Parallelized Spatiotemporal Slot Binding for Videos

Gautam Singh [1 2] **Yue Wang** [2 3] **Jiawei Yang** [3] **Boris Ivanovic** [2] **Sungjin Ahn** [* 4] **Marco Pavone** [* 2 5] **Tong Che** [2]

Abstract

While modern best practices advocate for scalable architectures that support long-range interactions, object-centric models are yet to fully embrace these architectures. In particular, existing object-centric models for handling sequential inputs, due to their reliance on RNN-based implementation, show poor stability and capacity and are slow to train on long sequences. We introduce *Parallelizable Spatiotemporal Binder* or *PSB*[†], the first temporally-parallelizable slot learning architecture for sequential inputs. Unlike conventional RNN-based approaches, PSB produces object-centric representations, known as slots, for all time-steps *in parallel*. This is achieved by refining the initial slots across all time-steps through a fixed number of layers equipped with causal attention. By capitalizing on the parallelism induced by our architecture, the proposed model exhibits a significant boost in efficiency. In experiments, we test PSB extensively as an encoder within an auto-encoding framework paired with a wide variety of decoder options. Compared to the state-of-the-art, our architecture demonstrates stable training on longer sequences, achieves parallelization that results in a 60% increase in training speed, and yields performance that is on par with or better on unsupervised 2D and 3D object-centric scene decomposition and understanding.

1. Introduction

A key function of the human brain is to translate the incoming stream of sensory inputs into a mental model of the world. Studies suggest that this mental model is compositional, constructed from building blocks such as objects. Furthermore, the process by which this mental model is constructed is systematic, enabling us to interpret unfamiliar environments as a composition of familiar entities (Fodor & Pylyshyn, 1988; Spelke & Kinzler, 2007; Lake et al., 2017). This systematic compositionality is critical for building autonomous agents because it would enable them to understand, plan, and act effectively and robustly in the physical world.

Towards this goal, the most relevant field in deep learning is that of object-centric learning (Greff et al., 2019; Lin et al., 2020b; Locatello et al., 2020; Greff et al., 2020). Object-centric learning aims to learn priors for grouping or binding low-level and unstructured sensory activations into a collection of vectors known as *slots*. Each slot captures a higher-level compositional entity such as an object. Broadly, this grouping or binding emerges as a result of architectural priors combined with self-supervised learning, e.g., by performing auto-encoding with specialized encoder and decoder architectures.

A notion that is widely acknowledged, but not yet fully embraced in object-centric learning, is to maximally utilize all available data with increasingly scalable architectures capable of capturing long-range dependencies (Dosovitskiy et al., 2020; Vaswani et al., 2017). Video data has been widely adopted in self-supervised learning and object-centric learning since videos contain temporal information such as object motion and behavior; learning from videos has been consistently shown to provide richer representations than training on static images (Kosiorek et al., 2018; Kipf et al., 2022; Singh et al., 2022; Feichtenhofer et al., 2022). However, current object-centric learning methods do not utilize the full potential of sequential datasets because of *RNN-based* modeling (Kipf et al., 2022; Singh et al., 2022; Elsayed et al., 2022). RNNs lead to major scaling issues—training instability on longer sequences due to gradient vanishing or exploding leads to degenerated performance and an increased training time complexity linear in sequence length (Pascanu et al., 2013). On the other hand, current state-of-the-art sequence models use *parallelizable* architectures (Vaswani et al., 2017; Gu et al., 2021) instead of RNNs to support fast and stable training on long sequences and capture long-range temporal dependencies.

To address this important gap in object-centric learning,

[*]Equal contribution [1]Rutgers University [2]NVIDIA Research [3]University of Southern California [4]KAIST [5]Stanford University. Correspondence to: Gautam Singh <singh.gautam@rutgers.edu>, Tong Che <tongc@nvidia.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[†]See our project page at this link.

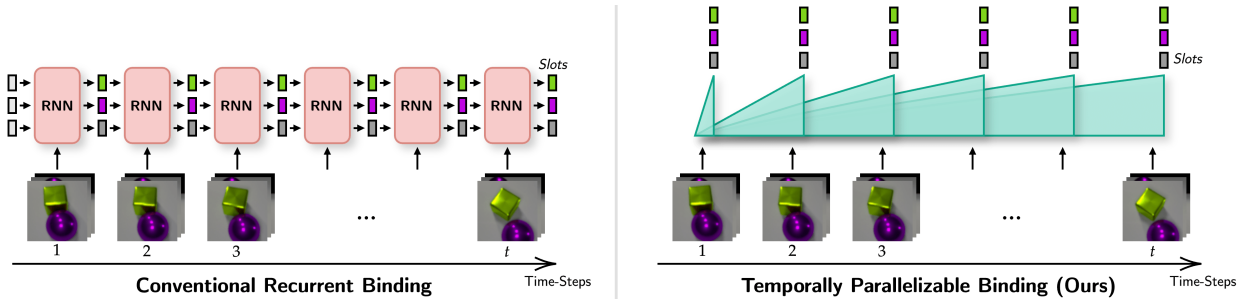


Figure 1. **Conventional Spatiotemporal Binding versus Ours.** *Left:* Conventional object-centric encoders summarize sequential sensory inputs into slots via recurrence, analogous to RNNs. *Right:* On the other hand, our proposed object-centric encoder achieves this without recurrence, allowing it to be parallelized over the sequence length, similarly to transformers.

in this work, we introduce *Parallelizable Spatiotemporal Binder* or PSB, the first parallelizable slot learning architecture for sequential inputs. PSB takes a sequence containing a set of input features for each time-step and generates a set of N-slot vectors corresponding to each time-step in a parallelizable manner. Unlike conventional object-centric learning models, which sequentially update N slots through iteration over the input sequence, our novel PSB architecture eliminates the need for such sequential iteration. Instead, it produces slots for all time-steps in parallel by refining initial slots through a fixed number of layers of our proposed *PSB block*. A PSB block leverages causal attention to allow each slot to directly see the input observations and the past slot states of the current layer. PSB is a general-purpose neural network module that can be used within any arbitrary architecture. In this work, we evaluate PSB as an encoder in various object-centric auto-encoding frameworks and demonstrate its effectiveness.

Our contributions are as follows: *i)* We introduce the first temporally parallelizable object-centric binding architecture, designed to efficiently process sequential data and alleviate the common drawbacks associated with RNNs. *ii)* We develop two novel parallelizable object-centric auto-encoder models by leveraging the proposed architecture as the encoder: one for 2D unposed videos and another for dynamic, posed multi-camera 3D scene videos, marking a key modeling advancement for these problems. *iii)* Compared to the recurrent state-of-the-art baseline, our encoder shows highly stable training on longer sequences with parallelization, resulting in 1.6× faster training speed. *iv)* Our encoder, when paired with a wide variety of decoders such as an alpha-mixture decoder, an auto-regressive transformer, NeRFs, and SlotMixer, matches or exceeds the state-of-the-art recurrent baseline's performance. Specifically, with the mixture decoder for 2D videos, we observe improvements ranging from 14.7-26.8% in FG-ARI and 2.9-7.6% in reconstruction PSNR. For dynamic 3D scenes using NeRFs, improvements range from 7.3-121% in slot linear-probing performance and 4-8% in PSNR for novel view synthesis. *v)* Via ablations, we obtain useful insights about the proposed design.

2. PSB: Parallelizable Spatiotemporal Binder

In this section, we describe our proposed architecture *Parallelizable Spatiotemporal Binder* or PSB. Our architecture aims to encode or summarize a given T-length sequence of features $\mathbf{e}_{1,1:L}, \ldots, \mathbf{e}_{T,1:L}$ (where $\mathbf{e}_{t,1:L} \in \mathbb{R}^{L \times D}$) into a T-length sequence of N slot vectors $\mathbf{s}_{1,1:N}, \ldots, \mathbf{s}_{T,1:N}$ (where $\mathbf{s}_{t,1:N} \in \mathbb{R}^{N \times D}$). For dynamic visual scenes, the N slots belonging to a specific t, i.e., $\mathbf{s}_{t,1:N}$, should capture an object-centric state of the world at time-step t. Furthermore, the n-th slots across all time-steps should consistently track the state of one specific object.

Formally, PSB works by taking initial slots $\mathbf{s}^{(0)}_{1:T,1:N}$ and transforming them conditioned on the input features $\mathbf{e}_{1:T,1:L}$ by applying M layers of our proposed PSB block:

$$\mathbf{s}^{(0)}_{1:T,1:N} \leftarrow \text{Initialize}(),$$
$$\mathbf{s}^{(i)}_{1:T,1:N} \leftarrow \text{PSBBlock}_i(\mathbf{s}^{(i-1)}_{1:T,1:N}, \mathbf{e}_{1:T,1:L}),$$

for $i = 1, \ldots, M$. The initialization of slots can be done via learned parameters or by sampling them randomly from a learned Gaussian. The output $\mathbf{s}^{(M)}_{1:T,1:N}$ of the last PSB block is considered as the slot representation for downstream use.

2.1. PSB Block

Broadly, a PSB block works by interleaving three operations: *(i)* bottom-up attention by the slots on the input features, *(ii)* self-attention among the slots, and *(iii)* an MLP.

2.1.1. Bottom-Up Attention

In this step, the slots access the bottom-up information from the inputs. For all $t = 1, \ldots, T$ in parallel, we perform:

$$\mathbf{s}_{t,1:N} \mathrel{+}= \texttt{CA}\big(\texttt{q=LN}(\mathbf{s}_{t,1:N}), \texttt{kv=LN}(\mathbf{e}_{1:T,1:L})\big),$$

where LN denotes layer normalization, CA denotes multi-headed cross-attention, argument q denotes the query and

Algorithm 1 Parallelizable Spatiotemporal Binder (PSB) Block. The algorithm receives *i)* a T-length sequence of N slot vectors denoted as $\mathbf{s}_{1:T,1:N} \in \mathbb{R}^{T\times N\times D}$, *ii)* a T-length sequence of L input feature vectors $\mathbf{e}_{1:T,1:L} \in \mathbb{R}^{T\times L\times D}$ and *iii)* an optional attention mask $\boldsymbol{\alpha} \in \{0,1\}^{T\times T}$ to enforce causal masking. The algorithm outputs the updated slots $\mathbf{s}_{1:T,1:N}$ conditioned on the input features $\mathbf{e}_{1:T,1:L}$.

Input: $\mathbf{s} \in \mathbb{R}^{T\times N\times D}$, $\mathbf{e} \in \mathbb{R}^{T\times L\times D}$, $\boldsymbol{\alpha} \in \{0,1\}^{T\times T}$
Layer params: Cross-Attention $\texttt{CA}$; Self-Attention $\texttt{SA}_1$, $\texttt{SA}_2$; LayerNorm $\texttt{LN}$; MLP $\texttt{MLP}$
  for $t = 1 \dots T$ **in parallel**
    $\mathbf{s}_{t,1:N}$ += $\texttt{CA}\big(\texttt{q=LN}\left(\mathbf{s}_{t,1:N}\right), \texttt{kv=LN}\left(\mathbf{e}_{1:T,1:L}\right), \texttt{attn\_mask=}\boldsymbol{\alpha}\big)$ # Slots attend input features.
  for $n = 1 \dots N$ **in parallel**
    $\mathbf{s}_{1:T,n}$ += $\texttt{SA}_1\big(\texttt{qkv=LN}(\mathbf{s}_{1:T,n}), \texttt{attn\_mask=}\boldsymbol{\alpha}\big)$ # Slots with the same index self-attend across time.
  for $t = 1 \dots T$ **in parallel**
    $\mathbf{s}_{t,1:N}$ += $\texttt{SA}_2\big(\texttt{qkv=LN}(\mathbf{s}_{t,1:N})\big)$ # Slots at the same time-step self-attend.
  $\mathbf{s}_{1:T,1:N}$ += $\texttt{MLP}\big(\texttt{LN}(\mathbf{s}_{1:T,1:N})\big)$ # Slots undergo an MLP.
  return $\mathbf{s}_{1:T,1:N}$

the argument $\texttt{kv}$ denotes the key and the value which are the same in this case. In implementing this cross-attention, we make three key design choices. First, we provide an option to apply *causal masking* to prevent the slots from seeing the inputs of the future time-steps. This makes our model useful as a perception module in agent-learning settings where the agent typically does not have access to future observations. Second, we employ *inverted-attention and renormalization* (Locatello et al., 2020; Wu et al., 2023a) to introduce competition among slots and to help them specialize to distinct objects. Third, to incorporate invariance to translation-in-time and to help the encoder generalize to any sequence length, we recommend using *relative positional bias* (Raffel et al., 2020) instead of absolute positional embedding (Vaswani et al., 2017) to incorporate the temporal position information in the attention process.

