsayed, G. F., Mahendran, A., Stone, A., Sabour, S., Heigold, G., Jonschkowski, R., Dosovitskiy, A., and Greff, K. Conditional Object-Centric Learning from Video. In *International Conference on Learning Representations (ICLR)*, 2022.

Li, R., Jabri, A., Darrell, T., and Agrawal, P. Towards practical multi-object manipulation using relational reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., and Bachem, O. Challenging common assumptions in the unsupervised learning of disentangled representations. In *International Conference on Machine Learning (ICML)*, pp. 4114–4124, 2019.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. Object-centric learning with slot attention. In *International Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Mamaghan, A. M. K., Papa, S., Johansson, K. H., Bauer, S., and Dittadi, A. Exploring the effectiveness of object-centric representations in visual question answering: Comparative insights with foundation models. *arXiv preprint arXiv:2407.15589*, 2024.

Menapace, W., Lathuiliere, S., Tulyakov, S., Siarohin, A., and Ricci, E. Playable video generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Menapace, W., Lathuilière, S., Siarohin, A., Theobalt, C., Tulyakov, S., Golyanik, V., and Ricci, E. Playable environments: Video manipulation in space and time. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Meo, C., Nakano, A., Lică, M., Didolkar, A., Suzuki, M., Goyal, A., Zhang, M., Dauwels, J., Matsuo, Y., and Bengio, Y. Object-centric temporal consistency via conditional autoregressive inductive biases. *arXiv preprint arXiv:2410.15728*, 2024.

Mosbach, M., Niklas Ewertz, J., Villar-Corrales, A., and Behnke, S. SOLD: Reinforcement Learning with Slot Object-Centric Latent Dynamics. In *arXiv preprint arXiv:2410.08822*, 2024.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. In *International Conference on Neural Information Processing Systems Workshops (NeurIPS-W)*, 2017.

Schmidt, D. and Jiang, M. Learning to act without actions. In *International Conference on Learning Representations (ICLR)*, 2024.

Seitzer, M., Horn, M., Zadaianchuk, A., Zietlow, D., Xiao, T., Simon-Gabriel, C.-J., He, T., Zhang, Z., Schölkopf, B., Brox, T., et al. Bridging the gap to real-world object-centric learning. In *International Conference on Learning Representations (ICLR)*, 2023.

Singh, G., Deng, F., and Ahn, S. Illiterate dall-e learns to compose. *arXiv preprint arXiv:2110.11405*, 2021.

Singh, G., Wu, Y.-F., and Ahn, S. Simple unsupervised object-centric learning for complex and naturalistic videos. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:18181–18196, 2022.

Struckmeier, O. and Kyrki, V. ILPO-MP: Mode Priors Prevent Mode Collapse when Imitating Latent Policies from Observations. *Transactions on Machine Learning Research (TMLR)*, 2023.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

Van Den Oord, A. and Vinyals, O. Neural discrete representation learning. 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

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

Wang, X., Li, X., Hu, Y., Zhu, H., Hou, C., Lan, C., and Chen, Z. Tiv-diffusion: Towards object-centric movement for text-driven image to video generation. *arXiv preprint arXiv:2412.10275*, 2024.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

Watters, N., Matthey, L., Burgess, C. P., and Lerchner, A. Spatial broadcast decoder: A simple architecture for disentangled representations in VAEs, 2019. URL https://openreview.net/forum?id=S1x7WjnzdV.

Wu, Z., Dvornik, N., Greff, K., Kipf, T., and Garg, A. SlotFormer: Unsupervised visual dynamics simulation with object-centric models. In *International Conference on Learning Representations (ICLR)*, 2023a.

Wu, Z., Hu, J., Lu, W., Gilitschenski, I., and Garg, A. Slotdiffusion: Object-centric generative modeling with diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 50932–50958, 2023b.

Ye, S., Jang, J., Jeon, B., Joo, S., Yang, J., Peng, B., Mandlekar, A., Tan, R., Chao, Y.-W., Lin, B. Y., et al. Latent action pretraining from videos. *International Conference on Learning Representations (ICLR)*, 2024.

Ye, W., Zhang, Y., Abbeel, P., and Gao, Y. Become a proficient player with limited data through watching pure videos. In *International Conference on Learning Representations (ICLR)*, 2022.

Yoon, J., Wu, Y.-F., Bae, H., and Ahn, S. An investigation into pre-training object-centric representations for reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, pp. 1094–1100, 2020.

Zadaianchuk, A., Seitzer, M., and Martius, G. Object-centric learning for real-world videos by predicting temporal feature similarities. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2024.

Zhang, C., Gupta, A., and Zisserman, A. Is an object-centric video representation beneficial for transfer? In *Asian Conference on Computer Vision (ACCV)*, pp. 1976–1994, 2022.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595, 2018.

Zoran, D., Kabra, R., Lerchner, A., and Rezende, D. J. Parts: Unsupervised segmentation with slots, attention and independence maximization. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10439–10447, 2021.

A. Limitations & Future Work

We recognize two main limitations that currently limit the scope of our PlaySlot framework to simple tabletop robotic scenarios, preventing it from generalizing to more complex domains.

Limited Decomposition Model The first limitation arises from the SAVi object-centric decomposition model used at the core of our framework. SAVi achieves a great decomposition performance on datasets with objects of simple shapes and textures. However, it fails to generalize to complex real-world robotic scenarios, hence limiting the current scope of PlaySlot to robotic tabletop simulations, or simple real-world environments as in Sketchy.

Single Latent Action The second limitation lies at the representational capability of our latent actions and action prototypes. In robotic scenarios, several actions often happen simultaneously, such as moving and rotating the robot, as well as opening or closing the gripper. Our current latent action representation jointly models the scene's inverse dynamics, encoding together all actions into a single latent space. This entangled representation limits our ability to control the agents in the scene with greater precision.

Future Work In future work, we plan to extend our proposed PlaySlot framework with more capable decomposition models, such as DINOSAUR (Seitzer et al., 2023) or SOLV (Aydemir et al., 2023), as well as scale our inverse dynamics and predictor models. Furthermore, we can employ factorized latent action vectors, which represent in a disentangled manner different actions that happen simultaneously, such as moving the robot arm and opening the gripper. We believe that this architectural modifications will enable us to use PlaySlot on more complex robotic simulations and perform real-world robotic experiments.

B. Implementation Details

In this section, we describe the network architecture and training details for each of the components in our PlaySlot framework. Our models are implemented in PyTorch (Paszke et al., 2017) and are trained on a single NVIDIA A100 GPU.

B.1. Object-Centric Learning

We closely follow Kipf et al. (2022) for the implementation of the SAVi object-centric decomposition model, which we employ as scene parsing and object rendering modules. We strictly adhere to the architecture of their proposed CNN-based image encoder $\mathcal{E}_{\text{SAVi}}$, slot decoder $\mathcal{D}_{\text{SAVi}}$, transformer-based dynamics transition module, and Slot Attention corrector. We use a variable number of 128-dimensional object slots, depending of the dataset. Namely, we employ eight object slots on BlockPush, four object slots on ButtonPress, and three object slots on GridShapes. On all datasets, we sample the initial object slots $\mathbf{S}_0$ from a Gaussian distribution with learned mean and covariance. Furthermore, we use three Slot Attention iterations for the initial video frame to obtain a good initial object-centric decomposition, and a single iteration for subsequent frames, which suffices to recursively update the slot representation state given the newly observed image features.

B.2. Inverse Dynamics Module

We propose two variants of our InvDyn module.

InvDyn_S: InvDyn_S jointly processes the objects slots from a single time step $\mathbf{S}_t$ along with an additional token `[ACT]` using a transformer encoder. We use a four-layer transformer encoder with a 256-dimensional tokens, four 64-dimensional heads and hidden dimension of 1024. This module aggregates information from the object slots into the `[ACT]` token, and outputs a single latent action $\hat{\mathbf{z}}_t$ that captures the agent's action, making it well-suited for single-agent environments.

InvDyn_M: InvDyn_M independently processes each object slot with a shared MLP, thus generating $N_\mathbf{S}$ latent action embeddings, each representing the action of a specific object in the scene. This design makes InvDyn_M well-suited for environments with multiple moving agents. We employ a two-layer MLP, featuring a ReLU nonlinear activation and layer normalization.

As described in Sec. 3.2.2, the generated latent actions $\hat{\mathbf{z}}_t$ are vector-quantized to assign them to their corresponding action prototype $\mathbf{p}_t$. On the ButtonPress, BlockPush, and Sketchy datasets, we use the InvDyn_S variant with eight different

16-dimensional action prototypes, whereas for GridShapes we use InvDyn_M with five distinct eight-dimensional action prototypes. Following common practice, we update the action prototypes using the exponential moving average updates of cluster assignment counts (Van Den Oord & Vinyals, 2017).

