Object Discovery In Multi-Scene NeRFs

Junkeun Yi

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-72

http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-72.html

May 9, 2024

Copyright © 2024, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Acknowledgement

I would like to thank Dr. Roei Herzig and Professor Trevor Darrell for their guidance these past two years. I would especially like to thank Dr. Herzig for entrusting me, an undergraduate with no prior Machine Learning research experience, to take control in working on interesting projects. I would also like to thank Chung Min Kim and Justin Kerr for their invaluable input in completing this thesis. To all those who have mentored and worked with me throughout my research journey, thank you for introducing me to this world and paddling through it with me. Finally, thank you to my friends for supporting me in my times of struggle.

Object Discovery In Multi-Scene NeRFs

by Junkeun Yi

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

[illegible]

Professor Trevor Darrell
Research Advisor

5/6/2024

(Date)

* * * * * * *

[illegible]

Professor Jiantao Jiao
Second Reader

05/06/2024

(Date)

Object Discovery In Multi-Scene NeRFs

by

Junkeun Yi

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Trevor Darrell, Chair
Professor Jiantao Jiao, second reader

Spring 2024

Object Discovery In Multi-Scene NeRFs

Copyright 2024
by
Junkeun Yi

Abstract

Object Discovery In Multi-Scene NeRFs

by

Junkeun Yi

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Trevor Darrell, Chair

Unsupervised Object-Centric Learning is an effective means of learning reusable representations that can explain data in meaningful partitions, through creating a separate representation for each 'object' in the scene [3]. In recent literature, Slot Attention [11] has shown to be an effective unsupervised method for learning object representations, showing effectiveness in tasks such as object discovery [10], composition [18], and novel view synthesis [16]. However, while showing remarkable results when operating on synthetic datasets such as CLEVR [5] and MultishapeNet [17], it has yet to show promising results when applied to natural images. Meanwhile, NeRFs [13] have shown that given many views, a neural field can be created that contains a dense and high resolution representation of a scene. In this thesis, we propose a new multi-scene NeRF method which leverages the ability of Slot Attention to discover object representations in the input data to train a model capable of representing many scenes. We demonstrate that when different scenes share common objects in the foreground area, the slot attention method can discover reusable representations of these objects to be shared in the volumetric rendering process.

To my family: Sung ho, Sun mi, and Shin woo.

Thank you.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank Dr. Roei Herzig and Professor Trevor Darrell for their guidance these past two years. I would especially like to thank Dr. Herzig for entrusting me, an undergraduate with no prior Machine Learning research experience, to take control in working on interesting projects. I would also like to thank Chung Min Kim and Justin Kerr for their invaluable input in completing this thesis. To all those who have mentored and worked with me throughout my research journey, thank you for introducing me to this world and paddling through it with me. Finally, thank you to my friends for supporting me in my times of struggle.

Chapter 1

Introduction

Given a handful of objects, one can create many different scenes by scattering the objects in different layouts. While training a NeRF [13] on each scene can be easily done using ready tools such as Nerfstudio [21], it can be counter intuitive as to why one should have to learn a new model from scratch for each scene when they share much in common. In this setting, it may be beneficial to create representations for the shared objects between the scenes, and reuse these representations for jointly training each scene containing these objects.

We start with chapter 2 by introducing the related works in object-centric learning and neural novel view synthesis.

In chapter 3, we propose a novel method to train a NeRF model to learn object representations to reuse in a multi-scene setting. Crucially, these object representations are learned in an unsupervised manner, and thus are discovered by the model in a fashion that aids its field representation of each scene.

In chapter 4, we show the experiments done on the method, changing the difficulty of each scene by varying the objects and background. We also show ablation experiments done on the method to show the impact of the design choices made.

Finally, in chapter 5 we provide an analysis for object-centric learning in the novel view synthesis setting and end with proposals for future research.

Chapter 2

Related Works

2.1 Object-Centric Learning

Object-Centric Learning is a way of addressing the binding problem in neural networks, in which the ability of a network to bind information into entities that can lead to human-level generalization [3]. Slot Attention [11] is an attention mechanism which inverts the commonly used dot-product attention [12][22] by applying the softmax operator over the queries. As the softmax operation leads to the queries competing to represent different parts of the input, it leads to compact and disentangled representations of the input, called slots. This technique has shown to be effective in object discovery [11], video prediction [10][23], and novel view synthesis [17][24][19][15], in which the slots are used as a bottleneck representation before the decoder layer such that the outputs can be represented by a compact set of vectors. Our work deviates from previous work in that we focus mostly in real-world data whereas the focus of previous work is in the synthetic domain. In addition, we guide each slot to represent a specific object, such as a single chair, while previous works use the slots to represent a template for a class of objects, such as the general notion of chairs.

2.2 Neural Novel View Synthesis

Neural Radiance Fields

In the NeRF [13] methods of novel view synthesis, a radiance field is trained to represent coordinate-based representations of a scene. These models employ volumetric rendering [6] to create a view of the scene. Recently, NeRF methods have evolved to include the use of proposal samplers [1] and multiresolution hash encodings [14] to increase the speed of training and quality of the generated views. We reuse the concepts introduced in these papers by creating a NeRF method resuing proposal samplers [1] and sinusoidal positional encodings [13].