2.1.2. Slot Interaction

Next, the slots self-attend, allowing each slot to read the other slots to *(i)* facilitate efficient allocation of slot resources to distinct objects, *(ii)* to help the slots align across time, and *(iii)* to allow each slot to become more informative by accessing the information of other slots. We execute this self-attention via two decoupled steps:

Time-Axis Self-Attention. In conventional recurrent object-centric encoders (Jiang et al., 2019; Lin et al., 2020a; Kipf et al., 2022), each slot $\mathbf{s}_{t,n}$ is informed about its previous states $\mathbf{s}_{<t,n}$ through an independent RNN assigned per slot. This conforms to the physical principle that distinct objects evolve largely independently of each other over time, while the temporal states of the same object are highly correlated. Incorporating this principle, we perform self-attention between all slots along the time axis sharing the same index n. Specifically, for all $n = 1, \dots N$ in parallel:

$$\mathbf{s}_{1:T,n} \mathrel{+}= \texttt{SA}_1\big(\texttt{qkv=LN}(\mathbf{s}_{1:T,n})\big),$$

where $\texttt{SA}_1$ denotes multi-headed self-attention and the argument $\texttt{qkv}$ denotes the query, key, and value which are the same in this case. As before, we provide the option to apply causal masking and we recommend using relative position bias to incorporate the temporal position information.

Object-Axis Self-Attention. Next, we let the N slots of each time-step interact, i.e., for all time-steps $t = 1, \dots, T$ in parallel:

$$\mathbf{s}_{t,1:N} \mathrel{+}= \texttt{SA}_2\big(\texttt{qkv=LN}(\mathbf{s}_{t,1:N})\big),$$

where $\texttt{SA}_2$ denotes multi-headed self-attention and the argument $\texttt{qkv}$ denotes the query, key, and value which are the same in the case of self-attention.

2.1.3. Processing Gathered Information via MLP

To process the information gathered via the bottom-up attention and slot interaction steps, the slots are fed to an $\texttt{MLP}$ for all $t = 1, \dots, T$ and $n = 1, \dots N$ in parallel:

$$\mathbf{s}_{t,n} \mathrel{+}= \texttt{MLP}\big(\texttt{LN}(\mathbf{s}_{t,n})\big),$$

The resulting slots are then passed to the downstream layers. The operation of a PSB block is also summarized in Algorithm 1. Note that all operations described above are performed through residual connections, making the PSB block suitable for deep stacking (He et al., 2016).

3. Object-Centric Learning via Parallelizable Spatiotemporal Binder

In this section, we outline two application scenarios of our proposed encoder.

3.1. 2D Unposed Videos

In this setting, we perform unsupervised object-centric representation learning from 2D unposed videos. For this, we adopt the video auto-encoding framework of (Kipf et al., 2022; Singh et al., 2022). In particular, a video contains T frames $\mathbf{x}_1, \ldots, \mathbf{x}_T$ where each frame is an RGB image $\mathbf{x}_t \in \mathbb{R}^{C \times H \times W}$. Our goal is to encode it into slot representations $\mathbf{s}_{1,1:N}, \ldots, \mathbf{s}_{T,1:N}$, where $\mathbf{s}_{t,1:N} \in \mathbb{R}^{N \times D}$ denotes a collection of N slot vectors for the t-th time-step.

To achieve this, we first encode each frame $\mathbf{x}_t$ via a CNN and flatten the resulting feature map, producing L feature vectors per frame: $\mathbf{e}_t = \text{CNN}_\phi(\mathbf{x}_t) \in \mathbb{R}^{L \times D}$. On these features, we apply our proposed PSB encoder to obtain the slots: $\mathbf{s}_{1,1:N}, \ldots, \mathbf{s}_{T,1:N} = \text{PSB}_\phi(\mathbf{e}_1, \ldots, \mathbf{e}_T)$. This slot inference process is trained in a self-supervised manner by decoding the slots and trying to reconstruct the original video frames. We consider two decoder choices. For visually simple datasets, we test an alpha-mixture decoder which is trained with MSE reconstruction loss $\mathcal{L}(\phi, \theta) = ||\mathbf{x}_t - \text{Decoder}_\theta(\mathbf{s}_{t,1:N})||^2$ (Locatello et al., 2020). For visually complex datasets, we test the auto-regressive image-transformer decoder (Singh et al., 2021; 2022) which is trained with cross-entropy loss to reconstruct a DVAE representation of the video frames. (See Appendix D.3.2)

3.2. 3D Posed Multi-Camera Videos

In this setting, we perform unsupervised object-centric representation learning on dynamic 3D scenes. In this setup, we aim to learn slot representations for a given T-length sequence of posed multi-camera observations denoted by $\mathbf{X}_1, \ldots, \mathbf{X}_T$. Here, each $\mathbf{X}_t$ consists of K distinct observations or *views* corresponding to the K cameras in the scene i.e., $\mathbf{X}_t = \mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,K}$. Each view $\mathbf{x}_{t,k}$ belongs to $\mathbb{R}^{C \times H \times W}$ where H and W are the image height and width, respectively and C is the number of channels. Since these are posed observations, $C = 9$ which includes 3 channels for RGB pixel color, 3 channels for the camera ray origin, and 3 for camera ray direction.

Novel View Prediction. We adopt a novel view prediction framework to train the model where a certain fraction of the input views are held out and the remaining are passed to the encoder to infer the slots. The slots are then decoded via a viewpoint-conditioned decoder that tries to predict all the available views—both the held-out and the shown ones. This is a common practice in 3D scene representation learning frameworks (Kumar et al., 2018; Singh et al., 2019; Sajjadi et al., 2022).

Set Latent Scene Representation (SLSR). To encode the views, we adopt the backbone of Sajjadi et al. (2022). For each time-step, we feed the K' (out of K) visible views to a CNN to obtain a feature map. We flatten the feature maps of each view, stack them together for all K' views, and feed them to a transformer. The transformer's output is known as Set Latent Scene Representation or SLSR:

$$\tilde{\mathbf{e}}_{t,k} = \text{CNN}_\phi(\mathbf{x}_{t,k}) \implies \mathbf{e}_t = \text{Transformer}_\phi(\tilde{\mathbf{e}}_{t,1:K'}),$$

where $\mathbf{e}_t \in \mathbb{R}^{L \times D}$ is the SLSR for the time-step t and is a collection of L feature vectors.

Slot Learning using PSB. Next, we provide the SLSRs $\mathbf{e}_1, \ldots, \mathbf{e}_T$ to our proposed encoder PSB and obtain slots: $\mathbf{s}_{1,1:N}, \ldots, \mathbf{s}_{T,1:N} = \text{PSB}_\phi(\mathbf{e}_1, \ldots, \mathbf{e}_T)$. We then decode the slots to reconstruct the available pixels of both the novel and the visible views. We employ viewpoint-aware decoders that take a ray (origin and direction) as input and predict the pixel color conditioned on the slot representation. In particular, to render a ray $\mathbf{r}$ of a time-step t, we can describe the decoding process as: $\hat{\mathbf{c}} = \text{Decoder}_\theta(\mathbf{r}, \mathbf{s}_{t,1:N})$, where $\mathbf{c} \in \mathbb{R}^3$. To train the complete model, we minimize the MSE loss of the predicted pixel against the true color of the pixel i.e., $\mathcal{L}(\phi, \theta) = ||\hat{\mathbf{c}} - \mathbf{c}||^2$.

Decoder Options. We consider two 3D decoders: NeRF (Mildenhall et al., 2020) and SlotMixer (Sajjadi et al., 2022). For NeRF, we maintain an MLP g_θ^{slot} which, for a given 3D coordinate $\mathbf{o}$ and a viewing direction $\mathbf{d}$, returns a color value and a density value conditioned on a slot representation: $\mathbf{c}_n, \sigma_n = g_\theta^{\text{slot}}(\mathbf{o}, \mathbf{d}, \mathbf{s}_n)$. Combining the outputs for N slots $\mathbf{s}_1, \ldots, \mathbf{s}_N$, we obtain the combined density $\sigma = \sum_{n=1}^N \sigma_n$ and color $\mathbf{c} = \sum_{n=1}^N \mathbf{c}_n \sigma_n / \sigma$. To obtain the color of an image pixel, we shoot the corresponding ray from the camera, sample $N_{\text{bins}}+1$ points along the ray, and integrate the colors along the ray as: $\sum_{i=1}^{N_{\text{bins}}} T_i \alpha_i \mathbf{c}_i$, where $T_i = \prod_{j=1}^{i-1}(1 - \alpha_i)$ is the transmittance, and $\alpha_i = 1 - \exp(-\sigma_i ||\mathbf{o}_{i+1} - \mathbf{o}_i||_2)$ is the opacity. To capture complex static topography, we investigate incorporating a static field and a sky field following Yang et al. (2023a), decoupling the static field modeling from the dynamic field modeling. For SlotMixer, we use its default implementation (Sajjadi et al., 2022). For a detailed description of the decoders, see Appendix D.4.3 and D.4.4.

4. Related Work

A large body of work has emerged on the topic of learning compositional and object-centric scene representations (Chen et al., 2016; Higgins et al., 2017; Burgess et al., 2019; Greff et al., 2019; Locatello et al., 2020; Greff et al., 2017; Engelcke et al., 2019; Engelcke et al., 2021; Anciukevicius et al., 2020; von Kügelgen et al., 2020; Greff et al., 2020; Singh et al., 2021; Chang et al., 2022; Zhang et al., 2022; Löwe et al., 2022; 2023; Stanić et al., 2023; Jiang et al., 2023; Wu et al., 2023b; Jia et al., 2023). Most architectures for videos start as static scene models that are extended by applying the same model recurrently on the video frames, thus making them non-parallelizable unlike ours (Kosiorek et al., 2018; Crawford & Pineau, 2019; Lin et al., 2020a;

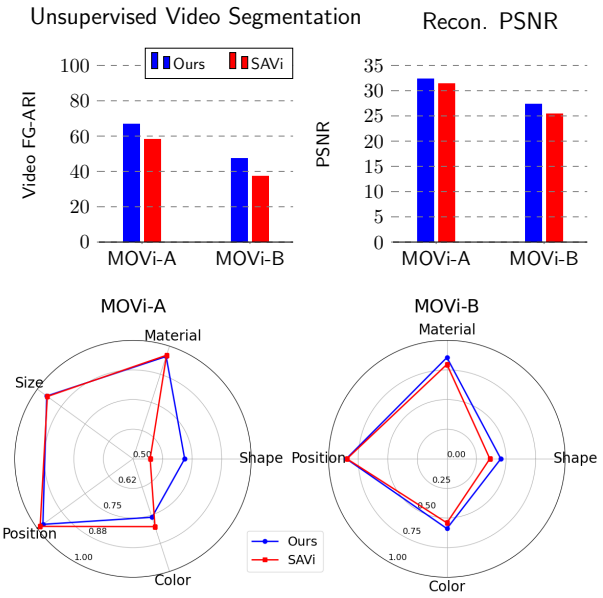


Figure 2. **Unsupervised Object-Centric Learning on MOVi-A and MOVi-B using Spatial Broadcast Decoder.** We compare our proposed encoder with the recurrence-based baseline encoder SAVi (Kipf et al., 2022). *Top-Left:* Video-level FG-ARI score (↑). *Top-Right:* Reconstruction PSNR (↑). *Bottom:* Slot linear probing performances (↑). Reported are the R^2 score for continuous-valued object factors (position and color) and classification accuracy for categorical object factors (shape, size, and material). We observe that our encoder surpasses the recurrent baseline SAVi in terms of FG-ARI and PSNR, and does markedly better in linear-probing performance for complex factors such as the object shape.

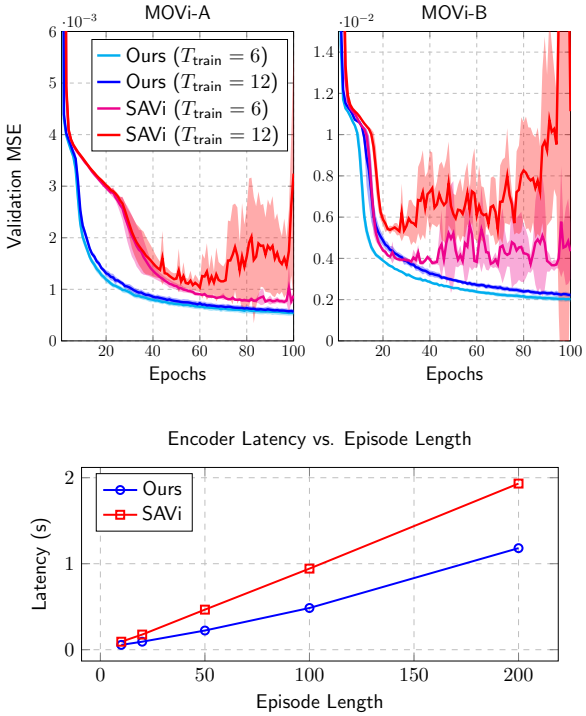


Figure 3. **Computational Drawbacks of RNN-based Object-Centric Learning.** We compare our proposed encoder with the recurrent baseline SAVi. *Top:* We show validation loss curves (mean and standard deviation computed over 5 seeds) for training runs on MOVi-A and MOVi-B. T_{train} denotes the length of each training episode. We note that as we increase the episode length from 6 to 12, SAVi becomes highly unstable while our model continues to train smoothly. *Bottom:* We report the time taken (in seconds) to perform one training step plotted as a function of the episode length. We observe a speed-up of about 1.6× over SAVi.