B.3. Conditional Object-Centric Predictor

Our conditional object-centric predictor (cOCVP) is inspired by the transformer-based SlotFormer (Wu et al., 2023a) architecture. Our cOCVP module features four layers, 256-dimensional tokens, eight 64-dimensional attention heads, and hidden dimension of 512.

To enable predictions conditioned on the inferred latent actions, cOCVP maps the action prototypes $\mathbf{p}_{1:t}$, variability embeddings $\boldsymbol{\sigma}^2_{\mathbf{z}1:t}$ and object slots into the token dimensionality. The slots are then conditioned by adding them with the projected action prototype and variability embedding from the corresponding time step. Furthermore, following Wu et al. (2023a), we augment these representations with a temporal sinusoidal positional encoding.

B.4. Policy Model and Action Decoder

The policy model f_π follows a similar architecture to InvDyn_S. f_π is a four layer transformer that jointly processes the objects slots from a single time step $\mathbf{S}_t$ and an additional token `[ACT]` in order to regress a latent action.

The action decoder is a three-layer MLP with a hidden dimension of 128 that maps latent action vectors into real-world actions.

B.5. 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.

InvDyn and cOCVP Training: We jointly train our InvDyn and cOCVPmodules given a pretrained SAVi decomposition model. These modules are 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 $\lambda_\text{Img} = 1$, $\lambda_\text{Slot} = 1$, and $\lambda_\text{VQ} = 0.25$.

f_π and $\mathcal{D}_\mathbf{a}$ Training: We train the f_π and $\mathcal{D}_\text{a}$ modules given pretrained and frozen SAVi, InvDyn and cOCVPmodules. These modules are trained with the Adam optimizer (Kingma & Ba, 2015), batch size of 64, and a fixed learning rate of 2×10^{-4}.

C. Dataset Details

C.1. BlockPush

This environment, inspired by Li et al. (2020) and simulated using MuJoCo (Todorov et al., 2012), features a robot arm on a tabletop interacting with multiple uni-colored blocks. We use two different dataset variants.

The first variant consists of a robot controlled by a random exploration policy, moving in the environment with minimal meaningful object interactions. This easily simulated dataset includes 20,000 training sequences and 2,000 validation sequences. We use this dataset variant for training SAVi, as well as for jointly training our inverse dynamics and conditional predictor modules.

The second variant contains a smaller subset of expert demonstrations where the robot is tasked to push the block of distinct color to a target location marked with a red sphere. This task evaluates the capabilities of an agent to reason about object relations and model object collisions. We collect 5,000 expert demonstrations using a pretrained policy (Mosbach et al., 2024) with a success rate of $\approx 80\%$ for this pushing task. We split this dataset into 4,500 training sequences, which are used for training the policy model and action decoder; and 500 evaluation sequences that are used for benchmarking the prediction models.

C.2. ButtonPress

This environment, based on MetaWorld (Yu et al., 2020) features a Sawyer robot arm tasked with pressing a red button. Unlike BlockPush, it involves a non-object-centric task with complex shapes and textures. As before, we use two different dataset variants.

The first variant contains 10,000 sequences, split into 9,000 training and 1,000 validation videos, with the robot controlled by a random exploration policy. We use this dataset variant to train SAVi, as well as our inverse dynamics and conditional predictor modules.

The second variant includes a small subset of expert demonstrations where the robot successfully presses the red button. We collect 1,000 expert demonstrations using an expert policy, from which 900 are used for training the policy model and action decoder, and 100 demonstrations are used for benchmarking the prediction models.

C.3. GridShapes

This dataset features one or more simple 2D shapes moving in a grid-like pattern on top of a colored background. The shapes can be either a ball, triangle or square, and have a random color. These shapes can move up, down, left, right or remain still, and revert their motion when an image boundary is reached, thus emulating a bouncing effect. To introduce some stochasticity into the motion, the shapes randomly change direction with a predefined probability of 0.25. We train and evaluate the models on several variants of the GridShapes dataset featuring different number of objects, ranging from one single moving shape, to five objects moving independently in the same sequence. This simple dataset serves as benchmark to evaluate a model's ability to jointly predict the motion of multiple moving agents in the scene.

D. Baselines

In our experiments, we compare our approach with different baseline models, including the stochastic and playable video prediction models SVG (Denton & Fergus, 2018) and CADDY (Menapace et al., 2021), as well as the object-centric video prediction models SlotFormer (Wu et al., 2023a) and OCVP (Villar-Corrales et al., 2023).

D.1. SVG

SVG (Denton & Fergus, 2018) is a generative model for video prediction that captures both deterministic dynamics and stochastic variations in video sequences. It combines a variational autoencoder (VAE) with recurrent neural networks (RNNs) to model stochastic temporal dynamics. SVG represents a probabilistic framework, where the next frame $\hat{\mathbf{X}}_{t+1}$ is generated based on the previous frame $\mathbf{X}_t$ and a latent sample $\hat{\mathbf{z}}_t$ drawn from a latent distribution. In our experiments, we adapt the original implementation[1] and use an SVG variant with a learned prior, VGG-like encoder and decoders, and two recurrent predictor layers.

Main Difference with PlaySlot: SVG operates with holistic scene representations, whereas our proposed method employs a structured object-centric representation. Furthermore, SVG encodes the stochastic scene dynamics into a single continuous latent vector. In contrast, PlaySlot follows a hybrid approach in which the latent action vectors are composed of an action prototype and action variability embeddings, making the prediction process more controllable and interpretable.

D.2. CADDY

CADDY (Menapace et al., 2022) is a recurrent encoder-decoder model designed for playable video generation, enabling user-controllable future video prediction. CADDY infers latent actions that encode the agent's actions between consecutive pairs of frames. These latent actions are parameterized with a discrete *one-hot action label*, which determines the high-level action taking place; and a high-dimensional *action variability embedding*, which describes the variability of each action and captures the possible non-determinism in the environment. We adapt the original implementation[2] for our experiments.

Main Difference with PlaySlot: CADDY operates with holistic scene representations, i.e. CNN features, whereas our proposed method employs a structured object-centric representation. Moreover, despite both methods using a hybrid parameterization of the latent actions, they differ in their implementation. CADDY learns a discrete one-hot action label

[1] https://github.com/edenton/svg

[2] https://github.com/willi-menapace/PlayableVideoGeneration

by minimizing multiple regularization objectives, including an action matching loss and several Kullback–Leibler (KL) divergences losses. In contrast, PlaySlot employs a discrete set of high-dimensional action prototypes, which are learned via vector-quantization of the latent space. We empirically verify that both approaches achieve comparable performance. However, our vector quantization approach requires significantly fewer hyper-parameters and is easier to tune.

D.3. SlotFormer

SlotFormer (Wu et al., 2023a) is an object-centric video prediction model that builds upon slot-based representations. First, it uses Slot Attention to decompose an image into object-centric latent representations, called slots. SlotFormer then employs a transformer-based autoregressive predictor module, which jointly processes all input slots in order to forecast future object representations. Finally, the predicted slots are decoded into object images and video frames. In our experiments, we adapt the original implementation[3].

Main Difference with PlaySlot: SlotFormer forecasts future slots in an unconditional manner, thus not being able to model stochastic environments or agents such as robots. PlaySlot addresses this challenge by inferring the scene inverse dynamics and using them to condition the prediction process.

D.4. OCVP

OCVP (Villar-Corrales et al., 2023), similar to SlotFormer, is a slot-based object-centric video prediction model. Differing from SlotFormer, OCVP leverages to specialized decoupled attention mechanisms, *relation* and *temporal* attention, which model the object interactions and dynamics, respectively. In our experiments, we use the OCVP-Seq variant with default settings adapted from original implementation[4].

Main Difference with PlaySlot: OCVP forecasts future slots in an unconditional manner, thus not being able to model stochastic environments or agents such as robots. PlaySlot addresses this challenge by inferring the scene inverse dynamics and using them to condition the prediction process.

E. Additional Results

E.1. Effect of the Number of Actions

In Tab. 3 we evaluate on the BlockPush dataset multiple PlaySlot variants using a different number of learned action prototypes. We show that using eight learned action prototypes achieves the best video prediction performance, while learning a concise semantically meaningful set of action prototypes.

Table 3: Evaluation of PlaySlot variants on the BlockPush dataset using a different number of learned action prototypes.