Object-Centric Novel View Synthesis

Many NeRF models have been trained to visualize object-centric views of a scene, including ObSuRF [20], uORF [24], and sVORF [15], the two latter models employing Slot Attention. In addition to NeRF models, models may use a light field to represent a scene, such as in COLF [19], and train a transformer model, such as in OSRT [17]. Our method deviates from the above methods in that instead of creating a model that can represent an infinite number of scenes given a reference image, we train only for a fixed number of scenes with no reference image. In addition, we redirect the focus of the model in learning object representations for specific objects in the scene instead of creating object template representations, which allows us to train a single NeRF model for all objects instead of having to train multiple models conditioned on each slot such as in uORF, sVORF and COLF. In a separate direction, NeRFs can be augmented to create hierarchical feature fields aligning to objects in many scales, as seen in GARField [8].

Chapter 3

Method

3.1 Preliminaries

In this section, we introduce some preliminaries to our method.

Neural Radiance Fields (NeRFs)

In the classic NeRF setting, a multilayer perceptron (MLP) is trained to represent a scene as a volumetric field, in which each particle in the field is associated with blocking and emitting light. A ray, represented as $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, has origin $\mathbf{o}$ and direction $\mathbf{d}$, and each pixel of the camera can be represented by such a ray to create a bundle of rays for each camera. For distance $t_k \in \mathbf{t}$ in t_n to t_f of the camera's near and far plane, each 3D position along the ray is transformed by a sinusoidal positional encoding to create the positional encoding of the point,

$$\gamma(\mathbf{r}(t_k)) = [\sin(\mathbf{r}(t_k)), \cos(\mathbf{r}(t_k)), \cdots, \sin(2^{L-1}\mathbf{r}(t_k)), \cos(2^{L-1}\mathbf{r}(t_k))]^\top \tag{3.1}$$

where L is a hyperparameter. The positional encoding is passed to an MLP parameterized by weights θ to output the density τ and RGB color $\mathbf{c}$:

$$\forall t_k \in \mathbf{t}, [\tau_k, \mathbf{c}_k] = \mathrm{MLP}(\gamma(\mathbf{r}(t_k)); \theta) \tag{3.2}$$

For the RGB color prediction, the MLP also inputs the view direction as well. The density and color estimate for each position is then used for the volumetric rendering, represented as the integral estimate

$$\mathbf{C}(\mathbf{r}; \theta; t) = \sum_k T_k(1 - \exp(-\tau_k(t_{k+1} - t_k)))\mathbf{c}_k \tag{3.3}$$

where $T_k = \exp(-\sum_{k'<k}(t_{k'+1} - t_{k'}))$ and $\mathbf{C}(\mathbf{r}; \theta; t)$ is the final color prediction for the given pixel represented by the ray $\mathbf{r}$.

Finally, given a set of images of a scene and known camera poses, the sum of squared differences between the input pixels and predicted pixel values is used as the loss for training the network with gradient descent.

Slot Attention

In the classic dot-product attention setting, there are three sets of vectors: the query $Q \in \mathbb{R}^{N_Q \times D}$, key $K \in \mathbb{R}^{N \times D}$, and value $V \in \mathbb{R}^{N \times D}$ vectors. K and V are created from the input, and Q can be separately initialized to apply cross-attention. Slot attention inverts the original method by applying the softmax operator over the queries, and by setting $N_Q << N$, the small number of query vectors, called slots, compete to explain the input. Formally, the attention between each input i and slot j is:

$$\text{attn}_{i,j} \coloneqq \frac{e^{M_{i,j}}}{\sum_l e^{M_{i,l}}}, \tag{3.4}$$

$$M \coloneqq \frac{1}{\sqrt{D}} K \cdot Q^\top \in \mathbb{R}^{\mathbb{N} \times \mathbb{N}_{\mathbb{Q}}} \tag{3.5}$$

where the softmax occurs over the slots Q and the normalization ensures the slots attend to all of the input. The slots are aggregated using a weighted mean:

$$\text{updates} \coloneqq W^\top \cdot V \in \mathbb{R}^{\mathbb{N}_{\mathbb{Q}} \times \mathbb{D}} \tag{3.6}$$

$$W_{i,j} \coloneqq \frac{\text{attn}_{i,j}}{\sum_{l=1}^{N} \text{attn}_{l,j}} \tag{3.7}$$

The slots, along with the updates, are passed into a gated recurrent unit [2] and an MLP with a residual connection [4] to create a new set of slots. Typically, slot attention is applied over several iterations to improve performance.

3.2 Method

In this section we describe our method for creating an object-centric multi-scene neural radiance field. Our model has three components, the scene-conditioned feature embedding module, the cross-scene slot attention module, and the scene reconstruction module, which roughly equate to an encoder, processor and decoder module. An overview of the model can be seen in figure 3.1.