Kipf et al., 2022; Greff et al., 2019; Veerapaneni et al., 2019; Elsayed et al., 2022; Singh et al., 2022; Zadaianchuk et al., 2023). This is also true for 3D-aware object-centric learning where static scene models (Chen et al., 2020; Li et al., 2020; Yu et al., 2021; Stelzner et al., 2021; Castrejon et al., 2022; Jabri et al., 2023; Jia et al., 2023) are sometimes extended to handle dynamic 3D scenes via recurrence (Crawford & Pineau, 2020; Li et al., 2021). While exceptions to this exist (Kabra et al., 2021; Henderson & Lampert, 2020; Gopalakrishnan et al., 2022), however, these learn global slots for the entire episode instead of per time-step slots in a parallelizable manner like ours. Orthogonal to object-centric learning, efforts to resolve concerns of scalability and parallelizability of RNNs have a long history (Oord et al., 2016; Van den Oord et al., 2016; Chang et al., 2017; Vaswani et al., 2017; Dauphin et al., 2017; Li et al., 2018; Gu et al., 2021; Trinh et al., 2018). However, ours is the first work that tackles the question of bringing such scalability and parallelization to object-centric learning. For an extended discussion of the related work, see Appendix A.

5. Experiments

In experiments, we demonstrate the advantages of our proposed parallelizable encoder compared to the conventional recurrent slot learning approach. We evaluate performance in two object-centric learning settings, specifically: *i)* learning from unposed 2D videos, and *ii)* learning from posed multi-camera videos of dynamic 3D scenes.

5.1. Learning from 2D Unposed Videos

5.1.1. Setup

Datasets. In this setting, we evaluate on the MOVi benchmark (Greff et al., 2022) comprising five datasets: MOVi-A, MOVi-B, MOVi-C, MOVi-D, and MOVi-E. We only use the RGB frames without any auxiliary inputs or supervision.

Baselines. We implement our model following the description in Section 3.1 and evaluate our proposed encoder paired with two decoders: an alpha-mixture decoder (see Appendix

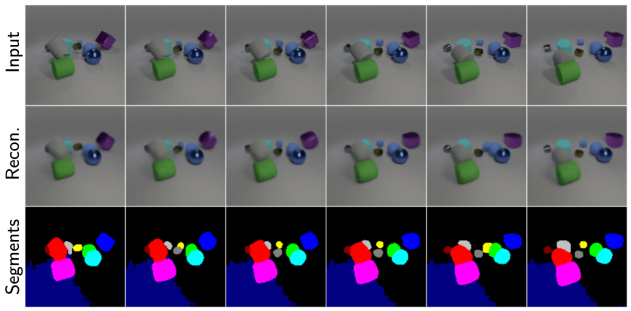


Figure 4. **Object-Centric Learning on MOVi-A using Our Proposed Encoder.** We visualize a given video and its reconstruction and decomposition into objects using the proposed model. We note that object identity is consistently maintained over time as evidenced by the segment colors across frames.

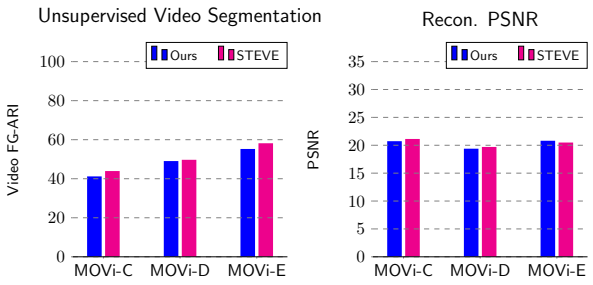


Figure 5. **Unsupervised Video Segmentation on MOVi-C, D and E using Autoregressive Image-Transformer Decoder.** We compare ours with STEVE which is based on the recurrent encoder of Kipf et al. (2022). *Left:* Video-level FG-ARI score (↑). *Right:* Reconstruction PSNR (↑).

D.3.1) for visually simple MOVi-A and B datasets and an autoregressive transformer decoder (see Appendix D.3.2) for visually complex MOVi-C, D and E datasets. For both decoder choices, we compare the performance by substituting our proposed encoder with the current state-of-the-art and one of the most popular neural network architectures for learning sequential slot representations from videos (Kipf et al., 2022; Elsayed et al., 2022). This amounts to a comparison with unsupervised SAVi (Kipf et al., 2022) when using an alpha-mixture decoder and STEVE (Singh et al., 2022) when using an autoregressive transformer decoder.

Metrics. To measure the performance, we report the *video-level FG-ARI* score, slot linear probing performance and the *reconstruction PSNR*. The video-level FG-ARI score is a common metric (Kipf et al., 2022; Elsayed et al., 2022; Kabra et al., 2021) to measure whether a video is accurately decomposed into slot-based representation and whether the decomposition is consistent across the time-steps. We take episode length to be 6. The linear probing performance for evaluating the representational informativeness of slots is measured in terms of R^2 score for continuous object factors and classification accuracy for the discrete object factors following Dang-Nhu (2021) (see Appendix E.1). The reconstruction PSNR measures how well the input frames can be reconstructed from the representation. For a fair comparison, we use an identical spatial-broadcast decoder for all baselines and datasets. For MOVi-C, D, and E, where the models are trained using an autoregressive transformer decoder, we freeze the slots and train a slot-conditioned spatial-broadcast decoder for evaluating FG-ARI and PSNR.

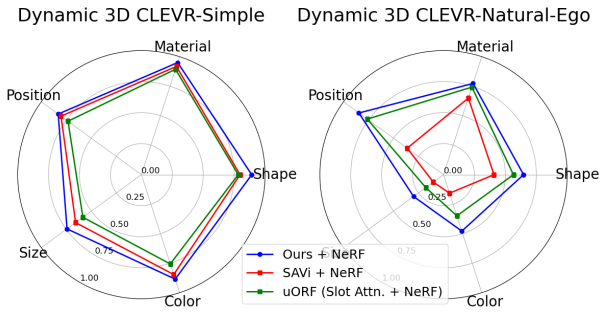
5.1.2. Results

Unsupervised Scene Decomposition and Representation. In terms of scene decomposition performance, in Fig. 2, we note that our proposed encoder demonstrates a significantly superior FG-ARI score compared to the recurrent encoder baseline SAVi. In terms of representation quality, in Fig. 2, we note that our encoder achieves a significantly better PSNR compared to the baseline. Furthermore, ours shows better overall linear-probing performance over the object factors, with a particularly marked improvement for the shape factor. The shape factor—being more complex than other factors like color—indicates that our encoder is capable of learning more expressive object representations than the recurrent baseline. Importantly, this improved expressiveness does not compromise the object-centric decomposition, as evidenced by our superior FG-ARI scores.

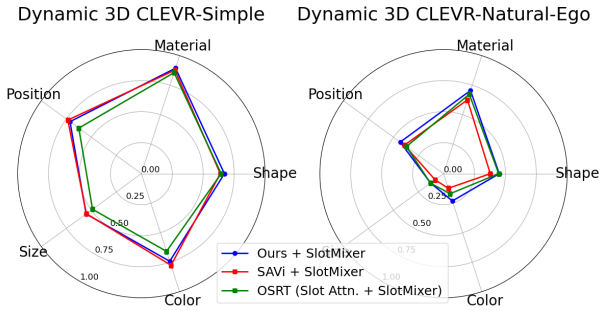
Training Stability on Long Episodes. Fig. 3 demonstrates the training stability of our encoder compared to the recurrent SAVi model. Owing to our encoder's ability to directly attend to any previous time-step without relying on recurrence, it exhibits high stability and trains smoothly on longer episodes. In contrast, the SAVi model displays considerable instability due to its recurrence mechanism, which is susceptible to gradient explosion.

Training Speed on Long Episodes. Fig. 3 also compares the duration of a single training step as a function of the episode length. Thanks to our parallelizable implementation which eliminates the need to iterate explicitly over the entire sequence length, our encoder demonstrates superior speed on very long sequences.

Compatibility with Expressive Decoders. Expressive decoders, e.g., autoregressive image transformers, have been advantageous for object-centric learning in visually complex scenes (Singh et al., 2021; 2022; Jiang et al., 2023; Wu et al., 2023b). Since these decoders have been typically paired with the Slot Attention family of encoders, it becomes a question whether our proposed encoder also maintains compatibility with such powerful decoders or not. In Fig. 5, we evaluate our encoder on visually complex scenes using the autoregressive transformer decoder. We find that while our

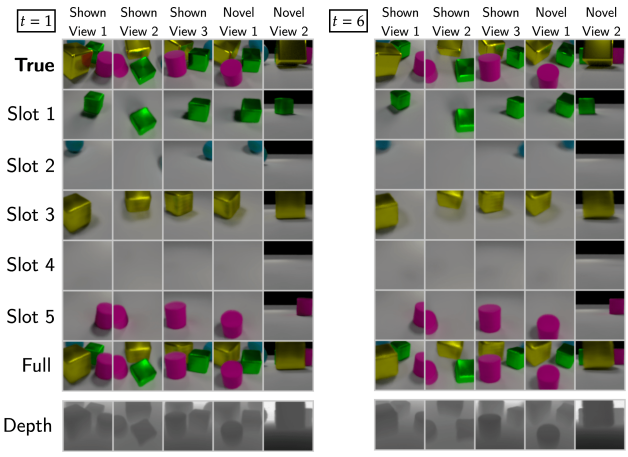


(a) NeRF Decoder

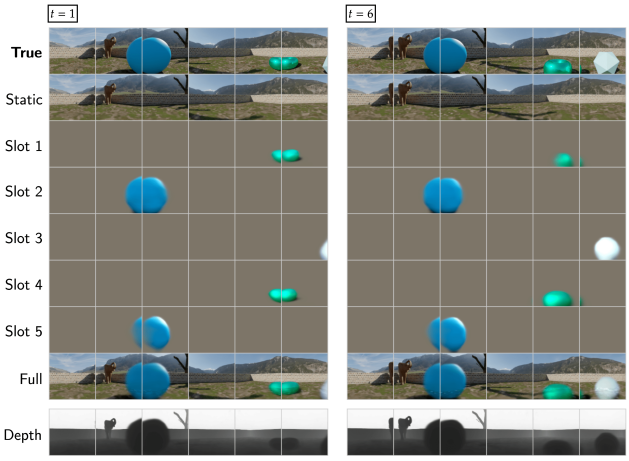


(b) SlotMixer Decoder

Figure 6. **Comparison of Encoders in Dynamic 3D Scenes.** We compare the encoder performances trained with two decoder options: NeRF (top) and SlotMixer (bottom). Reported are the R^2 score (↑) for continuous-valued object factors (position, size, and color) and classification accuracy (↑) for categorical object factors (shape and material). With both decoders, our encoder surpasses SAVi as well as the static 3D scene models, with a noticeably large margin in the case of NeRF decoder.



(a) Dynamic 3D CLEVR-Simple



(b) Dynamic 3D CLEVR-Natural-Ego

Figure 7. **Unsupervised Dynamic 3D Scene Understanding.** We visualize the RGB rendering of the individual slots and the RGB and depth rendering from all slots taken together for time-steps: $t = 1$ and $t = 6$. The model was trained as described in Section 3.2 with a NeRF decoder. We note the 3D scene decomposition, consistent alignment of slots across time-steps, and unsupervised depth inference. See the GIF version in the supplementary.

segmentation performance is slightly worse, the difference is not substantial. Therefore, in scenarios where training efficiency and stability are prioritized, our proposed encoder with powerful decoders remains a preferred option.

5.2. Learning from 3D Posed Multi-Camera Videos

5.2.1. Setup

Datasets. In this setting, we evaluate on two datasets: *Dynamic 3D CLEVR-Simple* and *Dynamic 3D CLEVR-Natural-Ego*. We synthesize these datasets as extensions of the CLEVR dataset to incorporate physical dynamics, multiple cameras, 3D camera pose information, moving ego observer, and visual complexity. We visualize these datasets in Fig. 7 and 15. For more details, see Appendix C.