	BlockPush		
# Actions	PSNR↑	SSIM↑	LPIPS↓
5	21.26	0.886	0.071
8	**21.41**	**0.890**	**0.066**
10	21.36	<u>0.889</u>	<u>0.067</u>
15	21.38	<u>0.889</u>	<u>0.067</u>

E.2. Learned Behaviors from Expert Demonstrations

We evaluate the ability of PlaySlot to learn robot behaviors from a small set of expert demonstrations. As outlined in Sec. 4.4, we train a policy model and an action decoder to imitate ButtonPress and BlockPush behaviors from a small set of expert demonstrations. We compare PlaySlot to oracle baselines that have direct access to expert action labels. This comparison allows us to assess the effectiveness of object-centric representations and inferred latent actions for downstream robotic tasks.

ButtonPress Behavior: We quantitatively compare our learned policy model, which imitates the ButtonPress behavior, against oracle baselines. These models have access to all available expert demonstrations and directly regress the ground-

[3] https://github.com/pairlab/SlotFormer

[4] https://github.com/AIS-Bonn/OCVP-object-centric-video-prediction

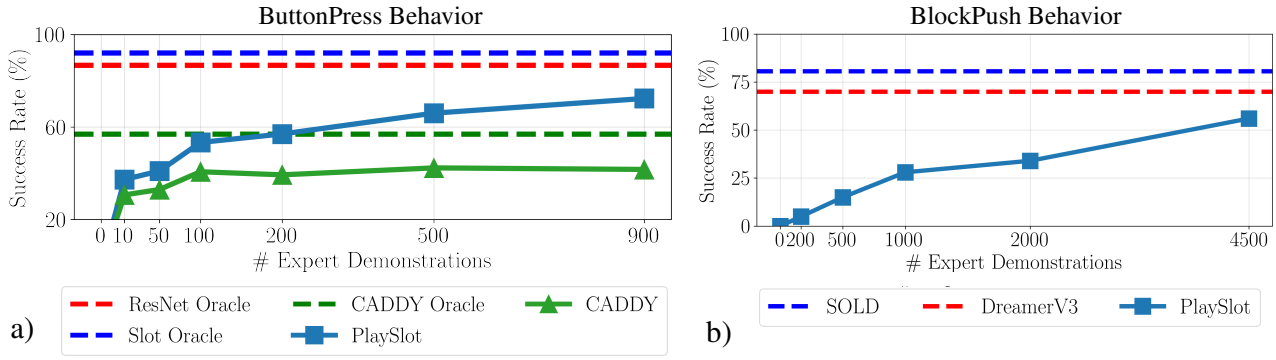


Figure 8: Success rate (%) as a function of the number of expert demonstrations for different models on the a) ButtonPress and b) BlockPush environments. Object-centric models (Slot Oracle, PlaySlot and SOLD) outperform their non-object centric counterparts. PlaySlot consistently improves with more expert demonstrations, outperforming CADDY.

truth actions from object-centric representations (Slot Oracle), ResNet feature maps (ResNet Oracle), and from feature maps output by the CADDY encoder (CADDY Oracle), respectively. Unlike these oracles, PlaySlot autoregressively predicts latent actions within its latent imagination before decoding them into executable actions. Additionally, we compare with a modified CADDY variant, which includes a small policy model and action decoder designed output latent actions from observed feature maps and decode them into real-world actions, respectively.

The results, shown in Fig. 8a), demonstrate that PlaySlot consistently outperforms CADDY across all training regimes. With as few as 200 demonstrations, PlaySlot achieves comparable performance to the CADDY Oracle, which has direct access to more expert demonstrations and ground-truth actions. As the number of demonstrations increases, PlaySlot continues to improve, approaching the performance of the oracle models, while CADDY struggles to generalize. These findings highlight the effectiveness of PlaySlot's object-centric representations in capturing meaningful action dynamics, enabling efficient behavior learning from limited expert data. Additionally, we observe that slot-based object-centric models outperform their holistic counterparts, further emphasizing the advantage of structured object-centric representations for learning robot behaviors.

Fig. 9 illustrates PlaySlot's learned behavior in the ButtonPress environment. The top row in each sequence (labeled as *Latent Behavior*) shows predicted trajectories within PlaySlot's latent imagination, where the model autoregressively generates actions using the policy model and predicts future scene states in the latent space. The bottom row (labeled *Sim. Actions*) depicts the simulated execution of the decoded latent actions in the environment. PlaySlot learns to solve the task within its latent imagination, successfully reasoning about the required action sequences before translating its latent actions into executable motions. Fig. 9b) shows a failure case where PlaySlot predicts within its latent imagination a trajectory that leads to successfully pressing the button. However, accumulated errors during action decoding cause the simulated execution to miss the button.

BlockPush Behavior: We quantitatively evaluate PlaySlot on the challenging BlockPush task, which requires reasoning about specific object properties. In this task, our learned policy model imitates the behavior based on expert demonstrations, with only approximately 80% of these demonstrations being successful. The imperfect nature of the expert data adds an extra layer of difficulty to the learning process.

We compare PlaySlot against two model-based reinforcement learning models with holistic (DreamerV3 (Hafner et al., 2023)) and object-centric (SOLD (Mosbach et al., 2024)) latent spaces, both of which learn the robot behavior by interacting with the environment. The results are shown in Fig. 8b). As the number of demonstrations increases, PlaySlot closes the gap with the baseline models, demonstrating its ability to generalize effectively from limited data.

Fig. 11 illustrates PlaySlot's learned behavior in the BlockPush environment. The top row in each sequence (labeled as *Latent Behavior*) shows predicted trajectories within PlaySlot's latent imagination, where the model autoregressively generates actions using the policy model and predicts future scene states in the latent space. The bottom row (labeled *Sim. Actions*) depicts the simulated execution of the decoded latent actions in the environment. PlaySlot learns to solve the task within its latent imagination, successfully reasoning about object properties and generating a precise sequence of latent

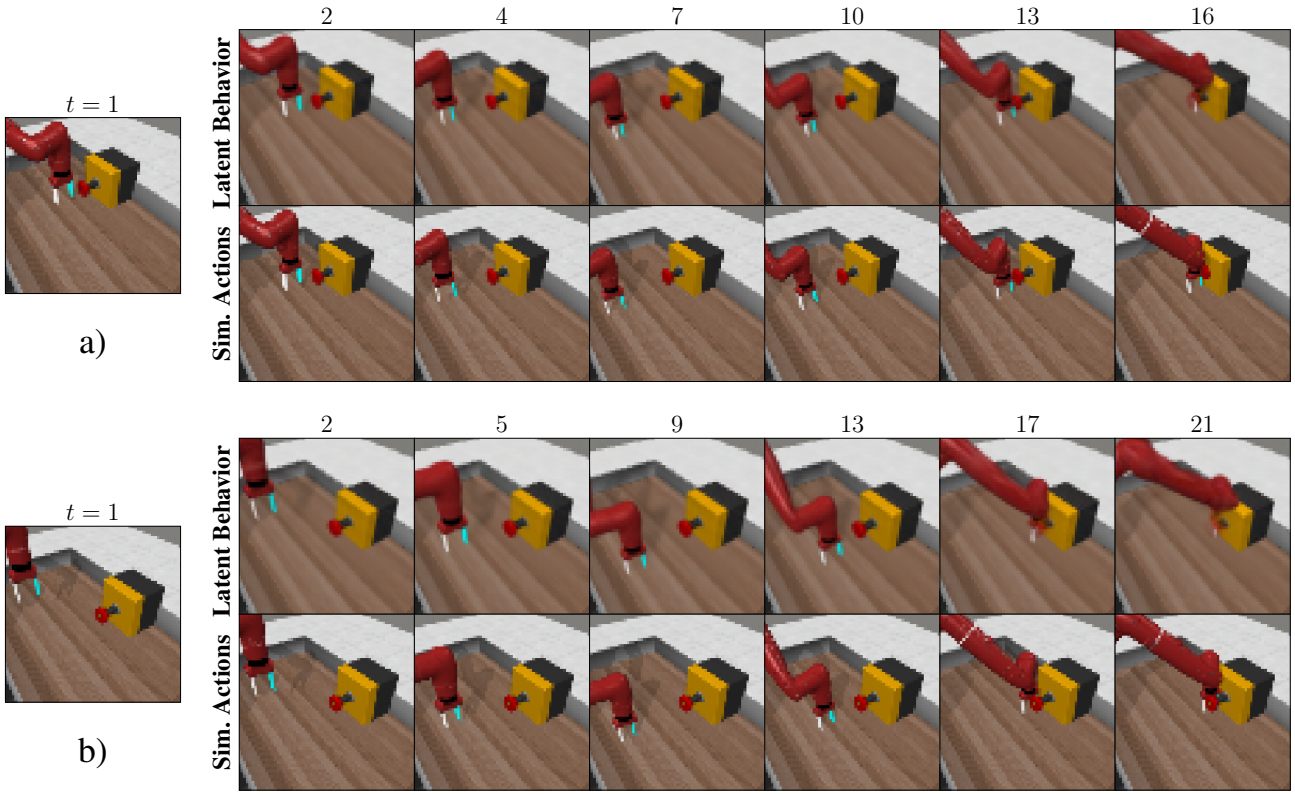


Figure 9: Predicted frames using latent actions from the learned policy, and sequences simulated by executing the decoded latent actions for two ButtonPress scenarios. **a)** PlaySlot successfully plans the trajectory within its latent imagination and translates latent actions into executable motions. **b)** PlaySlot fails to decode its latent actions into correct robot motion, missing the button.

actions, which can be decoded into executable motions. Fig. 11c) shows a failure case where PlaySlot controls the robot to interact with the correct block, but fails to place it in the target location.