Scene-Conditioned Feature Embedding

In our setup we assume there are N_s scenes with shared objects, with N_c views with known camera poses per scene. To train a NeRF field, we must first create a feature embedding for each 3D position of the ray associated with each pixel. To do so, we encode the 3D position using a sinusoidal encoding [13] and add a learned scene embedding $\mathbf{s}_i$ to each encoding corresponding to scene i. Finally, the resulting ray is passed into a Transformer Encoder [22] f_{enc} to get the feature embedding $\mathbf{x}_{r;i}$:

$$\mathbf{x}_{r;i} = f_{enc}(\gamma(\mathbf{r}(\mathbf{t}; i)) + s_i; \phi_{enc}) \in \mathbf{R}^{L \times D} \tag{3.8}$$

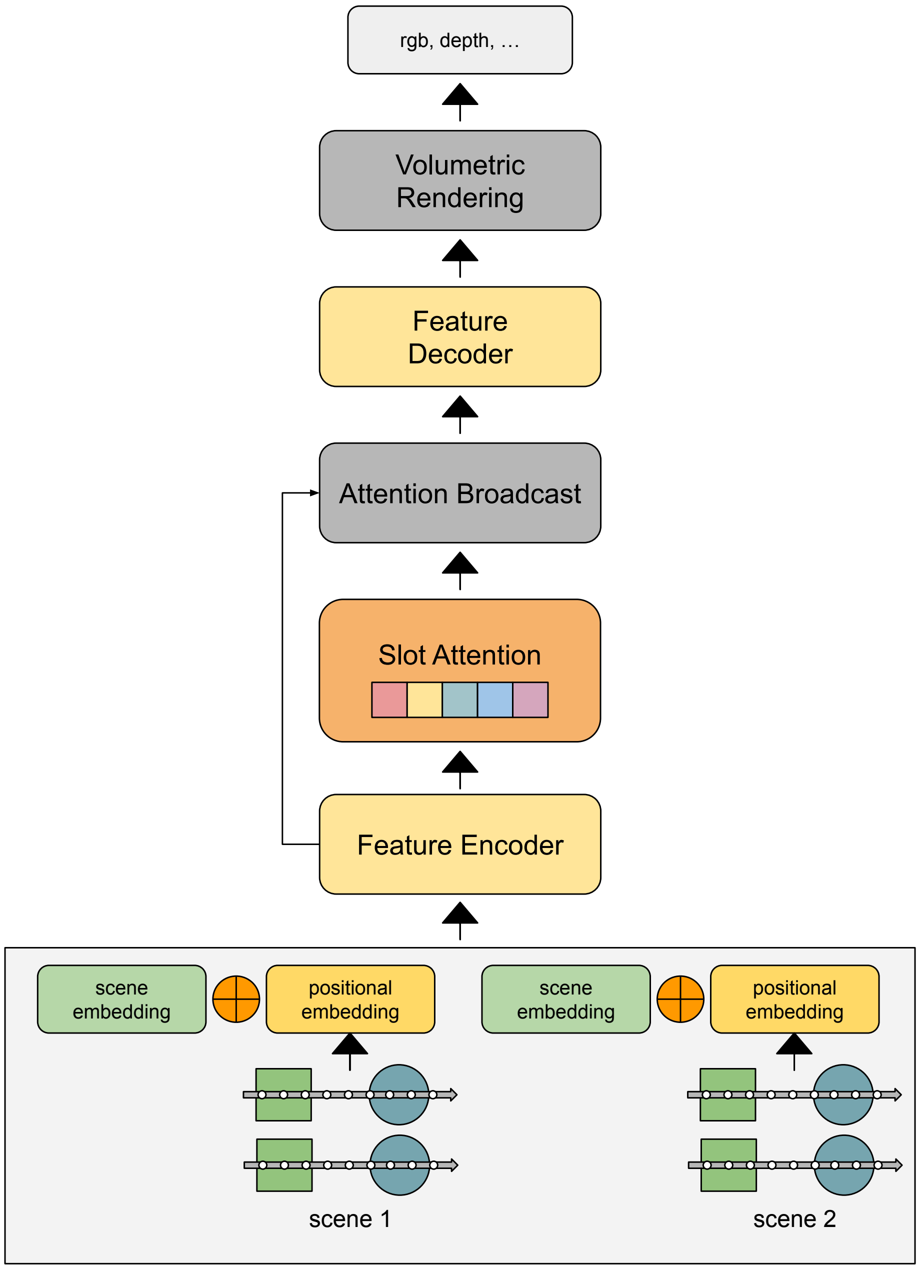

Figure 3.1: Model Overview of proposed method. Given ray samples from different scenes, 1) the scenes are encoded and scene-conditioned before being passed into the feature encoder. 2) The resulting features are passed into the cross-scene slot attention module to allocate each 3D position into a set of slots. 3) The resulting slots are broadcasted to the size of the ray samples by element-wise multiplication with attention weights with respect to the features, and passed into the feature decoder before finally going through volumetric rendering.

