Towards a More Object-Centric Dynamic Scene Reconstruction Model

by
Eliot Laidlaw

A Thesis submitted in partial fulfillment of the requirements for Honors
in the Department of Computer Science at Brown University

Providence, Rhode Island
October 2023

This thesis by Eliot Laidlaw is accepted in its present form by
the Department of Computer Science as satisfying the research requirement
for the awardment of Honors.

Date _____ _____

James Tompkin, Reader

Date _____ _____

Srinath Sridhar, Reader

Acknowledgements

This work was completed in collaboration with Yiqing Liang, Alex Meyerowitz, Srinath Sridhar, and James Tompkin. For chapter 3 and chapter 4, I was the primary investigator. The work in chapter 5 was done primarily by Yiqing, and much of the chapter is written by James and Yiqing as part of a paper submission. Thank you to Yiqing, Alex, and Srinath for always being ready to have a thoughtful discussion and for being a pleasure to work with. Previous work on TöRF [2] was completed in collaboration with Ben Attal, Aaron Gokaslan, Christian Richardt, Changil Kim, Matthew O'Toole and James, who I thank for sharing their expansive knowledge in the field and for taking me under their collective wing early in my time as an undergraduate researcher. Finally, my deepest thanks to James for being an amazing mentor over the last four years, guiding me into research and always putting his students first.

Contents

Chapter 1

Introduction

Reconstructing 3D dynamic scenes from one or more videos is a difficult problem in computer vision, with motivation in content creation, robotics, and other downstream tasks. Progress in novel view synthesis and scene reconstruction has accelerated recently with neural field-based representations mixing with more classical ones to improve reconstruction quality. In particular, the many variations of Neural Radiance Field [23] have demonstrated remarkable rendering quality. Dynamic scenes pose a more challenging problem than static ones, and neural field models have been developed for this setting as well. These methods, however, still lack the ability to represent long sequences and often take hours or days to optimize. Existing dynamic models also do not support rendering at interactive framerates, which is required for applications like virtual reality. In our previous work [2], optimization and rendering speed were both orders of magnitude slower than interactive rates. Additionally, we found that longer, more complex scenes or those with small camera baselines were difficult for the model to reconstruct.

Given these current limitations, it is our belief that at the core of representing a dynamic scene efficiently is an understanding of the objects that make it up. We propose three approaches that integrate object-level information into a hybrid scene representation. Our first approach uses a point cloud-based representation and develops the notion of object clusters of points that move together over time. Instead of representing scene motion as the output of a neural network, we use a rigid transformation for each cluster at each timestep. This makes our representation compact and easy to convert to a format that can be quickly rendered for interaction. Our second approach builds on the first but uses a NeRF instead of a point cloud as the underlying representation. Cluster assignment is still based on pretrained features, but instead of baking these clusters into the point cloud, they are parameterized by a network that is optimized simultaneously with geometry and appearance. The network outputs can still be translated into an explicit representation if desired, but the NeRF-first approach allows us to capitalize on the benefits of NeRF and similar methods. For each of these two approaches we generate preliminary results on a sample dynamic scene dataset. We make limiting assumptions about the dynamic scenes that we can represent, but these types of scenes do exist in the real world and further development would hopefully alleviate the limitations.

The final approach we propose focuses on discerning foreground objects in dynamic scenes. We augment Neural Scene Flow Fields (NSFF) [19] with semantic and attention feature heads that allow us to propagate object level information into the dynamic neural field. These semantic and attention features come from DINO-ViT [4], a pretrained transformer model, and are improved through semantic feature pyramids. After

volume integration, we perform saliency aware clustering to disentangle the important objects in the scene from the background. The knowledge of where salient objects are in space and time allows us to more easily develop a representation like the two previously described. The three approaches together provide a promising direction for representing dynamic scenes with a more object-centric model.

Chapter 2

Related Work

Scene reconstruction and novel view synthesis (NVS) are long-standing problems in computer vision for both static and dynamic scenes. Decomposing scenes into the objects that make them up has also been an area of study for researchers. The introduction of Neural Radiance Fields (NeRFs) has had an outsized impact on the field, and we focus on the work accomplished in this area.