Baselines. We implement our model following the description in Section 3.2 and evaluate the proposed encoder paired with two decoders: NeRF (Mildenhall et al., 2020; Yu et al., 2021; Stelzner et al., 2021) and SlotMixer (Sajjadi et al., 2022). For both decoder choices, we compare the perfor-

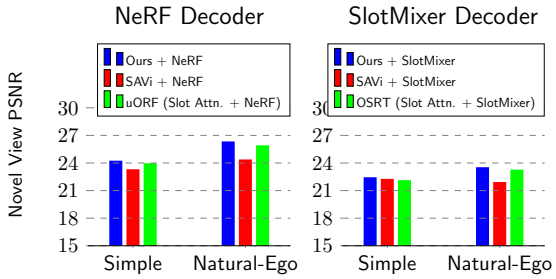


Figure 8. **Novel View Synthesis in Dynamic 3D Scenes.** We compare the PSNR (↑) of generated views on novel viewpoints. We note improved performance with a large margin relative to SAVi and by a smaller margin relative to static 3D scene models.

mance by substituting our proposed encoder with: *(i)* SAVi encoder, the current state-of-the-art for learning sequential slot representations from sequential inputs based on recurrence; and *(ii)* Slot Attention applied independently to each time-step. This use of Slot Attention corresponds to the uORF framework of Yu et al. (2021) when using a NeRF decoder and to the OSRT framework of Sajjadi et al. (2022) when using a SlotMixer decoder, the two state-of-the-art models for 3D-aware slot learning from static 3D scene observations. Since the focus of this study is on encoder architecture, for fair comparison, we use an identical SLSR backbone and identical decoder architectures across all compared models while substituting only the encoder that produces the slots.

Metrics. To measure performance, we focus on three main aspects of interest: *i)* representation quality, *ii)* novel view synthesis and *iii)* unsupervised segmentation across time and cameras. To measure representation quality, we perform slot linear probing and report the classification accuracy for the categorical object factors and the R^2-score for the continuous-valued object factors following Dang-Nhu (2021) (see Appendix E.1). To measure the performance on novel view synthesis, we report the PSNR on novel views that were not shown to the encoder. For segmentation, we report FG-ARI computed in 4 ways to measure representational consistency: *(i)* per camera per time-step, *(ii)* per camera across time-steps, *(iii)* across cameras per time-step, and *(iv)* across cameras and time-steps.

5.2.2. Results

3D-Aware Object-Centric Scene Representation and Decomposition. Fig. 6a and 6b show our evaluation of linear-probing performance on slots to predict object factors. With both NeRF and SlotMixer decoder, our proposed encoder surpasses the SAVi baseline as well as the static 3D scene models, with a greater gap when the NeRF decoder is used. A drawback of applying static scene models e.g., uORF and OSRT, on dynamic scenes is that they can suffer in producing aligned slot representations across time, evidenced by the FG-ARI score (reported in Fig. 9 in Appendix) computed for measuring temporal consistency. Another noteworthy point is that while temporal models like SAVi should theoretically benefit from the motion information that static scene models cannot access, SAVi's recurrent implementation and the resulting training instability prevent it from leveraging temporal information effectively—leading to its worse performance than even the static models.

Novel View Synthesis. In Fig. 8, we report the PSNR of generated views on unseen camera viewpoints i.e., viewpoints that were not given to the encoder to infer the slot representation. This metric measures the model's understanding of the underlying scene geometry as well as the slot representation quality. We observe that our model consistently surpasses both SAVi and per-timestep Slot Attention in performance.

5.3. Ablation Study

Learned vs. Random Slot Initialization. In Fig. 10 and 11 in the appendix, we evaluate the impact of initializing slots by randomly sampling them from a learned Gaussian. In this variant, we note a generally worse performance compared to initializing slots as learned parameters.

Decoupled vs. Joint Slot Interaction. In Fig. 10 and 11 in the appendix, we also analyze the effect of decoupling the slot-slot self-attention along the time and object axes versus letting all NT slots interact via a single self-attention step. In terms of performance alone, we do not notice a clear advantage of either version. However, it is also important to acknowledge the memory complexity: the joint interaction version requires $\mathcal{O}(N^2T^2)$ memory which can be costlier and hurt scalability compared to using the decoupled version which requires a lower $\mathcal{O}(NT^2) + \mathcal{O}(N^2T)$ memory.

No Inverted Attention and Renormalization. In Fig. 10 in the appendix, we evaluate the impact of using standard dot-product attention instead of inverted attention and renormalization that introduces competition among slots in the bottom-up attention step (Locatello et al., 2020; Wu et al., 2023a). Without inverted attention, we find that the video decomposition performance as measured by FG-ARI becomes worse, suggesting that inverted attention is a beneficial inductive bias to keep.

6. Discussion

In conclusion, this work introduces a novel temporally-parallelizable object-centric binding architecture for efficiently processing sequential data to learn slot representations, overcoming the drawbacks of conventional RNN-based architectures. We present two novel auto-encoder models utilizing this architecture for unsupervised object-centric learning, tailored for 2D unposed videos and dynamic 3D scene videos. Our comprehensive evaluations across various decoders demonstrate the proposed architecture's superior performance and computational efficiency.

Limitations and Avenues. We note the limitations of the current work and avenues to address them via future work. First, while our proposed design replaces RNNs with attention, thus providing the benefit of speed and parallelization, it also incurs a quadratic memory complexity in terms of sequence length. To address this, future explorations can consider the use of recent advances e.g., parallelizable SSMs (Gu et al., 2021), based on our framework. However, we should consider the fact that many of today's best-performing architectures are based on Transformers which also incur greater computation costs than their RNN-based

predecessors. However, Transformers are still widely used because their benefits outweigh their costs. We hope to see a similar outcome via our proposed encoder where the benefits of scalability should outweigh the costs in the long run. Second, it is of interest to apply the proposed architecture to longer episodes and real scenes to utilize the full potential of our scalable approach. Third, the proposed framework can be used to build object-centric dynamics models that can find applications in various agent learning scenarios, e.g., planning for autonomous vehicles, robotics, and RL.

Impact Statement

This paper presents a representation learning method for images and videos. While the proposed model uses generation as an objective function, the goal of the proposed model is not on generation but the quality of representation. Therefore, the negative impact of the proposed model on potential fake image generation is little. As the proposed method is very general representation method, it can be used in any application that might include the use of any intention.

Acknowledgements

Sungjin Ahn is supported by Brain Pool Plus Program (No. 2021H1D3A2A03103645) through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT.

References

Anciukevicius, T., Lampert, C. H., and Henderson, P. Object-centric image generation with factored depths, locations, and appearances. *arXiv preprint arXiv:2004.00642*, 2020.

Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., and Schmid, C. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6836–6846, 2021.

Bertasius, G., Wang, H., and Torresani, L. Is space-time attention all you need for video understanding? In *ICML*, volume 2, pp. 4, 2021.

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

Castrejon, L., Ballas, N., and Courville, A. Inferno: Inferring object-centric 3d scene representations without supervision. In *ICLR2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality*, 2022.

Chang, M., Griffiths, T. L., and Levine, S. Object representations as fixed points: Training iterative refinement algorithms with implicit differentiation. *arXiv preprint arXiv:2207.00787*, 2022.

Chang, S., Zhang, Y., Han, W., Yu, M., Guo, X., Tan, W., Cui, X., Witbrock, M., Hasegawa-Johnson, M. A., and Huang, T. S. Dilated recurrent neural networks. *Advances in neural information processing systems*, 30, 2017.

Chen, C., Deng, F., and Ahn, S. Roots: Object-centric representation and rendering of 3d scenes. *J. Mach. Learn. Res.*, 22:259:1–259:36, 2020.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pp. 2172–2180, 2016.

Crawford, E. and Pineau, J. Exploiting spatial invariance for scalable unsupervised object tracking. *arXiv preprint arXiv:1911.09033*, 2019.

Crawford, E. and Pineau, J. Learning 3d object-oriented world models from unlabeled videos. In *Workshop on Object-Oriented Learning at ICML*, 2020.

Dang-Nhu, R. Evaluating disentanglement of structured representations. *arXiv preprint arXiv:2101.04041*, 2021.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. Language modeling with gated convolutional networks. In *International conference on machine learning*, pp. 933–941. PMLR, 2017.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Du, Y., Smith, K. A., Ulman, T., Tenenbaum, J. B., and Wu, J. Unsupervised discovery of 3d physical objects from video. *ArXiv*, 2021.

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*, 2022.

Engelcke, M., Kosiorek, A. R., Jones, O. P., and Posner, I. Genesis: Generative scene inference and sampling with object-centric latent representations, 2019.

Engelcke, M., Jones, O. P., and Posner, I. Genesis-v2: Inferring unordered object representations without iterative refinement. *arXiv preprint arXiv:2104.09958*, 2021.

Feichtenhofer, C., Fan, H., Li, Y., and He, K. Masked autoencoders as spatiotemporal learners. *ArXiv*, abs/2205.09113, 2022.

Feng, L., Hajimirsadeghi, H., Bengio, Y., and Ahmed, M. O. Latent bottlenecked attentive neural processes. *arXiv preprint arXiv:2211.08458*, 2022.

Fodor, J. A. and Pylyshyn, Z. W. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2): 3–71, 1988.

Gopalakrishnan, A., Irie, K., Schmidhuber, J., and van Steenkiste, S. Unsupervised learning of temporal abstractions with slot-based transformers. *arXiv preprint arXiv:2203.13573*, 2022.

Greff, K., van Steenkiste, S., and Schmidhuber, J. Neural expectation maximization. In *Advances in Neural Information Processing Systems*, pp. 6691–6701, 2017.

Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., and Lerchner, A. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pp. 2424–2433. PMLR, 2019.

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

Greff, K., Belletti, F., Beyer, L., Doersch, C., Du, Y., Duckworth, D., Fleet, D. J., Gnanapragasam, D., Golemo, F., Herrmann, C., Kipf, T., Kundu, A., Lagun, D., Laradji, I., Liu, H.-T. D., Meyer, H., Miao, Y., Nowrouzezahrai, D., Oztireli, C., Pot, E., Radwan, N., Rebain, D., Sabour, S., Sajjadi, M. S. M., Sela, M., Sitzmann, V., Stone, A., Sun, D., Vora, S., Wang, Z., Wu, T., Yi, K. M., Zhong, F., and Tagliasacchi, A. Kubric: a scalable dataset generator. 2022.

Gu, A., Goel, K., and R'e, C. Efficiently modeling long sequences with structured state spaces. *ArXiv*, abs/2111.00396, 2021.

Guo, J., Deng, N., Li, X., Bai, Y., Shi, B., Wang, C., Ding, C., Wang, D., and Li, Y. Streetsurf: Extending multi-view implicit surface reconstruction to street views. *ArXiv*, abs/2306.04988, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Henderson, P. and Lampert, C. H. Unsupervised object-centric video generation and decomposition in 3d. *Advances in Neural Information Processing Systems*, 33: 3106–3117, 2020.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv: Learning*, 2016.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M. M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

Ho, J., Kalchbrenner, N., Weissenborn, D., and Salimans, T. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

Jabri, A., van Steenkiste, S., Hoogeboom, E., Sajjadi, M. S. M., and Kipf, T. Dorsal: Diffusion for object-centric representations of scenes $\{et al.\}$. 2023.

Jaegle, A., Gimeno, F., Brock, A., Vinyals, O., Zisserman, A., and Carreira, J. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pp. 4651–4664. PMLR, 2021.

Jia, B., Liu, Y., and Huang, S. Improving object-centric learning with query optimization. In *The Eleventh International Conference on Learning Representations*, 2023.

Jiang, J., Janghorbani, S., De Melo, G., and Ahn, S. Scalor: Generative world models with scalable object representations. In *International Conference on Learning Representations*, 2019.

Jiang, J., Deng, F., Singh, G., and Ahn, S. Object-centric slot diffusion. *NeurIPS*, 2023.

Kabra, R., Zoran, D., Erdogan, G., Matthey, L., Creswell, A., Botvinick, M., Lerchner, A., and Burgess, C. P. Simone: View-invariant, temporally-abstracted object representations via unsupervised video decomposition. *arXiv preprint arXiv:2106.03849*, 2021.

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.

Kosiorek, A. R., Kim, H., Posner, I., and Teh, Y. W. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Neural Information Processing Systems*, 2018.

Kosiorek, A. R., Strathmann, H., Zoran, D., Moreno, P., Schneider, R., Mokrá, S., and Rezende, D. J. Nerf-vae: A geometry aware 3d scene generative model. In *International Conference on Machine Learning*, pp. 5742–5752. PMLR, 2021.

Kumar, A., Eslami, S., Rezende, D. J., Garnelo, M., Viola, F., Lockhart, E., and Shanahan, M. Consistent generative query networks. *arXiv preprint arXiv:1807.02033*, 2018.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017.

Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., and Teh, Y. W. Set transformer. 2018.

Li, N., Eastwood, C., and Fisher, R. Learning object-centric representations of multi-object scenes from multiple views. *Advances in Neural Information Processing Systems*, 33:5656–5666, 2020.

Li, N., Raza, M. A., Hu, W., Sun, Z., and Fisher, R. Object-centric representation learning with generative spatial-temporal factorization. *Advances in Neural Information Processing Systems*, 34:10772–10783, 2021.

Li, S., Li, W., Cook, C., Zhu, C., and Gao, Y. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5457–5466, 2018.