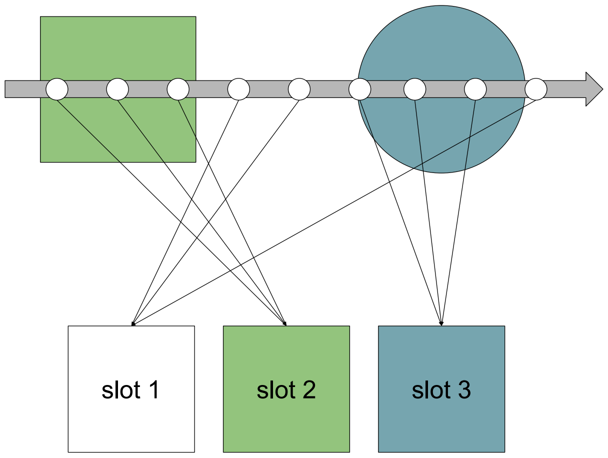

Figure 3.2: Example visualization of the goal of slot attention applied to a ray. Each 3D position (represented as white circles) in the ray will be aggregated into a slot most representative of the entity in the 3D position. Here, 3 slots for the square, circle, and empty space are shown. In practice, the same slots are trained to explain 3D positions across multiple rays from a variable number of scenes at the same time.

where L is the number of samples per ray and ϕ_{enc} are the weights of the feature encoder. Given N rays in a batch, the resulting feature bundle is $\mathbf{X} \in \mathbb{R}^{N,L,D}$. The resulting feature bundle is sent to the Cross-Scene Slot Attention Module to be encoded into a bottleneck representation consisting of a set number of slots.

Cross-Scene Slot Attention

In the Cross-Scene Slot Attention module, we apply slot attention to the features $\mathbf{X}$ returned from the previous section. Given features $\mathbf{X} \in \mathbb{R}^{N,L,D}$, we reshape it to one long sequence of shape $\mathbf{X} \in \mathbb{R}^{(N\times L),D}$ where the sequence length is $N \times L$. We also initialize **slots** $\in \mathbb{R}^{K,D}$, where K is the number of slots. Applying slot attention to the features, we get:

$$\mathbf{Z} = \text{slot\_attention}(\mathbf{X} \in \mathbb{R}^{(N\times L),D}; \textbf{slots} \in \mathbb{R}^{K,D}, \phi_{slots}) \in \mathbb{R}^{K,D} \tag{3.9}$$

where $\mathbf{Z} \in \mathbb{R}^{K,D}$ are the resulting slots that act as a compact representation of the rays across the scenes and ϕ_{slots} are the weights of the slot attention module.

One thing to note is that the attention is applied over the entire set of 3D positions instead of per ray. This is to ensure that the slots are able to see all objects in the scene instead of acting over one pixel value, enabling object-wise representation of the slots.

Another crucial factor is the application of slot attention over rays from different scenes. This is to ensure that the slots learn to compare objects across scenes. When trained without this cross-scene visibility, the model finds difficulty in reusing the slot representations across scenes in more complicated datasets. Finally, the resulting slots $\mathbf{Z}$ are sent to the scene reconstruction module for volumetric rendering and view reconstruction.

Scene Reconstruction

To recover the 3D position-wise features for volumetric rendering, the slots $\mathbf{Z} \in \mathbb{R}^{K,D}$ must be rebroadcasted back into the shape of the ray samples: (N, L). To do this, we apply an adaptation of the slot-mixer method proposed in OSRT [17]. We first compute the attention weights between the slots and the input features, expand the slots by the shape of the attention, perform an element-wise multiplication and sum over the slots:

$$\tilde{\mathbf{Z}} = \mathrm{sum}(\mathrm{attn} \otimes \mathrm{expand}(\mathbf{Z}); \mathrm{axis{=}slots}) \in \mathbb{R}^{N,L,D} \tag{3.10}$$

$$\mathrm{attn} \coloneqq \mathbf{X} \cdot \mathbf{Z} \in \mathbb{R}^{(N \times L),K} \tag{3.11}$$

The resulting 3D position-wide features $\tilde{\mathbf{Z}} \in \mathbb{R}^{N,L,D}$ are then passed to the feature decoder.

The feature decoder is a Transformer Encoder [22] trained to process each ray separately. It also consists of a density head and RGB head, which are MLPs that decode each 3D position feature into the density τ and RGB color $\mathbf{c}$. Together, the resulting densities and colors are:

$$[\tau_{1,\cdots,L}, \mathbf{c}_{1,\cdots,L}] = f_{dec}(\tilde{\mathbf{z}}_r \in \mathbb{R}^{L,D}; \phi_{dec}) \tag{3.12}$$

for each of the L 3D positions in the ray feature vector $\tilde{\mathbf{z}} \in \mathbb{R}^{L,D}$, where ϕ_{dec} are the weights of the feature decoder.

Finally, the $N \times L$ set of densities τ and colors $\mathbf{c}$ are used in volumetric rendering, where the resulting RGB color

$$\mathbf{C}(\mathbf{r}; i; t; \phi_{enc}, \phi_{slot}, \phi_{dec}) \tag{3.13}$$

is used as the predicted pixel value passed into the standard NeRF RGB reconstruction loss, training the weights $\phi_{enc}, \phi_{slot}, \phi_{dec}$ through gradient descent.

Chapter 4

Experiments

In this chapter, we describe the experimental results from applications of this method as well as any ablations done.

4.1 Datasets

Multiple sets of datasets were assembled through the Nerfstudio [21] platform to create set of scenes with shared objects.