E.3. Qualitative Results

E.3.1. Comparison with Baselines

Fig. 10 depicts a qualitative comparison between SVG, CADDY and PlaySlot on the ButtonPress and BlockPush datasets, respectively. On the ButtonPress dataset, as shown in Fig. 10a), all methods accurately model the trajectory of the robot arm. However, on the complex BlockPush task, depicted in Fig. 10b), SVG and CADDY fail to model the object collisions, failing to move the block of distinct color to the target location, and leading disappearing objects. In contrast, PlaySlot maintains sharp object representations and correctly models interactions between objects, leading to accurate frame predictions.

E.3.2. BlockPush Dataset

Fig. 12 shows two qualitative comparisons between PlaySlot, CADDY and SVG on the BlockPush dataset. Our proposed method, which explicitly models object interactions, preserves sharp object representations and accurately predicts future frames. In contrast, SVG and CADDY, which rely on holistic scene features for forecasting, struggle to model object collisions, resulting in blurry predictions and disappearing objects.

Fig. 13 shows the predicted video frames, slot masks, and objects representations on a BlockPush sequence. PlaySlot parses the scene into precise object images and masks, which can be assigned a unique color to obtain a segmentation of the scene. Our approach decomposes the BlockPush environment using eight object slots, where one slot represents the background, five slots to different blocks, one slot to the red target, and one slot to the robot arm. The sharp object images

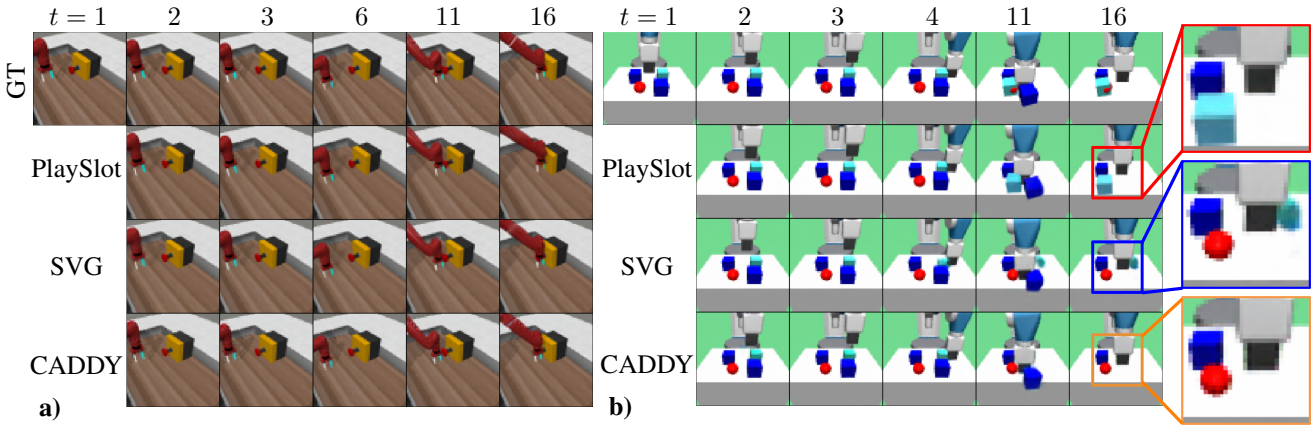


Figure 10: Qualitative comparison on (a) ButtonPress and (b) BlockPush datasets. Our method accurately predicts the scene dynamics, whereas the baselines fail to model object physics and interactions, leading to disappearing objects and failing to predict the pushing of the cyan block to the target location.

and masks demonstrate that PlaySlot encodes into each slot features from the corresponding object. This allows our method to directly reason about object properties, dynamics and interactions, allowing for accurate future frame predictions.

Fig. 14 depicts the effect each action prototype learned by PlaySlot on the BlockPush dataset. PlaySlot learns consistent semantically meaningful action prototypes that control the robot arm to move in a specific direction. We note that some actions prototypes, e.g, action 5 and 7, perform semantically similar actions but with different velocities.

E.3.3. BUTTONPRESS DATASET

Fig. 15 shows a comparisons between PlaySlot, CADDY and SVG on the ButtonPress dataset. All methods successfully predict the ground truth sequence by inferring and modeling the robot's dynamics,

Fig. 16 shows PlaySlot's predictions and object representations on a ButtonPress sequence. We visualize the ground truth sequence, the predicted frames, segmentation obtained by assigning a different color to each slot mask, as well as the object reconstructions for four slots. PlaySlot assigns one slot to the background, one slot for box and red button, and two slots for different parts of the robot arm.

Fig. 17 depicts the effect each action prototype learned by PlaySloton the ButtonPress dataset.

E.3.4. GRIDSHAPES DATASET

Fig. 18 depicts the effect each action prototype learned by PlaySlot on a variant of GridShapes with three shapes. PlaySlot successfully captures the five possible object movements—up, down, left, right, and stay—predicting future frames by modeling the motion of each object independently. However, we observe that PlaySlot sometimes generates artifacts when shapes reach the image boundaries. We attribute this to the training data, where objects change direction upon reaching the boundary mimicking a bouncing effect. This leads to poor prediction performance when conditioning the model to predict outside its training distribution.

E.3.5. SKETCHY DATASET

Fig. 19 shows PlaySlot's predictions and object representations on a Sketchy sequence. We visualize the ground truth sequence, the predicted frames, segmentation obtained by assigning a different color to each slot mask, as well as the object reconstructions for four slots. PlaySlot assigns two slots to the workspace and background, two slots for each part of the robot gripper, and two slots for different objects present in the scene.