2.1 Dynamic NeRFs

Since the introduction of NeRFs, much progress has been made on adapting the basic representation to work for dynamic scenes. Many approaches accomplish this by optimizing the NeRF at a static point in time and simultaneously learning a mapping for 3D points from other timesteps into that canonical "frame." These deformation-based methods include D-NeRF [29], NR-NeRF [43], and Nerfies [26]. VideoNeRF [50] optimizes a latent code per-frame to capture time varying information that is passed along with positional information to the network. NSFF [19] and NeRFlow [7] use scene flow to regularize a dynamic NeRF network. Raising the coordinate space to a higher dimension has been shown to improve reconstruction quality by HyperNeRF [27]. Additionally, in our previous work, we showed that time-of-flight information can be used to improve geometric reconstruction in scenes with little camera motion [2].

Another whole direction of research is the reconstruction of dynamic humans using NeRF-based methods. These methods use prior information about human bodies and their motion constraints to aid in reconstruction [30] [25] [47] [39] [28] [11].

In contrast to these methods, we focus on learning an object-centric representation that encodes scene motion as explicit object transforms. Our representation is easily converted to other representations that can be quickly rendered, which is not the case for prior work.

2.2 Efficient NeRFs

Significant progress has been made in making NeRF optimization and rendering more efficient. One promising direction is baking parts of the NeRF into representations that can be more quickly rendered [14] [6]. FastNeRF [12]

caches the radiance output in a manner that can be efficiently queried at render time. Other approaches attempt to limit the number of network queries altogether. [36] represents the scene as a light field so that rays can be rendered with a single network evaluation. NSVF [21] creates a sparse voxel grid that eliminates the need for network queries in empty space. Finally, KiloNeRF [33] optimizes thousands of tiny MLPs such that each MLP can represent a small region of the volume with a much smaller number of parameters and computation time.

The most effective method for improving the efficiency of implicit scene representations has been to use a hybrid approach. Several methods store features on a voxel grid that are then used to speed up the final rendering of the scene. DirectVoxGo [40] directly optimizes a voxel grid representation, while Plenoxels [34] and PlenOctrees [55] encode appearances using spherical basis functions within the voxel grid. A multiplane image can also be used as an underlying data structure in place of a voxel grid [48]. While these works greatly improve computation time, they come at an increased memory cost, which other methods have worked to alleviate. Instant NGP [24] uses a multiresolution hash grid to reduce memory cost at higher feature grid resolutions. [41] uses a dictionary based approach for multiresolution stored features. Tensor factorization of the voxel grid is also a viable direction for reducing memory cost [5]. TiNeuVox [9] works to incorporate the benefits of a hybrid approach into the dynamic NeRF setting.

Our method builds on the hybrid approach, but for dynamic scenes which have had minimal exploration in this area. We take it a step further than TiNeuVox by using a non-neural representation for scene motion in addition to the underlying geometry and appearance.

2.3 NeRFs with semantics and object-level information

NeRFs [51] have also spurred new scene decomposition research through volumes. ObSuRF [38] and uORF [56] are unsupervised slot attention works that bind a latent code to each object. Unsupervised decomposition is also possible on light fields [37]. For dynamic scenes, works like NeuralDiff [45] and D^2NeRF [49] focus on foreground separation, where foreground is defined to contain moving objects. Other works like N3F [44] and occlusions-4d [15] also decompose foregrounds into individual objects. N3F requires user input to specify which object to segment, and occlusions-4d takes RGB point clouds as input.

Other methods have integrated semantic information directly into the neural field. iLabel [57] adds a semantic head to propagate user-provided segmentations in the volume, while DFF [16] integrates pretrained features into the volume. PNF [18] and Panoptic-NeRF [10] attempt panoptic segmentation within neural fields, and Object-NeRF integrates instance segmentation masks into the field during optimization [52].

Another class of works reconstructs human bodies using body part-level information. ANR [30], NARF [25], HumanNeRF [47], and [39] use an articulation model to reconstruct people in motion. Neural body [28] and [11] use learnable codes for novel view synthesis of human bodies and faces, respectively. Vid2Actor [46] creates an implicit animatable person model from video.

Our work builds on the methods that integrate object-level information into the neural field, but does so in the dynamic setting. We do so by adding semantic features directly to the field, as introduced by other works. We also generalize to settings beyond just human bodies.