- **small objects**: The small objects dataset is a set of 4 scenes with up to 6 rigid objects scattered in different layouts on top of an unmoved table.
- **floor objects**: The floor objects dataset is a set of 11 scenes with up to 6 rigid objects scattered in different layouts on top of the floor with partial occlusion of objects. Unlike the small objects scenes, there is no fixed object across scenes like the table.
- **desk objects**: The desk objects dataset is a set of 2 scenes that are similar to the small objects scene, but with a complicated background and more objects.
- **tiger**: The tiger dataset is a set of 3 scenes with one shared object, a stuffed animal tiger oriented in different layouts. Since the tiger is a non-rigid object, the aim is for the slots to bind to parts of the tiger such as the limbs separately.

4.2 Setup

Each batch consists of 4096 rays sampled randomly from any of N_s scenes in the dataset. For each batch the scenes are sampled randomly, and the batch is separately equally into $4096/N_s$ rays from each scene. The size of the input encodings are set to 48, while the slot sizes are set between 64 and 128 depending on the dataset. The number of slots are varied from 16 to 32 depending on the dataset. The feature encoder and decoder consist of 4 layer

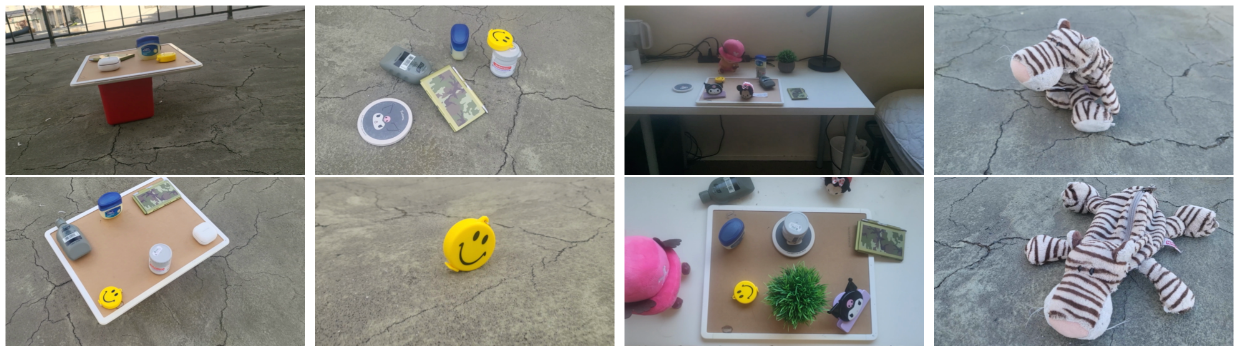
Figure 4.1: Example frames from each dataset. From left to right: small objects, floor objects, desk objects, tiger

Dataset	PSNR	SSIM	LPIPS
small objects	24.228	0.6603	0.4837
floor objects	24.704	0.5219	0.683
desk objects	23.282	0.7869	0.3152
tiger	21.18	0.4285	0.6744

Table 4.1: View Synthesis Quality

transformers with model dimensions matching the slot dimensions, while the density and RGB head are 2-layer MLPs. The models are trained with Adam [9] with a learning rate of 0.001 decreasing in an exponential decay schedule to 0.0001 for up to 300000 steps. Each model is trained on a single RTX 3090 GPU.

4.3 Novel View Synthesis and Object Discovery

We provide quantitative results on novel view synthesis for each dataset in table 4.1. Outputs from held out images not seen during training shown in figure 4.2. It can be seen that in the harder datasets, the model reconstruction quality is lower. Especially for the tiger dataset, where the texture is not simple (with stripes), the slots struggle to find ‘object’ parts and poor quality is reflected in the metrics. It should also be noted the predicted RGB quality is not on par with commonly available NeRF models, as can be seen in the low metric scores.

Qualitative Analysis

In this section, we discuss the qualitative results for outputs of applications of our model. In figure 4.3, rendered views of the different scenes in the small objects dataset as well as