Lin, Z., Wu, Y.-F., Peri, S., Fu, B., Jiang, J., and Ahn, S. Improving generative imagination in object-centric world models. In *International Conference on Machine Learning*, pp. 6140–6149. PMLR, 2020a.

Lin, Z., Wu, Y.-F., Peri, S. V., Sun, W., Singh, G., Deng, F., Jiang, J., and Ahn, S. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. *ArXiv*, abs/2001.02407, 2020b.

Liu, Z., Ning, J., Cao, Y., Wei, Y., Zhang, Z., Lin, S., and Hu, H. Video swin transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3202–3211, 2022.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Löwe, S., Lippe, P., Rudolph, M., and Welling, M. Complex-valued autoencoders for object discovery. *arXiv preprint arXiv:2204.02075*, 2022.

Löwe, S., Lippe, P., Locatello, F., and Welling, M. Rotating features for object discovery. *arXiv preprint arXiv:2306.00600*, 2023.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, 65:99–106, 2020.

Nguyen-Phuoc, T. H., Richardt, C., Mai, L., Yang, Y., and Mitra, N. Blockgan: Learning 3d object-aware scene representations from unlabelled images. *Advances in neural information processing systems*, 33:6767–6778, 2020.

Niemeyer, M. and Geiger, A. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11453–11464, 2021.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318. Pmlr, 2013.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Sajjadi, M. S., Duckworth, D., Mahendran, A., van Steenkiste, S., Pavetic, F., Lucic, M., Guibas, L. J., Greff, K., and Kipf, T. Object scene representation transformer. *Advances in Neural Information Processing Systems*, 35: 9512–9524, 2022.

Sajjadi, M. S., Mahendran, A., Kipf, T., Pot, E., Duckworth, D., Lučić, M., and Greff, K. Rust: Latent neural scene representations from unposed imagery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17297–17306, 2023.

Sajjadi, M. S. M., Meyer, H., Pot, E., Bergmann, U. M., Greff, K., Radwan, N., Vora, S., Lucic, M., Duckworth, D., Dosovitskiy, A., Uszkoreit, J., Funkhouser, T. A., and Tagliasacchi, A. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6219–6228, 2021.

Seitzer, M., van Steenkiste, S., Kipf, T., Greff, K., and Sajjadi, M. S. Dyst: Towards dynamic neural scene representations on real-world videos. *arXiv preprint arXiv:2310.06020*, 2023.

Sharma, P., Tewari, A., Du, Y., Zakharov, S., Ambrus, R., Gaidon, A., Freeman, W. T., Durand, F., Tenenbaum, J. B., and Sitzmann, V. Seeing 3d objects in a single image via self-supervised static-dynamic disentanglement. *arXiv preprint arXiv:2207.11232*, 2022.

Singh, G., Yoon, J., Son, Y., and Ahn, S. Sequential neural processes. *Advances in Neural Information Processing Systems*, 32, 2019.

Singh, G., Deng, F., and Ahn, S. Illiterate dall-e learns to compose. In *International Conference on Learning Representations*, 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*, 35:18181–18196, 2022.

Spelke, E. S. and Kinzler, K. D. Core knowledge. *Developmental science*, 10(1):89–96, 2007.

Stanić, A., Gopalakrishnan, A., Irie, K., and Schmidhuber, J. Contrastive training of complex-valued autoencoders for object discovery. *arXiv preprint arXiv:2305.15001*, 2023.

Stelzner, K., Kersting, K., and Kosiorek, A. R. Decomposing 3d scenes into objects via unsupervised volume segmentation. *ArXiv*, abs/2104.01148, 2021.

Trinh, T., Dai, A., Luong, T., and Le, Q. Learning longer-term dependencies in rnns with auxiliary losses. In *International Conference on Machine Learning*, pp. 4965–4974. PMLR, 2018.

Turki, H., Zhang, J. Y., Ferroni, F., and Ramanan, D. Suds: Scalable urban dynamic scenes. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12375–12385, 2023.

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

Van Steenkiste, S., Kurach, K., Schmidhuber, J., and Gelly, S. Investigating object compositionality in generative adversarial networks. *Neural Networks*, 130:309–325, 2020.

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Neural Information Processing Systems*, 2017.

Veerapaneni, R., Co-Reyes, J. D., Chang, M., Janner, M., Finn, C., Wu, J., Tenenbaum, J. B., and Levine, S. Entity abstraction in visual model-based reinforcement learning. *arXiv preprint arXiv:1910.12827*, 2019.

von Kügelgen, J., Ustyuzhaninov, I., Gehler, P., Bethge, M., and Schölkopf, B. Towards causal generative scene models via competition of experts. *arXiv preprint arXiv:2004.12906*, 2020.

Wu, T., Zhong, F., Tagliasacchi, A., Cole, F., and Oztireli, C. D2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. arxiv preprint [2022-07-07], 2022.

Wu, Y.-F., Greff, K., Deepmind, G., Elsayed, G. F., Mozer, M. C., Kipf, T., and van Steenkiste, S. Inverted-attention transformers can learn object representations: Insights from slot attention. 2023a.

Wu, Z., Hu, J., Lu, W., Gilitschenski, I., and Garg, A. Slotdiffusion: Object-centric generative modeling with diffusion models. *NeurIPS*, 2023b.

Yang, J., Ivanovic, B., Litany, O., Weng, X., Kim, S. W., Li, B., Che, T., Xu, D., Fidler, S., Pavone, M., and Wang, Y. Emernerf: Emergent spatial-temporal scene decomposition via self-supervision. *ArXiv*, abs/2311.02077, 2023a.

Yang, Z., Chen, Y., Wang, J., Manivasagam, S., Ma, W.-C., Yang, A. J., and Urtasun, R. Unisim: A neural closed-loop sensor simulator. In *CVPR*, 2023b.

Yoon, J., Singh, G., and Ahn, S. Robustifying sequential neural processes. In *International Conference on Machine Learning*, pp. 10861–10870. PMLR, 2020.

Yu, H.-X., Guibas, L. J., and Wu, J. Unsupervised discovery of object radiance fields. *ArXiv*, abs/2107.07905, 2021.

Zadaianchuk, A., Seitzer, M., and Martius, G. Object-centric learning for real-world videos by predicting temporal feature similarities. *arXiv preprint arXiv:2306.04829*, 2023.

Zhang, R., Che, T., Ivanovic, B., Wang, R., Pavone, M., Bengio, Y., and Paull, L. Robust and controllable object-centric learning through energy-based models. *arXiv preprint arXiv:2210.05519*, 2022.

A. Additional Related Work

Additional Works in Object-Centric Learning. Some works leverage GANs to generate scenes from object-centric noise latents (Van Steenkiste et al., 2020; Nguyen-Phuoc et al., 2020; Niemeyer & Geiger, 2021), however, these focus on static scenes and at the same time, do not provide an encoder to learn slots, although such an encoder may be learned as a post-processing step after the GAN is trained. (Du et al., 2021) seeks to learn 3D bounding boxes from 2D videos, however, it requires heavy use of manual hand-crafted priors to perform the back-projection from 2D masks to 3D bounding boxes.

Related Transformer Architectures for Videos. Some architectures have focused on applying transformers on videos (Bertasius et al., 2021; Arnab et al., 2021; Liu et al., 2022), however, note that these video transformer models work with patch-level representations and do not seek to learn object-centric slots in an unsupervised manner like ours.

3D Scene Representation Learning. Several works develop auto-encoders for learning to represent 3D scenes (Kumar et al., 2018; Singh et al., 2019; Yoon et al., 2020; Sajjadi et al., 2021; Seitzer et al., 2023), including without available camera pose information (Sajjadi et al., 2023). Another line of work seeks to develop generalizable NeRFs that can learn a NeRF representation without needing per-scene optimization. For this, they are equipped with an encoder that provides a representation given the input views via a single feedforward pass (Kosiorek et al., 2021). However, these representations are not decomposed in an object-centric manner. Static-dynamic decomposition has been pursued in this line (Sharma et al., 2022), however, it does not provide per-object decomposition without post-processing. Another parallel line of work aims to learn implicit scene representations via per-scene NeRF training (Yang et al., 2023b; Guo et al., 2023; Turki et al., 2023; Yang et al., 2023a; Wu et al., 2022). However, lacking an encoder, expensive training is needed to train them, and representations of distinct scenes carry different semantics. Furthermore, object-centric decomposition requires non-trivial post-processing.

Related Architectures for Compression and Attention. Another line of work seeks to summarize or compress the input activations into a small number of latents without asking for object-centric decomposition of the inputs. If such is the goal, our architecture may still find use in such applications. Along this line, several architectures have been proposed (Jaegle et al., 2021; Feng et al., 2022; Yoon et al., 2020; Lee et al., 2018; Wu et al., 2023a). In implementing the self-attention among slots, our axis-wise attention (along time and object axis) can be seen to be similar to axial attention (Ho et al., 2019), which provides memory cost savings.

B. Additional Experiment Results

B.1. FG-ARI Performance on Dynamic 3D Scenes

We report and compare the FG-ARI computed along several axes in order to measure segmentation consistency within a view, segmentation across time-steps but with the same camera, segmentation across cameras within a time-step, and segmentation across both cameras and time-steps. We note that when it comes to being consistent across both time-steps and cameras, our encoder outperforms all baselines. We also note that applying static 3D scene models (i.e., OSRT and uORF) per time-step independently can lead to poor consistency in the slots across time, as the slots of each time-step are not aware of the slots of the other time-steps.

B.2. Analysis of the Proposed Encoder on Dynamic 3D Datasets

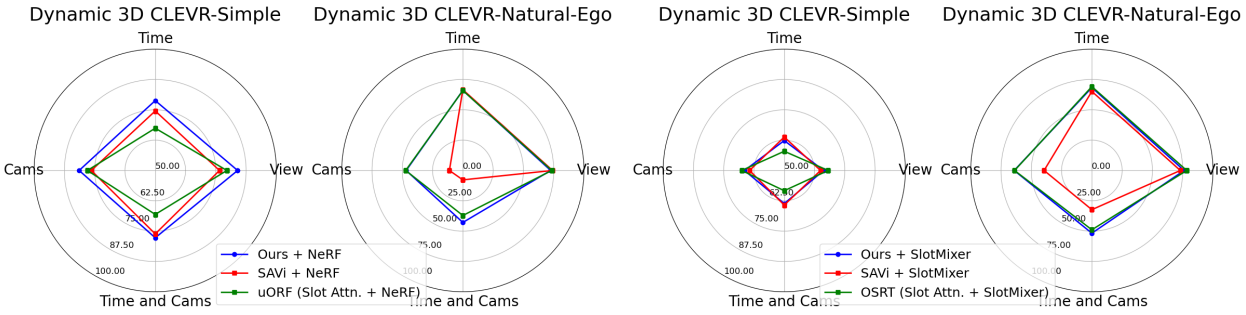
In Fig. 11, we ablate the design choices underlying our proposed encoder on the dynamic 3D datasets.

B.3. Effect of Decoupling Static and Dynamic Fields in NeRF Decoder

In Fig. 12, we ablate this aspect and report the results. We find that static-dynamic decoupling improves performance both in terms of representation quality as suggested by the linear-probing result as well as in terms of unsupervised segmentation. The improvement is more marked in the visually complex and egocentric CLEVR-Natural-Ego dataset.

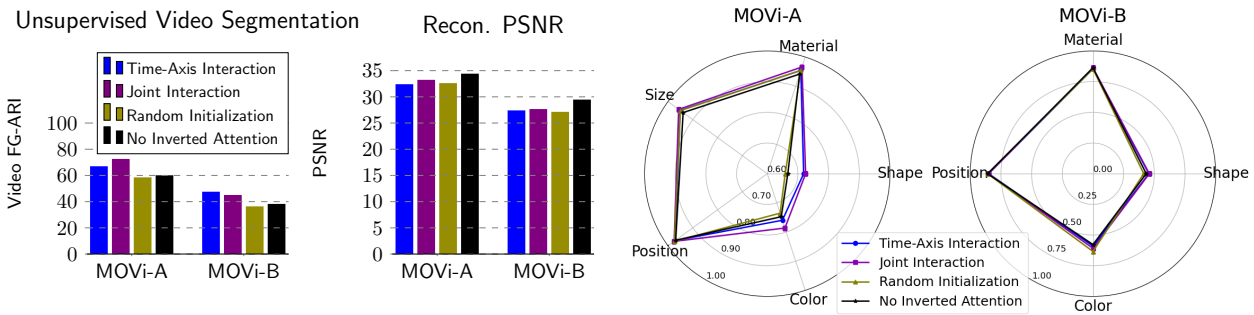
B.4. Length Generalization on Longer-Than-Training Sequences

In Fig. 13, we report the video-level FG-ARI score on MOVi-A and MOVi-B for sequence lengths: 6, 12, 18, and 24; while the training sequence length used is 6. For our model, we use a *sliding-window approach* to apply the model on longer sequence i.e., the slots of each time-step are inferred from the sequence of most recent 6 frames. For SAVi, we follow



(a) NeRF Decoder (b) SlotMixer Decoder

Figure 9. **FG-ARI Performance on Dynamic 3D Scenes.** We report and compare the FG-ARI computed along several axes in order to measure segmentation consistency within a view, segmentation across time-steps but with the same camera, segmentation across cameras within a time-step, and segmentation across both cameras and time-steps.