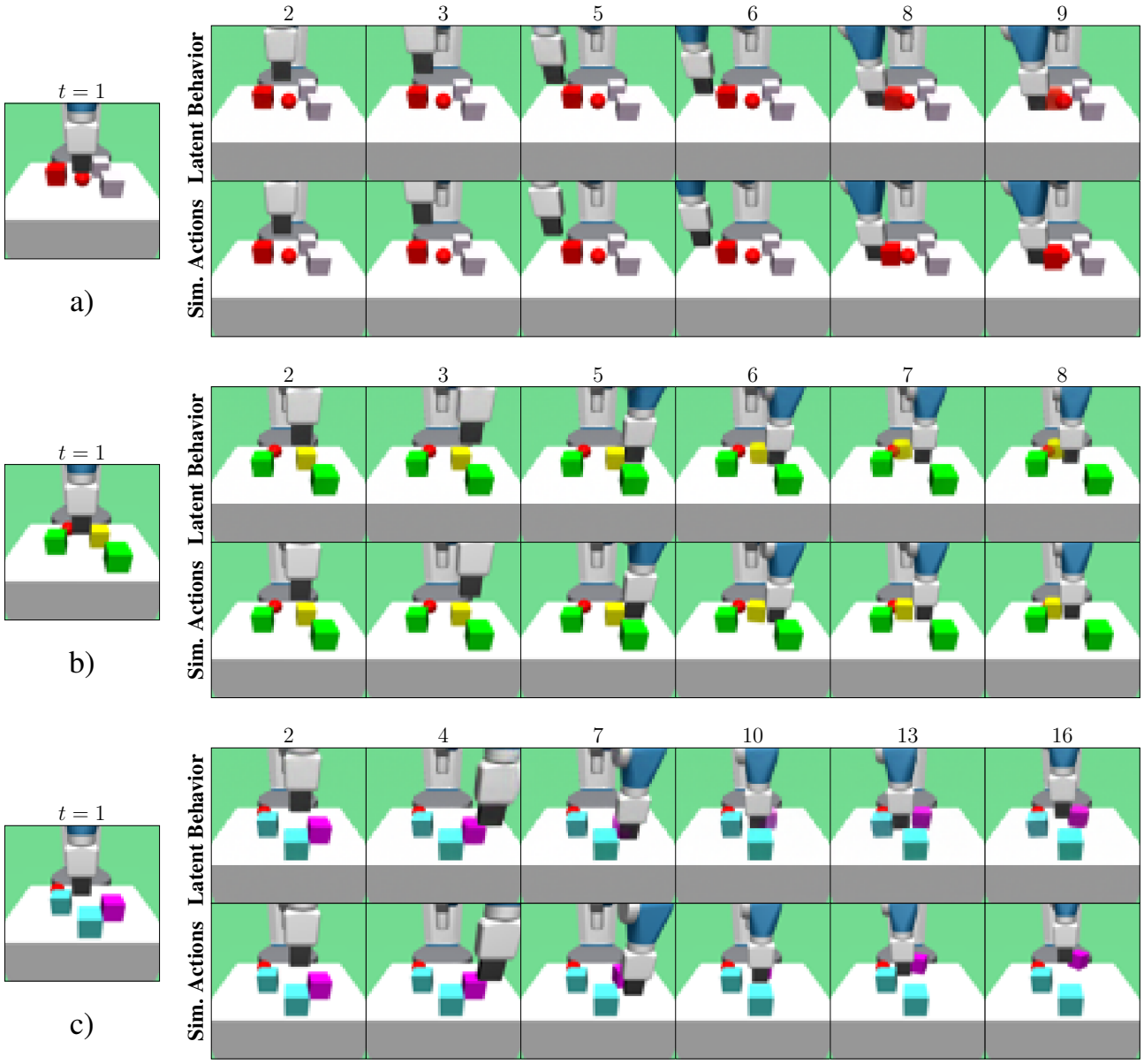


Figure 11: Predicted frames using latent actions from the learned policy, and simulation computed by executing the decoded latent actions for three BlockPush scenarios. **a) & b)** PlaySlot identifies the block of distinct color and generates executable latent actions to push it to the target location. **c)** PlaySlot identifies the block but fails to push it into the target location.

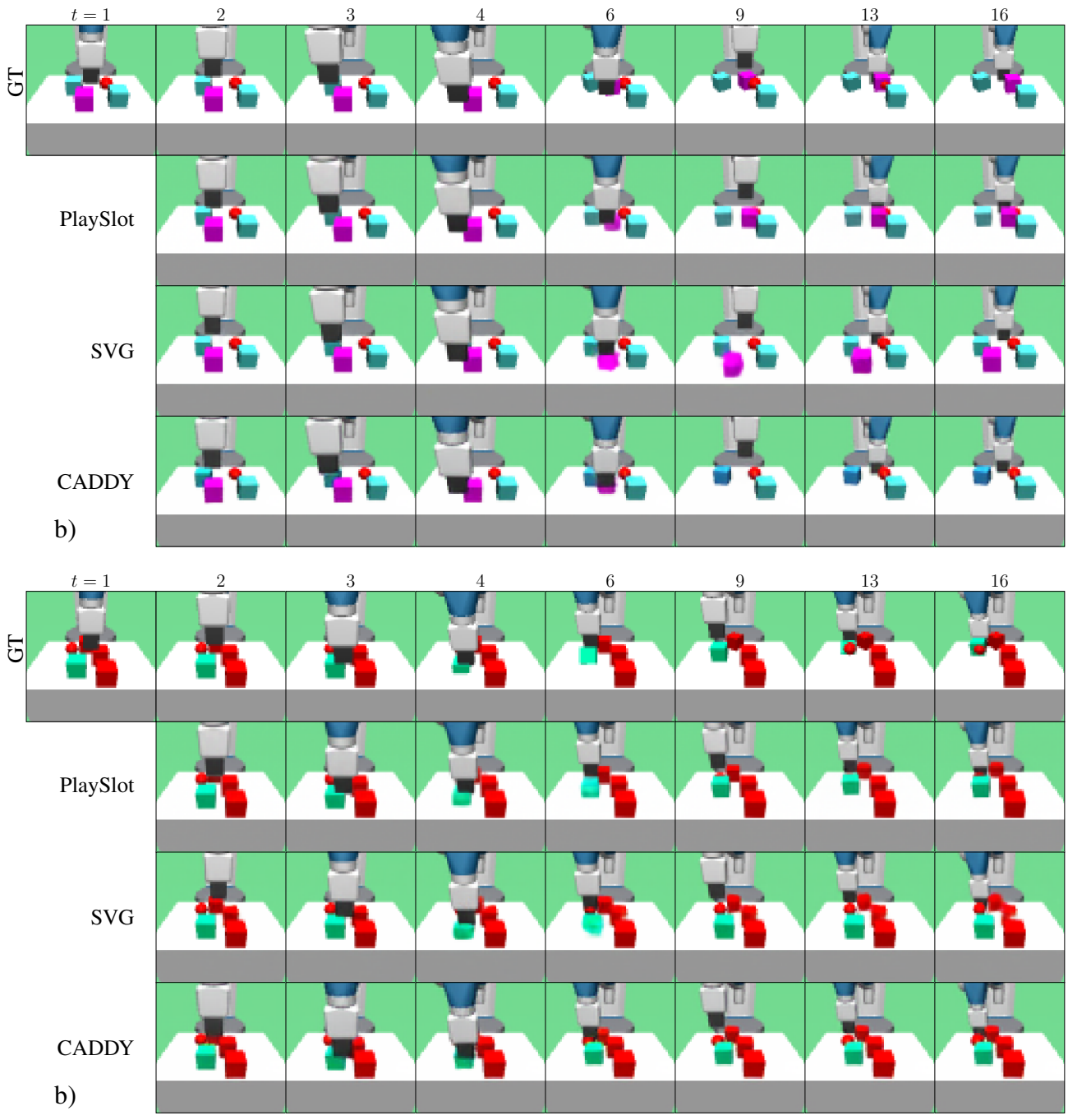


Figure 12: Qualitative comparison on BlockPush. Our method accurately predicts the scene dynamics and object interactions, whereas the baselines fail to model object collisions, leading to blurry or disappearing objects.