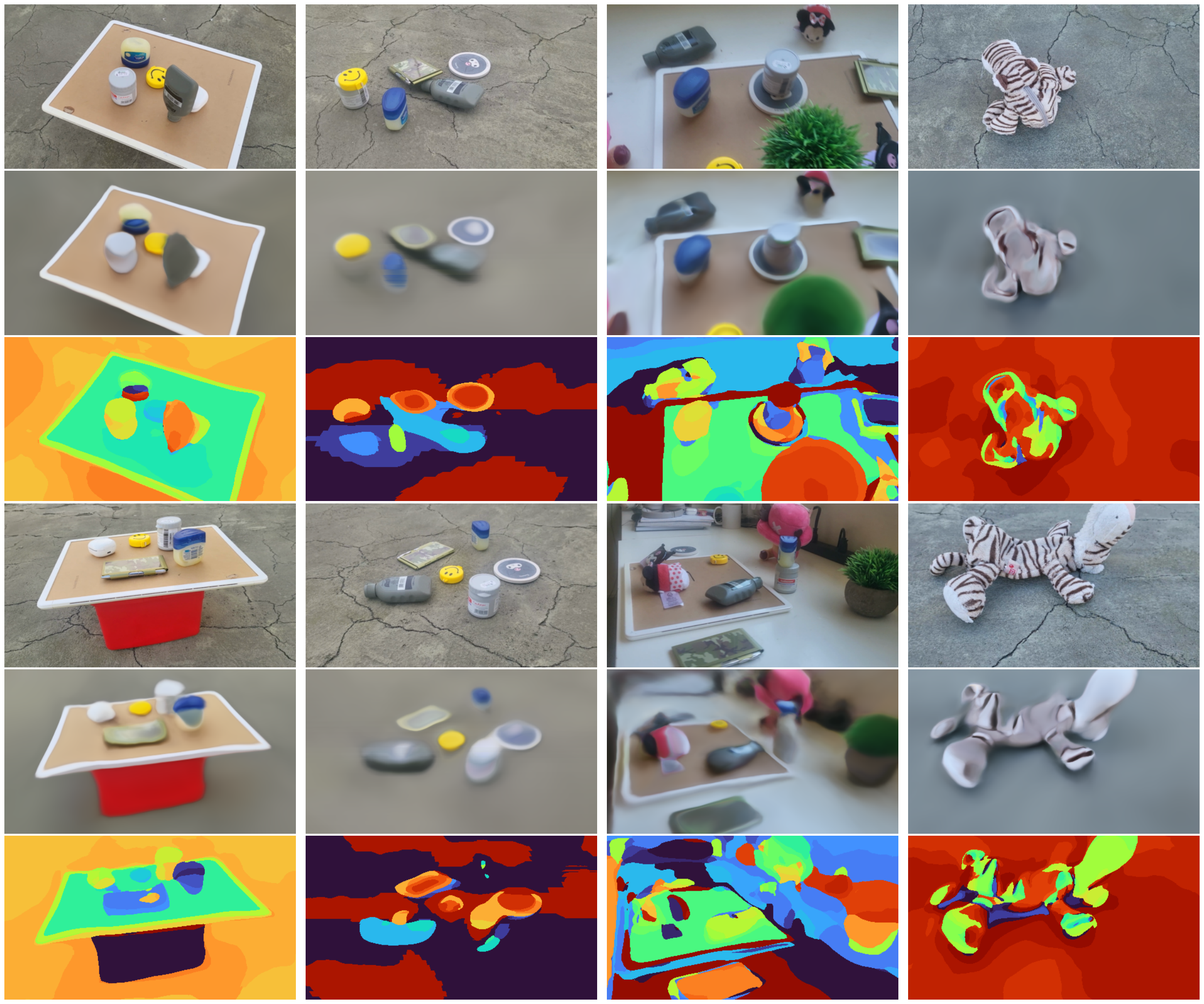

Figure 4.2: Example outputs from dataset. Tiled as ground truth image, reconstructed image, and attention map.

the visualized attention map is shown. Since the attention map happens over the entire ray samples instead of at the pixel level, the slot which are activated the most often along the ray is visualized and set as the label for the visualized attention map. This model was trained for 150000 steps with slots size of 128 and 32 slots, with 4 scenes sampled per batch. It can be seen that many of the objects have a unique slot representing them, and that many of these slots are consistent across the different scenes (thus represented by the same color). However, there are still issues with some objects having multiple slots representing them across different scenes, as well as the RGB quality not being very detailed.

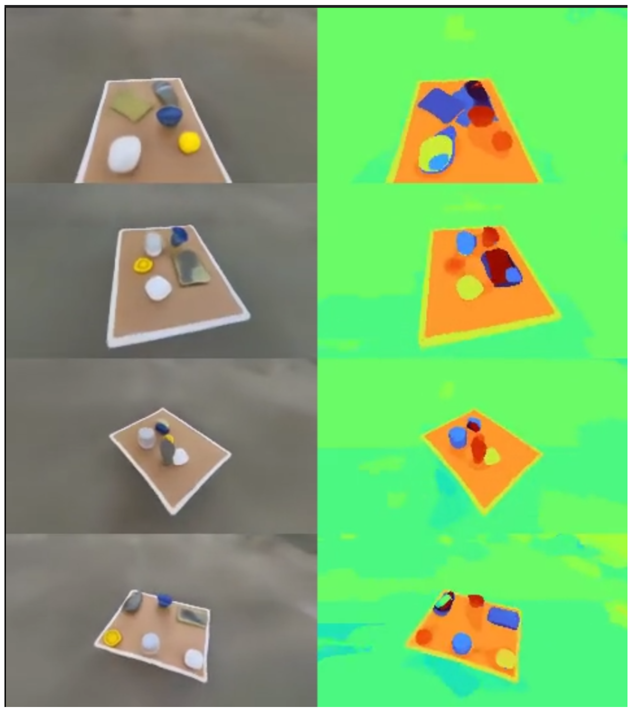
Figure 4.3: Rendered views on the small objects dataset. Each row is a separate scene. Left: RGB, Right: Attention Mask.