(a) FG-ARI and PSNR (b) Linear Probes

Figure 10. **Analysis of the Proposed Encoder on MOVi-A and MOVi-B.** We report and compare the following: *Left:* Unsupervised segmentation performance and Reconstruction quality. *Right:* Linear-probing performance.

the default approach by running the recurrence on the entire available sequence. On moderately longer sequence lengths i.e., 6 and 12, our model outperforms the baseline SAVi, showing better object-centric decomposition and consistency. On even longer sequences, i.e., sequence lengths 18 and 24, our performance is matched or slightly worse than SAVi, pointing towards an area of improvement for the proposed model.

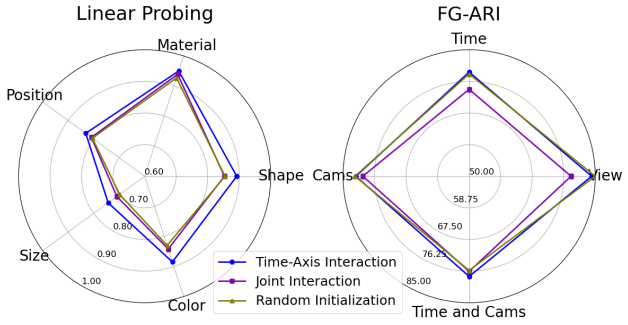
B.5. Video Transformer + Spatial Pooling

We conducted an experiment with a standard transformer encoder and tested it on MOVi-A and MOVi-B. We first describe how we implemented it, followed by the results.

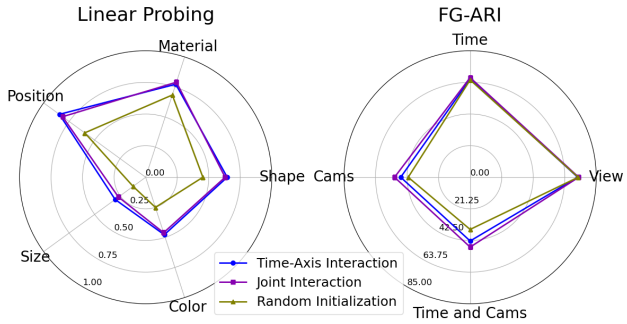
- **Encoder (Video Transformer + Spatial Pool).** We feed each video frame to a CNN to obtain a 16×16-shaped feature map per frame. For a 6-frame video, this leads to feature vectors of shape $6 \times 16 \times 16$. We flatten these features into a single set of size $6 \times 16 \times 16 = 1536$, add spatial and temporal positional embeddings, and feed them to a transformer. The output of the transformer is unflattened back to shape $6 \times 16 \times 16$. As we have too many features per frame (i.e., $16 \times 16 = 256$ features per frame), we cannot directly treat them as object representations or *slots*. Following (Kabra et al., 2021), we spatially pool the features to reduce them to a total of 16 features per frame—which we consider to be slots.

- **Decoder (Spatial Broadcast Decoder).** These slots are then decoded to reconstruct the input frames using an identical decoder as used for our model and other baselines on MOVi-A and MOVi-B.

We report the video-level FG-ARI to measure decomposition and linear probing performance to measure the slot representation quality. As can be seen in Table 1, using a video transformer with spatial pooling yields worse object-centric

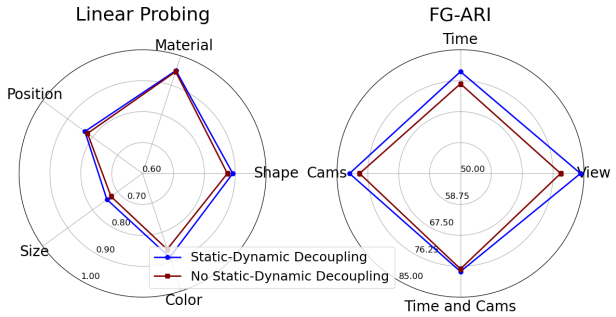


(a) Dynamic 3D CLEVR-Simple

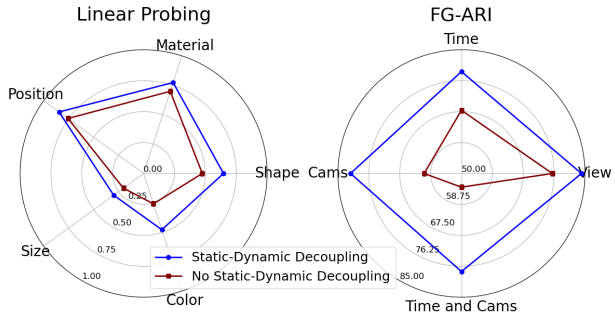


(b) Dynamic 3D CLEVR-Natural-Ego

Figure 11. **Analysis of the Proposed Encoder on Dynamic 3D Scene Understanding.** We report and compare the following: *Left:* Linear-probing performance. *Right:* Unsupervised segmentation performance.



(a) Dynamic 3D CLEVR-Simple



(b) Dynamic 3D CLEVR-Natural-Egocentric

Figure 12. **Effect of Static-Dynamic Decoupling in the NeRF Decoder.** We report and compare the following: *Left:* Unsupervised segmentation performance. *Right:* Linear-probing performance.

Dataset	Metric	Video Transformer + Spatial Pool	Ours
MOVi-A	Video FG-ARI	44.66	**67.01**
	Position Probe	0.952	**0.970**
	Color Probe	0.542	**0.759**
	Shape Probe	0.476	**0.718**
	Material Probe	0.868	**0.954**
	Size Probe	0.841	**0.950**
MOVi-B	Video FG-ARI	43.29	**47.57**
	Position Probe	0.837	**0.850**
	Color Probe	0.199	**0.589**
	Shape Probe	0.245	**0.456**
	Material Probe	0.721	**0.854**

Table 1. Analysis of a baseline model that leverages a video transformer followed by spatial pooling to obtain slots versus our proposed encoder.

decomposition and representation in the learned slots. Furthermore, on examining the segments qualitatively, we noted that when using the video transformer, the segments divide the image into square-shaped patches instead of meaningful objects.

B.6. Analysis of Memory and FLOPs

In this section, we report the memory consumption and the FLOPs incurred by the proposed slot learning architecture compared to the RNN-based baseline SAVi.

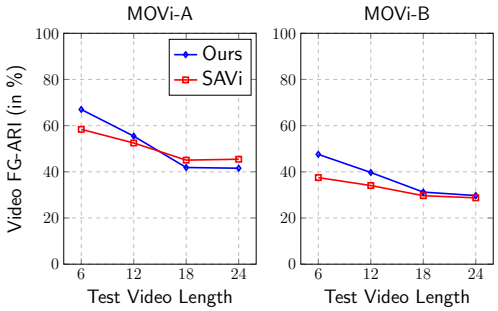


Figure 13. **Generalization on Sequence Length Longer than Training.** We report the video-level FG-ARI score on MOVi-A and MOVi-B for sequence lengths: 6, 12, 18, and 24; while the training sequence length used is 6. For our model, we use a sliding-window approach to apply the model on longer sequence i.e., the slots of each time-step are inferred from the sequence of most recent 6 frames. For SAVi, we follow the default approach by running the recurrence on the entire available sequence.

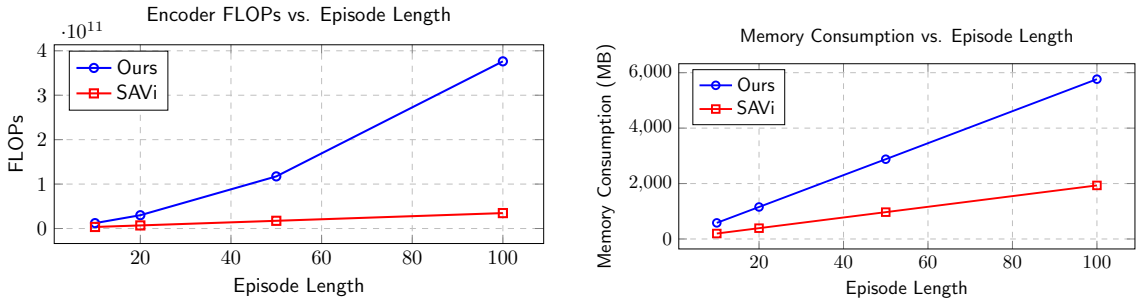


Figure 14. Analysis of FLOPs incurred and memory consumption of our model relative to SAVi.

From the plots in Fig. 14 and Fig. 3, we can make the following points:

1. First, this remarkable and significant as it says that our method allows processing more information while being faster simultaneously. While it is true that our model requires more FLOPs, if all FLOPs can be executed in parallel and efficiently on hardware such as a GPU, then having more FLOPs doesn't harm speed. This is clear from the wall-clock time reported in our paper where ours is indeed faster during training than SAVi.

2. Second, despite having fewer FLOPs, SAVi is unable to reap the benefits because it is unstable on longer sequences due to issues like gradient explosion/vanishing. This is clear from the Fig. 3 (top).

3. Third, today's best-performing architectures are primarily based on Transformers which also incur greater computation costs than their RNN-based predecessors. However, the Transformer is still widely used because its benefits outweigh its costs. We hope to see a similar outcome via our proposed encoder where the benefits of scalability should outweigh the costs in the long run.

4. Ours is the first step of bringing parallelization to slot learning for videos. It would be an interesting next step to address this quadratic cost, e.g., by leveraging recent advances like S4 (Gu et al., 2021).

B.7. Complexity of Internal Operations of SAVi versus Ours

Here, we argue why it is fair to compare SAVi versus Ours despite ours requiring greater FLOPs as noted in the previous section. For this, we first write down the complexity of the internal operations of SAVi and our proposed model in Table 2. Here, T is the number of time steps in a training episode. We can see that the higher computation of our model is due to gathering of information from the past inputs which is quadratic in episode length T. This is reasonable since capturing the temporal dependencies is where attention mechanisms excel over RNNs. On the other hand, the computation used for

processing the gathered information (i.e., the MLP) is of the same order as that of SAVi i.e. $O(T)$. For these reasons, we think that our comparison is fair.

To further ensure a fair comparison, we have used the same number of refinement iterations per time-step in the SAVi baseline as the number of layers we use for our model PSB. Also, we use an identical hidden size in all compared models.

Table 2. Comparison of Computational Complexity for SAVi and our model PSB.

	SAVi	**PSB (Ours)**
Bottom-Up Attention	$O(T)$	$O(T^2)$
Gather information from the past	$O(T)$	$O(T^2)$
Slot interaction (within each time-step)	$O(T)$	$O(T)$
MLP	$O(T)$	$O(T)$

B.8. Inference Complexity

The complexity during inference per each newly arriving input is $O(1)$ for SAVi and $O(W^2)$ for ours (and $O(W)$ if previous hidden states are cached), where W is the window size for the sliding context window for our model. Note that while the computation required by our model is greater, however, with GPU parallelization, the practical execution time can be significantly reduced. We will add this discussion to our revised version. Also, we discuss in the paper the potential of extending our model to achieve a constant inference cost by adopting SSM models.

B.9. Static and Dynamic Field Decoupling in NeRF Decoder

In Fig. 12, we compare the effect of having separate field models for the static topography and the dynamic objects of the 3D scene. We find that the static-dynamic decoupling leads to improved representation quality and object-wise decomposition of the dynamic objects. The benefit of such decoupling is more pronounced in the case of CLEVR-Natural-Ego dataset which is more visually complex than the CLEVR-Simple dataset.

C. Details of Dynamic 3D Scene Datasets

In this work, we synthesize two dynamic 3D scene datasets to evaluate scene understanding. We provide sample episodes in Fig. 15. In the following sections, we provide detailed specifications of these datasets.

C.1. Dynamic 3D CLEVR-Simple

The dataset consists of 2000 scene episodes. In each scene episode, the 2-4 objects are randomly instantiated with random velocities heading towards the center of the arena. As the scene plays out, videos are recorded by 5 pin-hole cameras randomly positioned on a hemisphere looking toward the origin of the scene. The objects can take one of four possible shapes: *sphere*, *cylinder*, *cube*, and *monkey*; one of 32 possible colors; one of 2 possible materials: *rubber* and *metal*; and one of 3 possible sizes. The full video length per episode is 32. We train the models on 6-length episodes randomly cropped from the 32-length episodes. The dataset is generated using Blender[2]. We extended the codebase of the original CLEVR dataset[3] to generate this dataset.