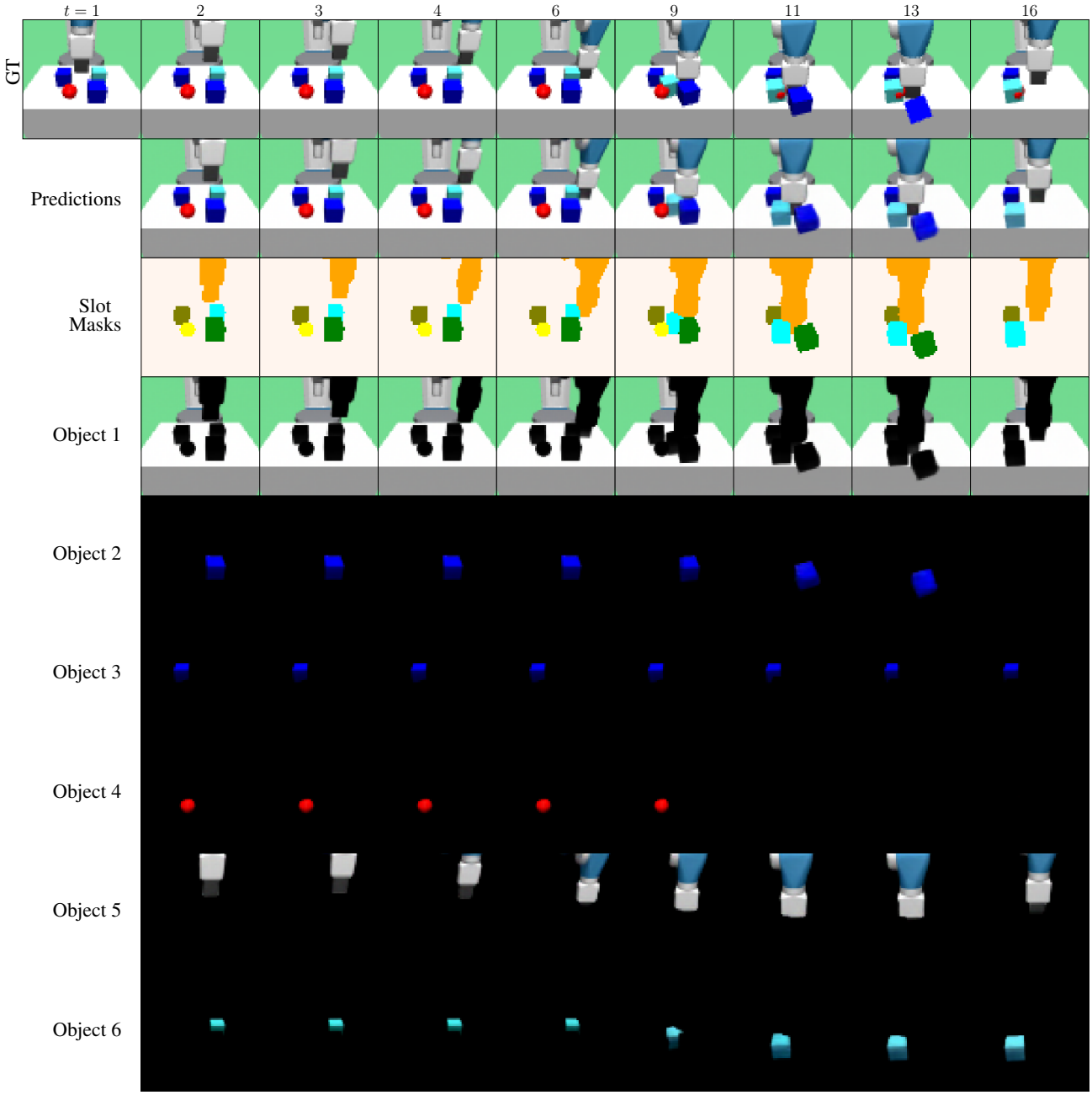


Figure 13: PlaySlot predictions and object representations on a BlockPush sequence. We visualize the ground truth sequence, the predicted frames, segmentation obtained by assigning a different color to each slot mask, as well as the object reconstructions for five slots. PlaySlot assigns one slot to the background, one slot for the robot, one slot for the red target, and the remaining slots to the blocks.

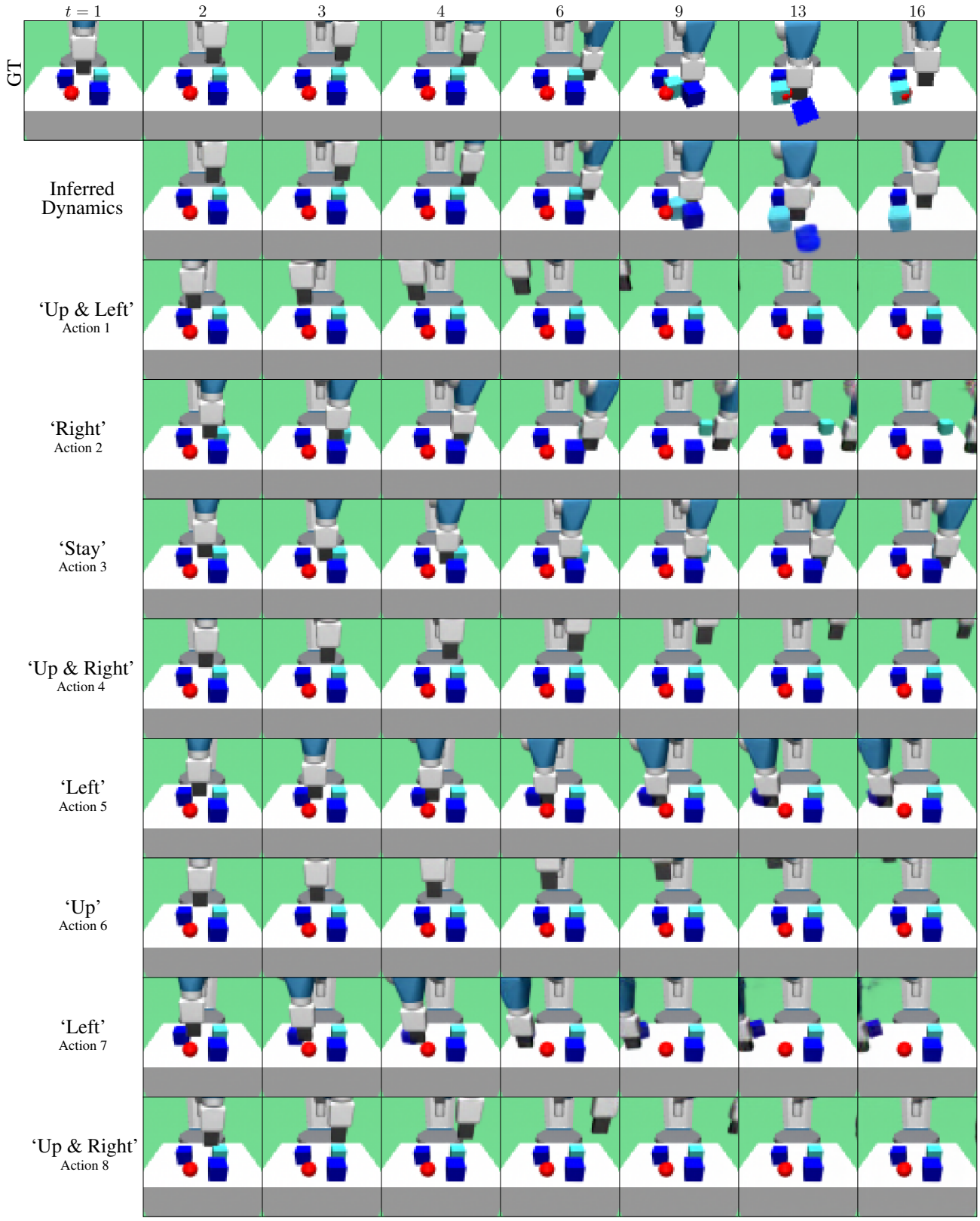


Figure 14: PlaySlot predictions conditioned on different latent actions, including the inferred inverse dynamics, as well as each action prototypes learned on BlockPush. We generate a sequence by repeatedly conditioning the prediction process on a a single action prototype. The model learns action prototypes that control the robot to move consistently on a specific direction.

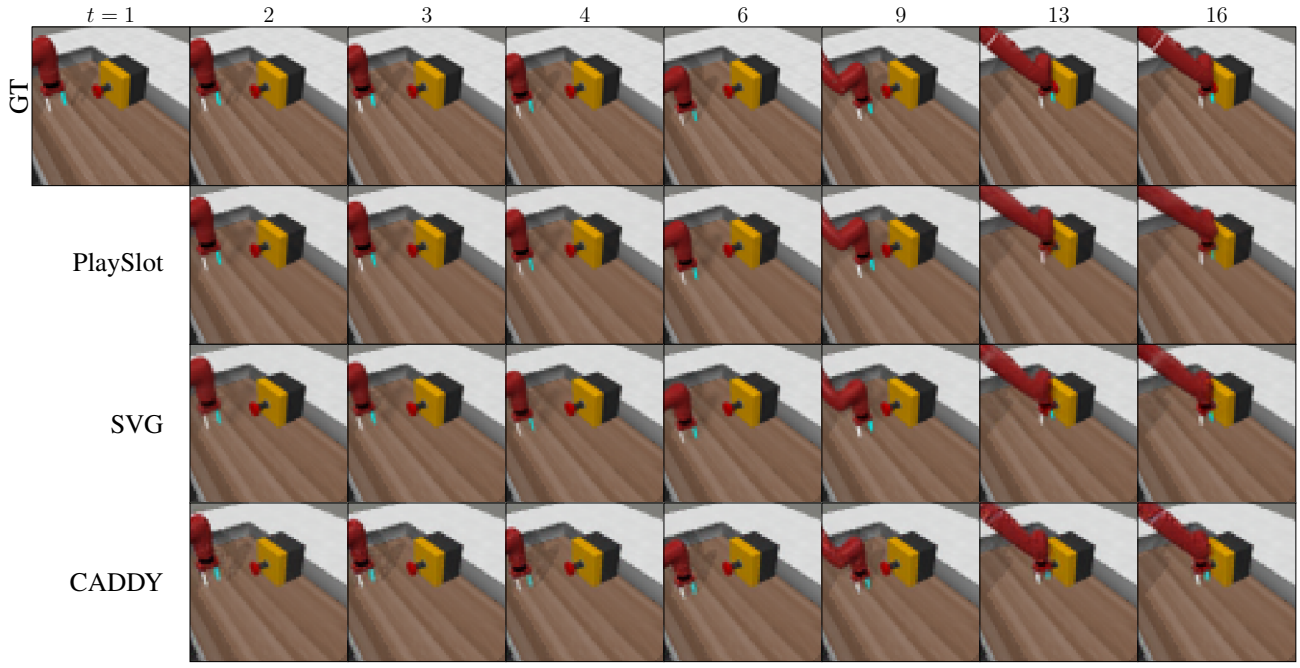


Figure 15: Qualitative comparison on ButtonPress. All methods successfully reconstruct the ground truth sequence by inferring the robot's trajectory.