Dataset	PSNR	SSIM	LPIPS
floor objects - base	**24.319**	**0.5193**	0.6933
floor objects - incremental	23.989	0.5162	**0.6926**

Table 4.2: Incremental Learning Quality at 120000 steps, averaged metrics over all 11 scenes. The ablated model has only started training on all the scenes at step 100000

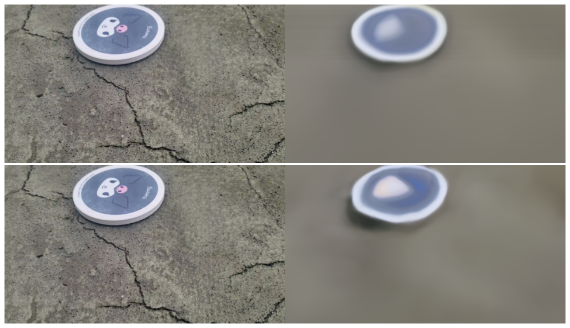

Figure 4.4: Top: base model output, Bottom: ablated model output

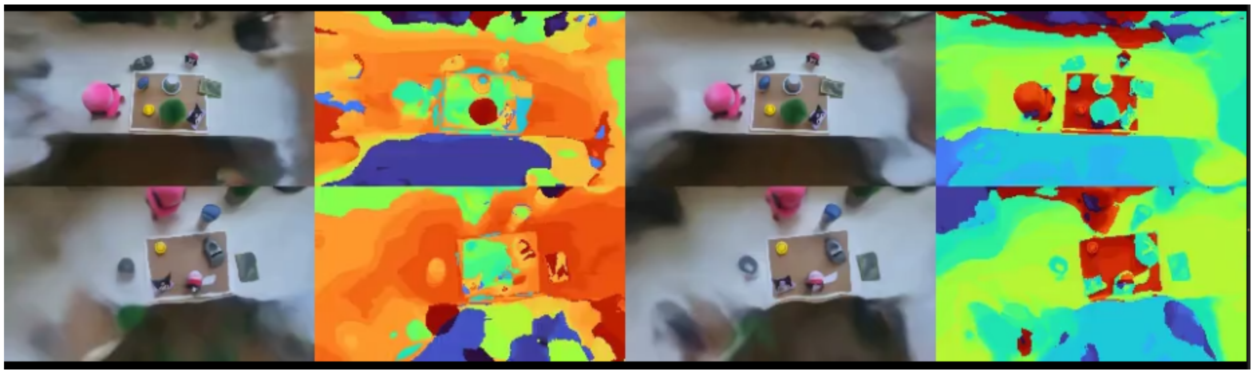

Figure 4.5: Effect of the cross-scene attention in the desk objects dataset. Left 2 columns are outputs from a model trained to perform without cross-scene attention, instead applying slot attention only within rays of each scene. The right two columns are outputs from the base model trained with cross-scene slot attention between rays. Visualized are the rendered RGB and attention maps for each of the 2 scenes in the dataset.

4.4 Ablations

Incrementally Adding Scenes

We ablate the floor objects scene by trying a training regimen where initially only 2 scenes are trained, and after every 10000 steps we incrementally add another scene to be trained as well. Quantitative results can be seen in 4.2 for results at 120000 steps. The quality of reconstruction on both models perform similarly. In figure 4.4, the RGB reconstruction of a scene seen only after step 100000 is visualized at step 120000. On the top is the base model output, where the scene was trained from step 0. It can be seen that even though in the ablated model the scene was only trained for 20000 steps, the visual quality is similar to that of the base model.

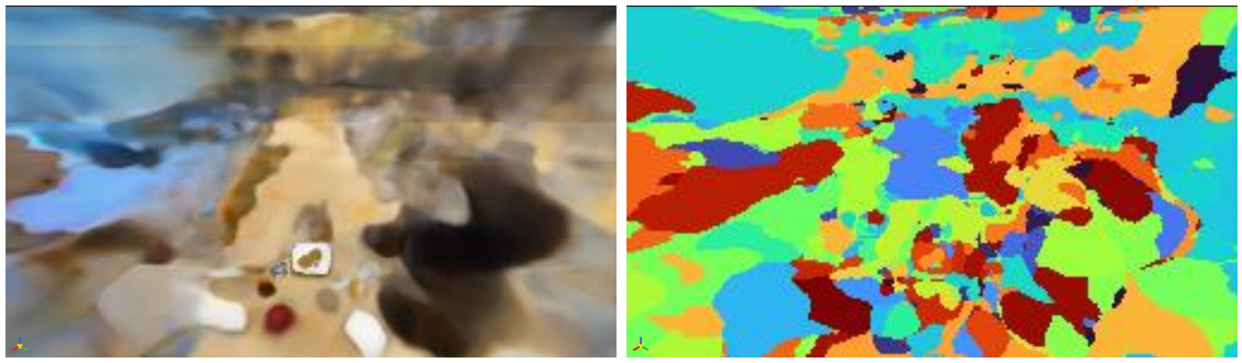

Figure 4.6: Teatime dataset from LERF trained on 4 rotations of the scene. Only one scene is visualized, with predicted RGB and attention map.

Cross-Scene Attention

We examine the effects of the Cross-Scene Slot Attention by ablating the model to perform slot attention only within a scene instead of applying the slot attention across scenes. In figure 4.5, we visualize outputs of this ablation to the desk objects dataset. In the top row, the first scene is visualized and in the bottom row the second scene is visualized. The left two columns are from the ablated model, and the right two columns are from the base model. While the rendered RGB quality is similar for the base model and ablation, the attention maps are much more consistent in the base model. While in the base model many of the slots bind to specific objects and is consistent between scenes, in the ablated model many of the slots are not consistent across the two scenes and many slots are shared between the actual objects and the background.