C.2. Dynamic 3D CLEVR-Nature-Ego

The dataset consists of 2000 scene episodes. In each scene episode, the 3-5 objects are randomly instantiated with random heading velocities. An ego object is also instantiated with a random velocity. The ego object is mounted with 6 cameras. As the scene plays out, the objects can collide with each other, with the static topography, and with the moving ego (which can conversely affect the motion of the ego object). The videos are recorded by pin-hole cameras. The objects can take one of six possible shapes: *sphere*, *cylinder*, *cube*, and *cone*, *icosahedron*, and *torus*; one of 32 possible colors; one of 2 possible materials: *rubber* and *metal*; and one of 3 possible sizes. The full video length per episode is 24. We train the models on

[2] https://www.blender.org

[3] https://github.com/facebookresearch/clevr-dataset-gen

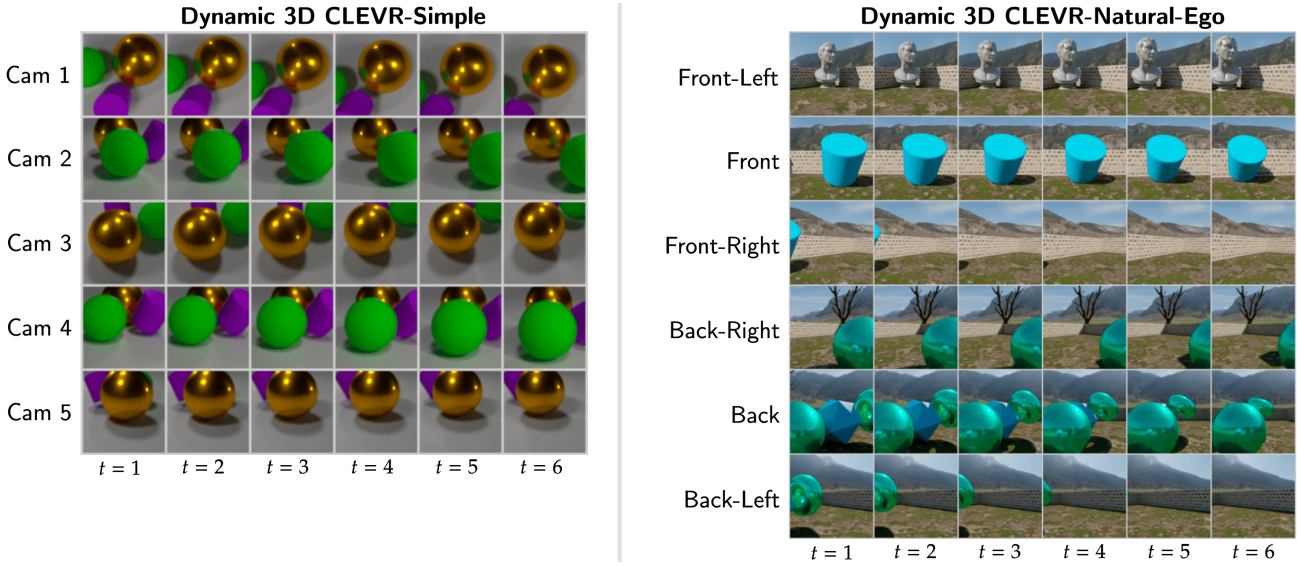


Figure 15. **Samples of the Proposed Dynamic 3D Scene Datasets.** On the left, we show 5 cameras and 6 time-steps of the Dynamic 3D CLEVR-Simple dataset. On the right, we show 6 ego-centric cameras and 6 time-steps of the Dynamic 3D CLEVR-Natural-Ego dataset.

6-length episodes randomly cropped from the 24-length episodes. The static topography consists of 4 rooms as visualized in Fig. 16 and the objects and ego are instantiated in one of the 4 rooms randomly within each episode. Each room consists of a unique static object such as an elephant, a tree, a cat statue, or a statue of a human face. The scene uses realistic lighting, background, and materials. The dataset is generated using Blender[4]. We extended the codebase of the original CLEVR dataset to generate this dataset.

D. Additional Model Details

D.1. Parallelizable Spatiotemporal Binder

D.1.1. Slot Initialization

The slots can be initialized either via learning or by sampling them from a learned Gaussian. We describe the initialization for both options.

Learned Initialization. We maintain N learned parameters $\theta_n^{\text{slot\_init}} \in \mathbb{R}^D$ for $n = 1, \dots, N$. Then, the NT slots are initialized as follows.

$$\mathbf{s}_{t,n}^{(0)} = \theta_n^{\text{slot\_init}}.$$

That is, the initialization is shared for each n-th slot across all time-steps.

Random Initialization. We maintain learned parameters: mean $\boldsymbol{\theta}^{\text{slot\_init\_mean}} \in \mathbb{R}^D$ and standard deviation $\boldsymbol{\theta}^{\text{slot\_init\_sigma}} \in \mathbb{R}^D$ that are shared for all slots. With these mean and standard deviation parameters, we sample N slots from a Gaussian and then broadcast the N samples across the T time-steps to initialize the TN slots.

$$\begin{aligned} \hat{\mathbf{s}}_n^{(0)} &\sim \mathcal{N}(\boldsymbol{\theta}^{\text{slot\_init\_mean}}, \boldsymbol{\theta}^{\text{slot\_init\_sigma}}), && \forall n \in 1, \dots, N, \\ \mathbf{s}_{t,n}^{(0)} &= \hat{\mathbf{s}}_n^{(0)}, && \forall t \in 1, \dots, T. \end{aligned}$$

Same as the case of learned initialization, the initialization is shared for each n-th slot across all time-steps.

[4] https://www.blender.org

Figure 16. **Environment Map for CLEVR-Nature-Ego Dataset.** We illustrate the top-view of the static topography and a randomly placed ego object that carries 6 cameras: `Front_Left`, `Front`, `Front_Right`, `Back_Left`, `Back`, and `Back_Right`. The ego and between 3 to 5 randomly chosen object shapes are spawned inside one of the randomly picked rooms. All objects including the ego are instantiated with a random position and velocity at the start of the episode. Then, the objects and the ego evolve through time under the influence of Blender's physics engine. 24-length RGB video recordings from the 6 ego cameras along with their respective camera poses constitute an episode of the dataset.

D.1.2. Bottom-Up Attention

Here, we describe a 1-headed version of the inverted dot-product attention that we employ to implement bottom-up attention in our proposed architecture.

$$
\begin{aligned}
Q &= W_Q(Q) && \in \mathbb{R}^{N_Q \times D}, \\
K &= W_K(K) && \in \mathbb{R}^{N_K \times D}, \\
V &= W_V(V) && \in \mathbb{R}^{N_K \times D}, \\
A &= \texttt{softmax}\left(\frac{QK^T}{\sqrt{D}} + \boldsymbol{\beta}, \quad \texttt{axis='queries'}\right) && \in \mathbb{R}^{N_Q \times N_K}, \\
A_{i,j} &= \frac{A_{i,j}}{\sum_{j=1}^{N_K} A_{i,j}} && \in \mathbb{R}, \\
\texttt{Attention}(Q, K, V) &= AV && \in \mathbb{R}^{N_Q \times D},
\end{aligned}
$$

where Q, K, V denote queries, keys, and values, respectively, W_Q, W_K, and W_V denote the linear projection matrices for the queries, keys and the values, respectively, $\boldsymbol{\beta}$ denotes the relative positional bias matrix and $\texttt{Attention}(Q, K, V)$ denotes the result of the attention. The argument `axis` denotes which axis of the attention weights is the softmax applied to. In this case, we apply softmax along the axis of queries followed by renormalization across keys in line with the competitive attention proposed by Locatello et al. (2020).

Multi-Headed Implementation. In the case of multi-headed attention, there are independent linear projection matrices per head. Furthermore, there is an additional output linear projection matrix W_O which takes a concatenation of the outputs of each head and maps it to an output result. In the multi-headed case, the softmax operation is applied across both queries and heads. Furthermore, in the multi-headed case, we have distinct, independently learned relative position bias matrices per each head.

Causal Masking. If a causal mask is provided, then the cells in the attention matrix $\frac{QK^T}{\sqrt{d_k}} + \boldsymbol{\beta}$ that are meant to be masked are replaced with $-\texttt{inf}$.

D.1.3. Attention Along Time Axis

This attention is implemented via standard multi-headed dot-product attention with causal masking and adding a relative position bias to the attention matrix.

D.1.4. Attention Along Object Axis

This attention is implemented via standard multi-headed dot-product attention. No causal masking or relative position bias is required in this case since this is simply the attention between the slots of the same time-step.

D.1.5. Non-Linearity

The non-linearity is implemented via a 2-layer MLP. We use the GELU non-linearity (Hendrycks & Gimpel, 2016) in the intermediate layer of the MLP.

D.1.6. Residual Connections and Layer-Normalizations

Prior to all attention layers as well as the MLP, we layer-normalize the inputs to the layers. The output of each layer is added to the value of the main branch before the application of the layer. That is, all operations are performed via residual connections, allowing our architecture to be potentially scaled to a deep stack containing a large number of layers.

D.1.7. Applying to Sequences Lengths Beyond Training Length

To apply the trained encoder on sequences longer than those shown in training, one may adopt a sliding window approach.

D.2. Object-Centric Learning on 2D Unposed Videos

D.2.1. CNN Backbone

The CNN backbone maps each image in a video to a set of features. We adopt the backbone that is also used by Kipf et al. (2022) and Singh et al. (2022). We describe it in Table 3. Like the previous works (Kipf et al., 2022; Singh et al., 2022), we add 2D positional embeddings to the CNN output, flatten it to form a set of features, apply layer normalization to the output, and feed to a 2-layer MLP with hidden size 192 and output size 192.

Table 3. The CNN backbone architecture used for object-centric learning on 2D unposed videos.

Layer	Kernel Size	Stride	Padding	Channels	Activation
Conv	5×5	1	2	192	ReLU
Conv	5×5	1	2	192	ReLU
Conv	5×5	1	2	192	ReLU
Conv	5×5	1	2	192	None

D.3. Parallelizable Spatiotemporal Binder

For the proposed PSB layer, we employ the configurations listed in the following Table 4.

D.3.1. Spatial Broadcast Decoder

We implement the spatial broadcast decoder (or simply SBD) in the same manner as Locatello et al. (2020) and Kipf et al. (2022). The SBD takes slot representations and spatially broadcasts them across a predefined grid, adds positional encodings, and feeds them through a CNN. The implementation details are summarized in Table 5.

The CNN outputs the RGB image and the alpha mixing logits. We pass the logits through a softmax operation across the N

Table 4. **PSB Configuration.** The table describes the hyperparameters used in the implementation of the PSB encoder for object-centric learning on 2D unposed videos.

Parameter	Value
Model Dimension	192
Number of PSB Layers	3
Number of Attention Heads	
- Bottom-Up Attention Heads	1 (for matching capacity with SAVi)
- Time-Axis Attention Heads	4
- Object-Axis Attention Heads	4
MLP Configuration	
- Number of Layers	2
- Hidden Dimension	768
- Output Dimension	192

Table 5. Spatial Broadcast Decoder architecture for transforming slot representations into RGB and alpha-mixing logits.

Layer	Kernel Size	Stride	Padding	Channels	Activation
Slot Normalization	-	-	-	-	-
Positional Embedding	-	-	-	-	-
ConvTranspose2d	5×5	2	2 (Output Padding: 1)	64	ReLU
ConvTranspose2d	5×5	2	2 (Output Padding: 1)	64	ReLU
ConvTranspose2d	5×5	2	2 (Output Padding: 1)	64	ReLU
ConvTranspose2d	5×5	2	2 (Output Padding: 1)	3 + 1	None

slot decodings. The softmax output acts as the mixing weights for the N RGB object images corresponding to the N slots, respectively. We use the same positional encoding approach as used in Locatello et al. (2020) and Kipf et al. (2022). That is, the 2D coordinate of each feature is linearly projected to an appropriately sized embedding.

D.3.2. Autoregressive Image Transformer Decoder

For the autoregressive image transformer decoder, we adopt the implementation of SLATE (Singh et al., 2021; 2022). That is, conditioned on the slots, we predict the VQ code representation of the image in an autoregressive manner. The VQ code is obtained via a Discrete VAE trained jointly with the rest of the model with codebook size 4096 as prescribed by Singh et al. (2021; 2022). The transformer implementation uses 8 layers with 4-heads and a hidden size of 192.

D.4. Object-Centric Learning on 3D Posed Multi-Camera Videos

D.4.1. Set Latent Scene Representation (SLSR) Backbone

The Set Latent Scene Representation or SLSR representation is computed by taking multiple posed views of a specific time-step and producing a collection or a set of features or latents, thus called Set Latent Scene Representation (Sajjadi et al., 2021; 2022). This is done by first applying a per-view CNN, flattening the resulting features, stacking these features together for all the views, and lastly, feeding this collection of features to a transformer.

CNN. In Table 6, we describe the CNN configuration that we use for the 3D Dynamic *CLEVR-Simple* dataset and in Table 7.