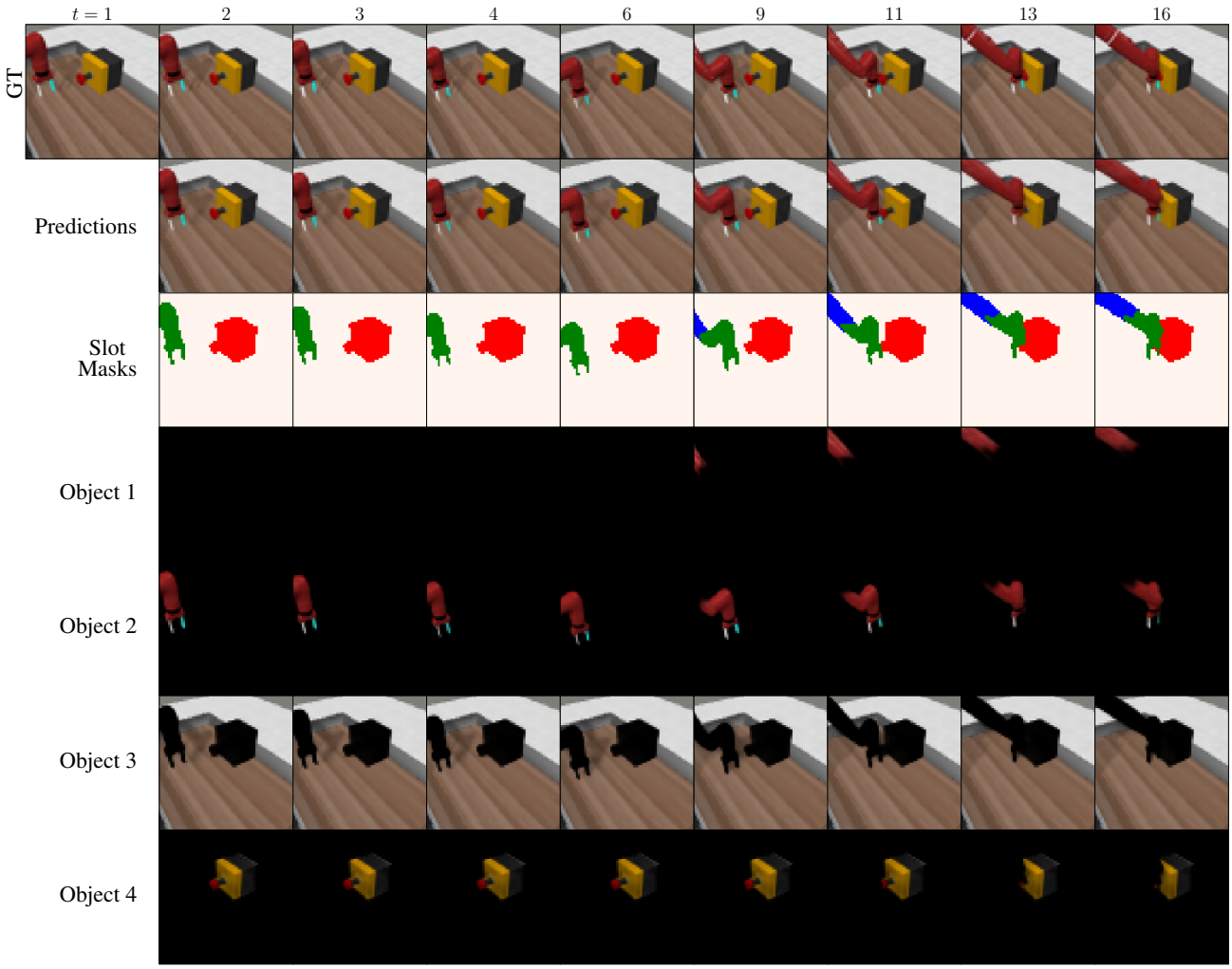


Figure 16: PlaySlot predictions and object representations on a ButtonPress sequence. We visualize the ground truth sequence, the predicted frames, segmentation obtained by assigning a different color to each slot mask, as well as the object reconstructions for four slots. PlaySlotassigns one slot to the background, one slot for box and red button, and two slots for different parts of the robot arm.

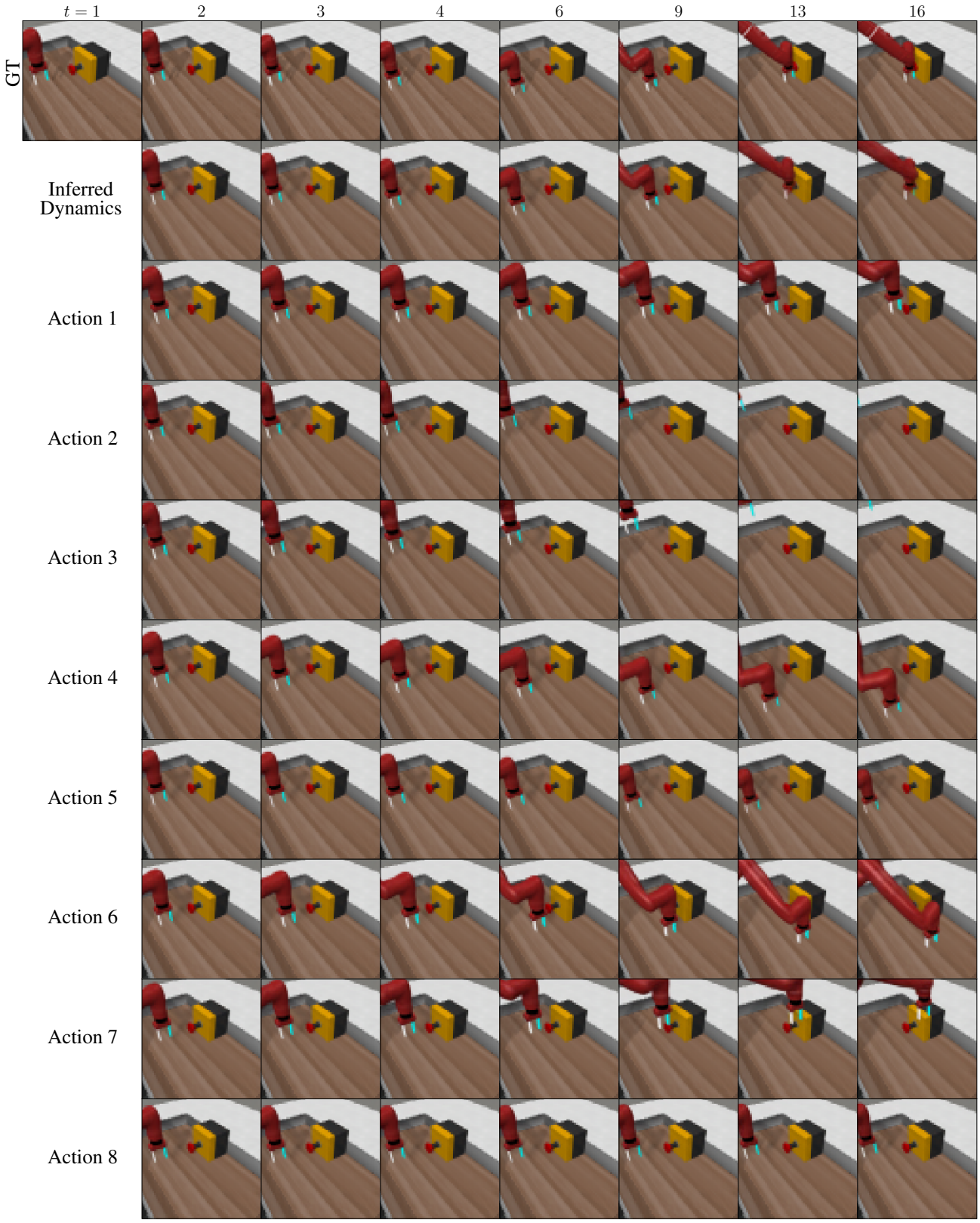


Figure 17: PlaySlot predictions conditioned on different latent actions, including the inferred inverse dynamics, as well as each action prototypes learned on ButtonPress. We generate a sequence by repeatedly conditioning the prediction process on a a single action prototype. The model learns action prototypes that control the robot to move consistently on a specific direction.