Training on one scene

The model performs best when trained on multiple scenes, and struggles when it is trained on one scene. This can be attributed to the inability to make conclusions about object boundaries when given only a static scene of objects, as it is ambiguous what parts are separate or joined. This effect can be observed even when the number of scenes is artificially expanded with data augmentations. We trained the model in the teatime scene from the LERF datasets [7], creating 4 copies of the scene by applying z-axis rotations onto the base scene in 3 different angles. Outputs can be seen in figure 4.6. As can be seen, the attention map is incoherent. In fact, we observed that the attention map was worse when trained on multiple rotations of the scene than when trained with only scene.

4.5 Additional Visualizations

Additional visualizations of rendered videos can be found in this google drive: link

Chapter 5

Conclusion

Unsupervised object-centric methods have found a unique niche in novel view synthesis settings thanks to their ability to create visually informative representations, allowing for human interpretation of the activations of a neural network. This thesis expanded upon these works by proposing a novel method of training a Neural Radiance Field for a multi-scene setting with reusable object representations. Our method successfully discovers simple objects in scenes, creating visually discriminate representations for them. However, our method does struggle in harder scenes where the objects are not rigid and has textured surfaces not limited to single colors.

Object-centric learning is a natural step in finding relationships between visual information, as can be seen in the experimental results of this paper. Making associations across scenes for shared object information can lead to rich feature representations that are potentially reusable. In future work, such representations can potentially be used for downstream tasks that pertain to the compositionality of a scene, such as in scene editing, object removal, or question answering.

Bibliography

[1] Jonathan T. Barron et al. *Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields*. 2022. arXiv: `2111.12077 [cs.CV]`.

[2] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: `1406.1078 [cs.CL]`.

[3] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. *On the Binding Problem in Artificial Neural Networks*. 2020. arXiv: `2012.05208 [cs.NE]`.

[4] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: `10.1109/CVPR.2016.90`.

[5] Justin Johnson et al. *CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning*. 2016. arXiv: `1612.06890 [cs.CV]`.

[6] James Kajiya and Brian von herzen. "Ray Tracing Volume Densities". In: *ACM SIGGRAPH Computer Graphics* 18 (July 1984), pp. 165–174. DOI: `10.1145/964965.808594`.

[7] Justin Kerr et al. *LERF: Language Embedded Radiance Fields*. 2023. arXiv: `2303.09553 [cs.CV]`.

[8] Chung Min Kim et al. *GARField: Group Anything with Radiance Fields*. 2024. arXiv: `2401.09419 [cs.CV]`.

[9] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: `1412.6980 [cs.LG]`.

[10] Thomas Kipf et al. *Conditional Object-Centric Learning from Video*. 2022. arXiv: `2111.12594 [cs.CV]`.

[11] Francesco Locatello et al. *Object-Centric Learning with Slot Attention*. 2020. arXiv: `2006.15055 [cs.LG]`.

[12] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. *Effective Approaches to Attention-based Neural Machine Translation*. 2015. arXiv: `1508.04025 [cs.CL]`.

[13] Ben Mildenhall et al. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. arXiv: `2003.08934 [cs.CV]`.

[14] Thomas Müller et al. "Instant neural graphics primitives with a multiresolution hash encoding". In: *ACM Transactions on Graphics* 41.4 (July 2022), pp. 1–15. ISSN: 1557-7368. DOI: `10.1145/3528223.3530127`. URL: `http://dx.doi.org/10.1145/3528223.3530127`.

[15] Di Qi, Tong Yang, and Xiangyu Zhang. *Slot-guided Volumetric Object Radiance Fields.* 2024. arXiv: `2401.02241 [cs.CV]`.

[16] Mehdi S. M. Sajjadi et al. *Object Scene Representation Transformer.* 2022. arXiv: `2206.06922 [cs.CV]`.

[17] Mehdi S. M. Sajjadi et al. *Scene Representation Transformer: Geometry-Free Novel View Synthesis Through Set-Latent Scene Representations.* 2022. arXiv: `2111.13152 [cs.CV]`.

[18] Gautam Singh, Fei Deng, and Sungjin Ahn. *Illiterate DALL-E Learns to Compose.* 2022. arXiv: `2110.11405 [cs.CV]`.

[19] Cameron Smith et al. *Unsupervised Discovery and Composition of Object Light Fields.* 2023. arXiv: `2205.03923 [cs.CV]`.

[20] Karl Stelzner, Kristian Kersting, and Adam R. Kosiorek. *Decomposing 3D Scenes into Objects via Unsupervised Volume Segmentation.* 2021. arXiv: `2104.01148 [cs.CV]`.

[21] Matthew Tancik et al. "Nerfstudio: A Modular Framework for Neural Radiance Field Development". In: *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings.* SIGGRAPH '23. ACM, July 2023. DOI: `10.1145/3588432.3591516`. URL: `http://dx.doi.org/10.1145/3588432.3591516`.

[22] Ashish Vaswani et al. *Attention Is All You Need.* 2023. arXiv: `1706.03762 [cs.CL]`.

[23] Ziyi Wu et al. *SlotFormer: Unsupervised Visual Dynamics Simulation with Object-Centric Models.* 2023. arXiv: `2210.05861 [cs.CV]`.

[24] Hong-Xing Yu, Leonidas J. Guibas, and Jiajun Wu. *Unsupervised Discovery of Object Radiance Fields.* 2022. arXiv: `2107.07905 [cs.CV]`.