Transformer. The specifications of the cross-view transformer are provided in the Table 8.

D.4.2. Parallelizable Spatiotemporal Binder

For the proposed PSB layer, we employ the configurations listed in the following Table 9.

Table 6. Description of the CNN architecture used to compute SLSR for the dynamic 3D *CLEVR-Simple* dataset.

Layer	Kernel Size	Stride	Padding	Channels	Activation
GroupNorm	-	-	-	192	-
Residual Block 1					
GroupNorm	-	-	-	192	-
Conv2d	5×5	1	2	192	ReLU
Conv2d	5×5	1	2	192	-
Residual Block 2					
GroupNorm	-	-	-	192	-
Conv2d	5×5	1	2	192	ReLU
Conv2d	5×5	1	2	192	-
Size-Halving					
GroupNorm	-	-	-	192	-
Conv2d	5×5	2	2	192	-
Residual Block 3					
GroupNorm	-	-	-	192	-
Conv2d	5×5	1	2	192	ReLU
Conv2d	5×5	1	2	192	-
Residual Block 4					
GroupNorm	-	-	-	192	-
Conv2d	5×5	1	2	192	ReLU
Conv2d	5×5	1	2	192	-

D.4.3. NeRF Decoder

The NeRF decoder consists of a feedforward network that takes as input *i)* a slot vector $\mathbf{s}$, *ii)* 3D coordinate $\mathbf{o}$, and *iii)* the ray viewing direction $\mathbf{d}$ and outputs a density σ and an RGB color $\mathbf{c}$. We implement this MLP in the following manner.

Encoding Scalars. Before providing the 3D coordinate of the point and the 3D vector that denotes the viewing direction to an MLP, we would like to encode it as a vector. For this, we use sine-cosine embedding computed in the following manner.

$$\text{vectorize}(s) = \begin{bmatrix} \sin(\gamma_1 s) \\ \cos(\gamma_1 s) \\ \sin(\gamma_2 s) \\ \cos(\gamma_2 s) \\ \vdots \\ \sin(\gamma_{D/2} s) \\ \cos(\gamma_{D/2} s) \end{bmatrix}$$

where $\gamma_i = \pi \cdot 2^{i-1}/\texttt{max\_value}$ for $i = 1, 2, \ldots, D/2$, D is the dimensionality of the output vector, and `max_value` is a predefined maximum value that the scalar may take. In this work, we choose $D = 16$.

Network. The network consists of two MLP blocks. The first MLP block takes the embedding of $\mathbf{o}$ and the slot vector $\mathbf{s}$ and outputs an intermediate representation. The MLP is specified in Table 10. This intermediate representation is mapped to the density value σ via a linear head. Next, the same intermediate representation is added to an embedding of the ray direction $\mathbf{d}$ and is given to a second MLP block to output another hidden representation. This hidden representation is mapped to the color value $\mathbf{c}$ via a second linear head. This second MLP block is described in Table 11.

Static-Dynamic Decoupling and the Sky Head. To decouple the static and dynamic fields, we learn a dedicated NeRF decoder with separate weights that do not consume slots as input. The idea is to allow it to capture the background static topography that remains fixed across all episodes of the dataset without conditioning on slots whose role is to capture the dynamic and variable elements of the scene. To implement the static field, we learn two separate decoders, one for the static

topography and another for the sky or the far field. For capturing static topography, the decoder implementation is identical to that described above except that no slot input is given to the network. For implementing the sky field, we take an MLP architecture identical to the first MLP block of the object field implementation described above (and in Table 10). To this, we provide an encoding of the ray direction as input (instead of providing an embedding of the 3D point on the ray), take the output of this MLP, and map it to a color value via a linear head. To obtain the color of an image pixel incorporating the sky head, we shoot the corresponding ray from the camera, sample N_{bins} points along the ray, and integrate the colors along the ray as:

$$\sum_{i=1}^{N_{\text{bins}}} T_i \alpha_i \mathbf{c}_i + T_{N_{\text{bins}}+1} \mathbf{c}_{\text{sky}},$$

where $T_i = \prod_{j=1}^{i-1}(1-\alpha_i)$ is the transmittance, $\alpha_i = 1 - \exp(-\sigma_i ||\mathbf{o}_{i+1} - \mathbf{o}_i||_2)$ is the opacity and $\mathbf{c}_{\text{sky}}$ is the sky color obtained from the sky head for the ray's viewing direction. An overview of the NeRF with a separate sky head, static head, and object head is shown in Fig. 17 (left).

Shadow Head. Another element of modeling visually complex scenes is to model the scene lighting and shadows. To capture the shadows explicitly within the NeRF framework, we can optionally add a shadow head to obtain a shadow coefficient from the intermediate hidden state output by the first MLP block. The aim of the shadow coefficient ρ is to suppress the color value that would have been had there been no shadow at that 3D position. Specifically, the following formula is used to update the *shadeless* color value of a 3D point.

$$\sigma = \sigma_{\text{static}} + \sum_{n=1}^{N} \sigma_n, \qquad \mathbf{c}_{\text{shadeless}} = \frac{\sigma_{\text{static}} \mathbf{c}_{\text{static}} + \sum_{n=1}^{N} \mathbf{c}_n \sigma_n}{\sigma}, \qquad \mathbf{c} = \left[\rho_{\text{static}} \prod_{n=1}^{N} \rho_n \right] \mathbf{c}_{\text{shadeless}}$$

D.4.4. SlotMixer Decoder

The SlotMixer decoder directly maps an embedding of the ray shooting from the camera and maps it to a color value via a transformer-like implementation that conditions this function on the slot representation of the scene. Our implementation follows Sajjadi et al. (2022) whose overview is provided in Fig. 17 (right). For the allocation transformer, we use 3 transformer layers with 4-headed attention and hidden size 192. We follow Sajjadi et al. (2022) in implementing the mixing block as prescribed in the original paper. To implement the MLP, we use the specification provided in Table 12.

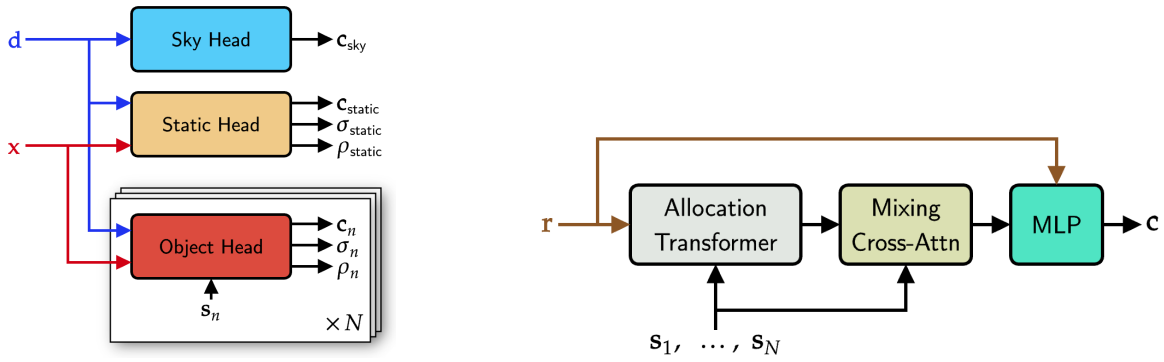


Figure 17. **3D Decoders.** *Left:* Our NeRF decoder is parametrized in terms of three MLPs to model the sky, the static topography and the objects, respectively. *Right:* SlotMixer decoder (Sajjadi et al., 2022) directly maps a camera ray to a pixel color conditioned on the slots via a transformer-like architecture.

D.5. Optimization

To train the models, we use a linear learning rate warm-up in the first 30000 training steps and use exponential decay thereafter with a half-life of 1M steps. We use a peak learning rate of $3e-4$. All models are trained to 300K steps. We used a batch size of 24 episodes with 6 time-steps per episode. We use the AdamW optimizer (Loshchilov & Hutter, 2017) with $\beta_1 = 0.9$ and $\beta_2 = 0.95$.

E. Additional Evalutation Details

E.1. Permutation-Invariant Slot Linear Probing

To perform permutation-invariant slot linear probing, we are provided with pre-trained and frozen slot representations of an episode denoted as $\mathbf{s}_{1:T,1:N}$. The ground-truth positions of the scene objects of an episode are denoted by $\mathbf{y}^{\text{position}}_{1:T,1:M}$, where M represents the number of actual objects in the scene. Let $\mathbf{s}_{1:T,\pi}$ represent a permutation of N slots taken M at a time, applying the same permutation π across all time steps. The first step involves determining the correct permutations $\pi_1, \ldots, \pi_B$ for all episodes within a large batch of B episodes. These permutations are used to correctly assign each slot to an object's label for training the linear probes. To determine these permutations, we follow the methodology outlined in (Dang-Nhu, 2021). We employ an approach akin to the EM algorithm, starting with arbitrary permutations $\pi_1, \ldots, \pi_B$. During the E-step, linear probes are trained using these permutations to predict the ground truth object positions $\mathbf{y}^{\text{position}}_{1:T,1:M}$. Subsequently, in the M-step, we iterate over all possible permutations to identify the one that best minimizes the probing error. The E and M steps are repeated until convergence is achieved. Finally, the determined permutations $\pi_1, \ldots, \pi_B$ are used to predict all object factors, such as color $\mathbf{y}^{\text{color}}_{1:T,1:M}$, shape $\mathbf{y}^{\text{shape}}_{1:T,1:M}$, size $\mathbf{y}^{\text{size}}_{1:T,1:M}$, etc. and performance is reported.

Table 7. Description of the CNN architecture used to compute SLSR for the dynamic 3D *CLEVR-Natural-Ego* dataset.

Layer	Kernel Size	Stride	Padding	Channels	Activation
GroupNorm	-	-	-	192	-
Residual Block 1					
GroupNorm	-	-	-	192	-
Conv2d	5×5	1	2	192	ReLU
Conv2d	5×5	1	2	192	-
Residual Block 2					
GroupNorm	-	-	-	192	-
Conv2d	5×5	1	2	192	ReLU
Conv2d	5×5	1	2	192	-
Size-Halving					
GroupNorm	-	-	-	192	-
Conv2d	5×5	2	2	192	-
Residual Block 3					
GroupNorm	-	-	-	192	-
Conv2d	5×5	1	2	192	ReLU
Conv2d	5×5	1	2	192	-
Residual Block 4					
GroupNorm	-	-	-	192	-
Conv2d	5×5	1	2	192	ReLU
Conv2d	5×5	1	2	192	-
Size-Halving					
GroupNorm	-	-	-	192	-
Conv2d	5×5	2	2	192	-
Residual Block 5					
GroupNorm	-	-	-	192	-
Conv2d	5×5	1	2	192	ReLU
Conv2d	5×5	1	2	192	-
Residual Block 6					
GroupNorm	-	-	-	192	-
Conv2d	5×5	1	2	192	ReLU
Conv2d	5×5	1	2	192	-

Parameter	Value
Number of Layers	3
Number of Heads	4
Hidden Dimensions	192

Table 8. Cross-View Transformer Specifications.

Table 9. **PSB Configuration.** The table describes the hyperparameters used in the implementation of the PSB encoder for object-centric learning on dynamic 3D posed multi-camera videos.

Parameter	Value
Model Dimension	192
Number of PSB Layers	3
Number of Attention Heads	
- Bottom-Up Attention Heads	4
- Time-Axis Attention Heads	4
- Object-Axis Attention Heads	4
MLP Configuration	
- Number of Layers	2
- Hidden Dimension	768
- Output Dimension	192

Layer	**Configuration**
LayerNorm	192
Residual Block 1	
LayerNorm	192
Linear	$192 \rightarrow 192$
ReLU	-
Linear	$192 \rightarrow 192$
Residual Block 2	
LayerNorm	192
Linear	$192 \rightarrow 192$
ReLU	-
Linear	$192 \rightarrow 192$

Table 10. Specifications of the first block of the MLP implementation of the NeRF decoder.

Layer	**Configuration**
LayerNorm	192
Residual Block 1	
LayerNorm	192
Linear	$192 \rightarrow 192$
ReLU	-
Linear	$192 \rightarrow 192$

Table 11. Specifications of the second block of the MLP implementation of the NeRF decoder.

Layer	Configuration
LayerNorm	192
Residual Block 1	
LayerNorm	192
Linear	$192 \rightarrow 192$
ReLU	-
Linear	$192 \rightarrow 192$
Residual Block 2	
LayerNorm	192
Linear	$192 \rightarrow 192$
ReLU	-
Linear	$192 \rightarrow 192$
Residual Block 3	
LayerNorm	192
Linear	$192 \rightarrow 192$
ReLU	-
Linear	$192 \rightarrow 192$
Residual Block 4	
LayerNorm	192
Linear	$192 \rightarrow 192$
ReLU	-
Linear	$192 \rightarrow 192$

Table 12. MLP Specifications for the SlotMixer decoder.