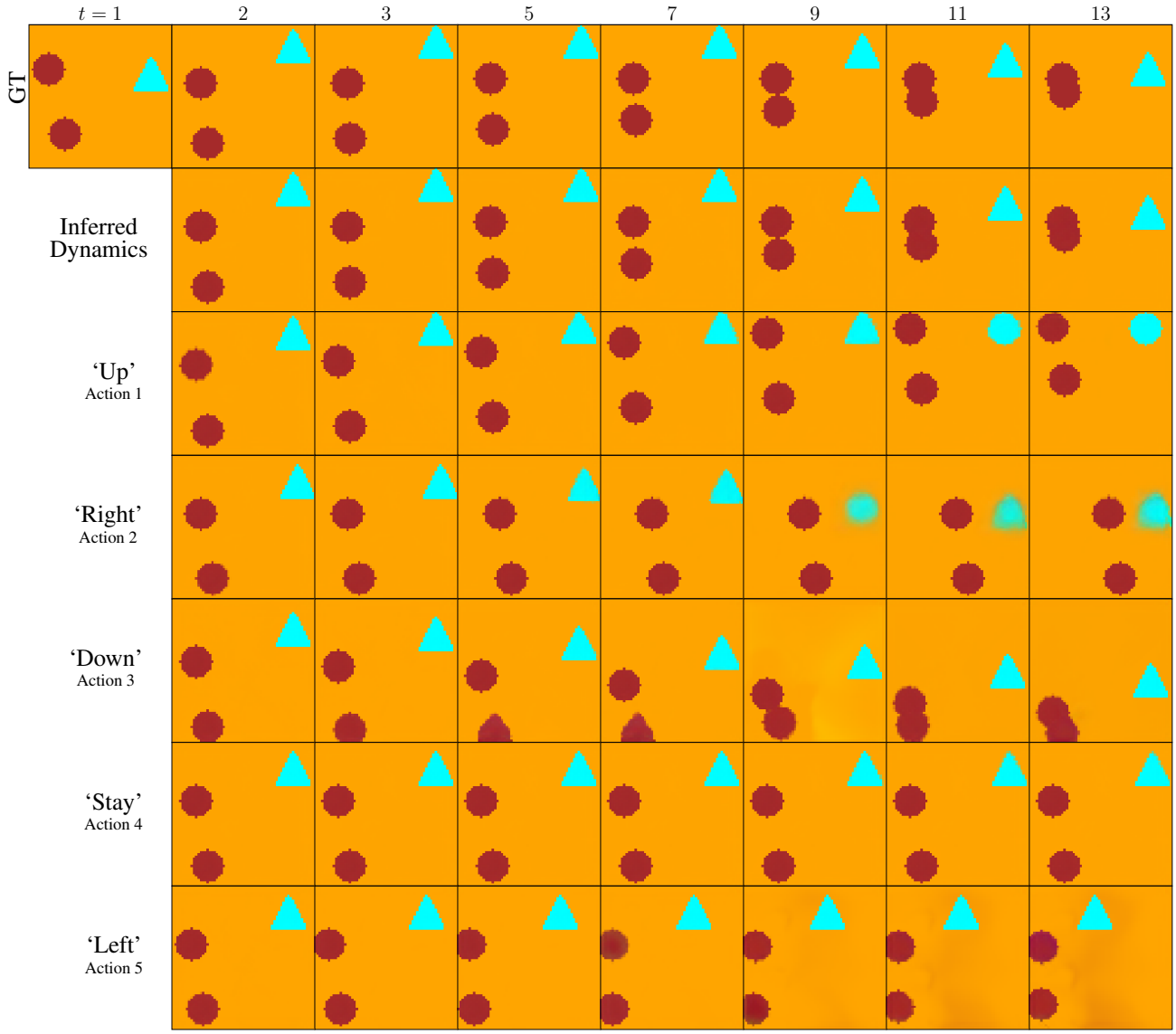


Figure 18: PlaySlot predictions conditioned on different latent actions, including the inferred inverse dynamics, as well as each action prototypes learned on the GridShapes dataset. We generate a sequence by repeatedly conditioning the prediction process on a a single action prototype. The model learns the five possible actions and achieves sharp predictions by forecasting the motion of each object individually.

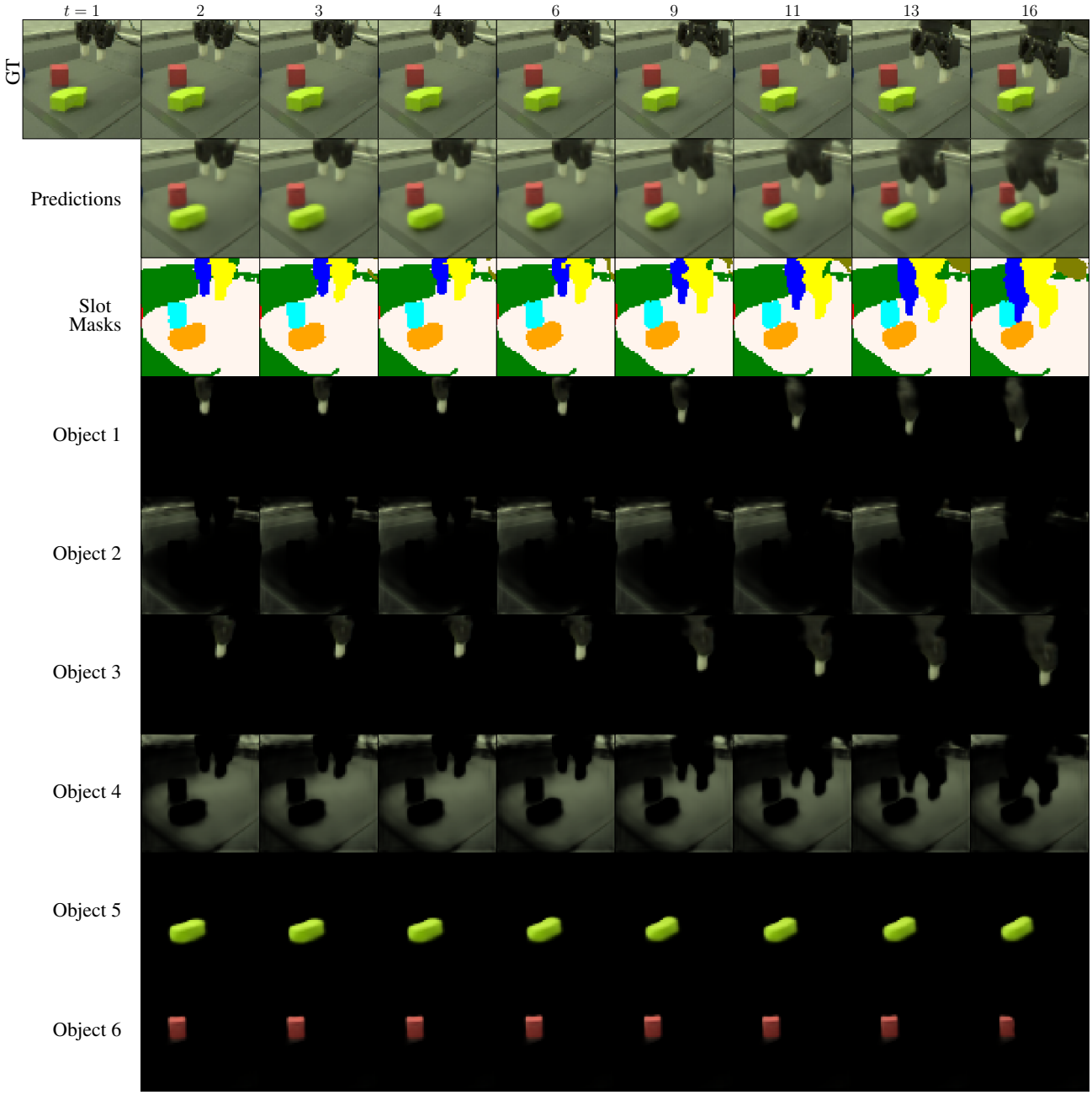


Figure 19: PlaySlot predictions and object representations on a Sketchy sequence. We visualize the ground truth sequence, the predicted frames, segmentation obtained by assigning a different color to each slot mask, as well as the object reconstructions for four slots. PlaySlot assigns two slots to the workspace and background, two slots for each part of the robot gripper, and two slots for different objects present in the scene.