Chapter 3

Point Cloud-Based Cluster Model

3.1 Motivation

To represent longer videos with a method that allows for fast rendering and "online" optimization, we first devise a method involving a discrete clustering of the scene. To make the space-time novel view synthesis problem easier, we make critical assumptions that objects in the scene we reconstruct are non-deformable, move by rigid transformations, and do not change in appearance over time. These are limiting assumptions, but they give us a starting point to build from.

Given the premise of a scene comprised of rigid objects, we simplify the reconstruction task to three subtasks that give us the ability to represent the scene at any timestep:

1. Represent the scene statically at a certain "canonical" timestep.
2. Segment the scene at that canonical timestep into clusters at least approximately corresponding to the objects in the scene.
3. Determine the rigid transformations for each object from the canonical timestep to each other timestep.

With this set of information, we can morph our canonical frame representation into a representation of the scene at any timestep. This representation has other benefits (and drawbacks) that we will discuss.

3.2 Method

3.2.1 Static canonical frame representation

The first task we tackle is representing the scene statically in the canonical timestep. To keep things simple, the canonical timestep will be the first frame of our video sequence unless otherwise noted.

The first canonical frame representation we use is a NeRF, $F^* : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$ which maps a 3D position $\mathbf{x} = (x, y, z)$ and viewing direction $\mathbf{d} = (\theta, \phi)$ to a color $\mathbf{c} = (r, g, b)$ and density σ. To render an image using this representation we query the network at several points along each ray and accumulate the color values based

on the scene density at each point. The second canonical frame representation we use is a point cloud $\mathcal{P}$ where each point consists of a 3D position (x, y, z) and a color value with transparency (r, g, b, a). To render an image using this representation, we use standard rasterization techniques.

3.2.2 Scene segmentation

Next, we devise a representation for our segmentation of the scene into clusters that move together over time. We assume here that there are c clusters in the scene.

The first representation we use is voxel-based. For each voxel in a predefined volume, we define a cluster assignment vector $\mathbf{a} = (a_1, \ldots, a_c)$, where $a_j \in [0, 1]$ is the weight with which the voxel is assigned to cluster j. The second representation adds cluster assignments directly to our point cloud $\mathcal{P}$. In addition to 3D position and RGBA color, each point now contains a cluster assignment vector $\mathbf{a}$ as defined above.

In practice, these representations are similar or even the same. If the points in the point cloud are chosen to lie on a 3D grid with the same number of subdivisions as the voxel grid, the cluster assignment vectors will be the same for each corresponding point and voxel.

3.2.3 Cluster transformations

The final piece of our dynamic scene representation is a parameterization of motion over time. For each timestep i and cluster j, we define a rigid transformation $T_j^i : \mathbf{x}_j^* \to \mathbf{x}_j^i$ that maps a point from cluster j in the canonical frame to its position at timestep i. Under the hood these transformations are parameterized by translation vectors $\mathbf{t}_j^i \in \mathbb{R}^3$ and rotation vectors $\mathbf{r}_j^i \in \mathrm{SO}(3)$.

3.2.4 Rendering the scene

With these three components in place, we are able to render the scene from any viewpoint and timestep. For rendering we will assume that we are using the point cloud representation for the canonical frame and segmentation.

The first step in rendering at timestep i is to transform the points in our canonical frame point cloud $\mathcal{P}$ to their position at timestep i. For a given point p with position $\mathbf{x}$ and cluster assignment vector $\mathbf{a} = (a_1, \ldots, a_c)$, we compute its position at timestep i as the mean of $\mathbf{x}$ transformed as if it were part of each cluster, weighted by $\mathbf{a}$:

$$\mathbf{x}^i = \frac{1}{c} \sum_{j=1}^{c} a_j T_j^i(\mathbf{x}) \tag{3.1}$$

The resulting point cloud after mapping each point's canonical frame position $\mathbf{x}^*$ to $\mathbf{x}^i$ is then easily rendered using rasterization techniques. In practice we use PyTorch3D's point cloud renderer[32].

3.2.5 Building the representation

Now that we've defined the dynamic scene representation and how to render from it, we provide a process for building it using images over time. The first step is to learn the canonical scene representation. For this, we optimize the canonical NeRF F^* using images of the scene from the canonical frame. Because the representation that

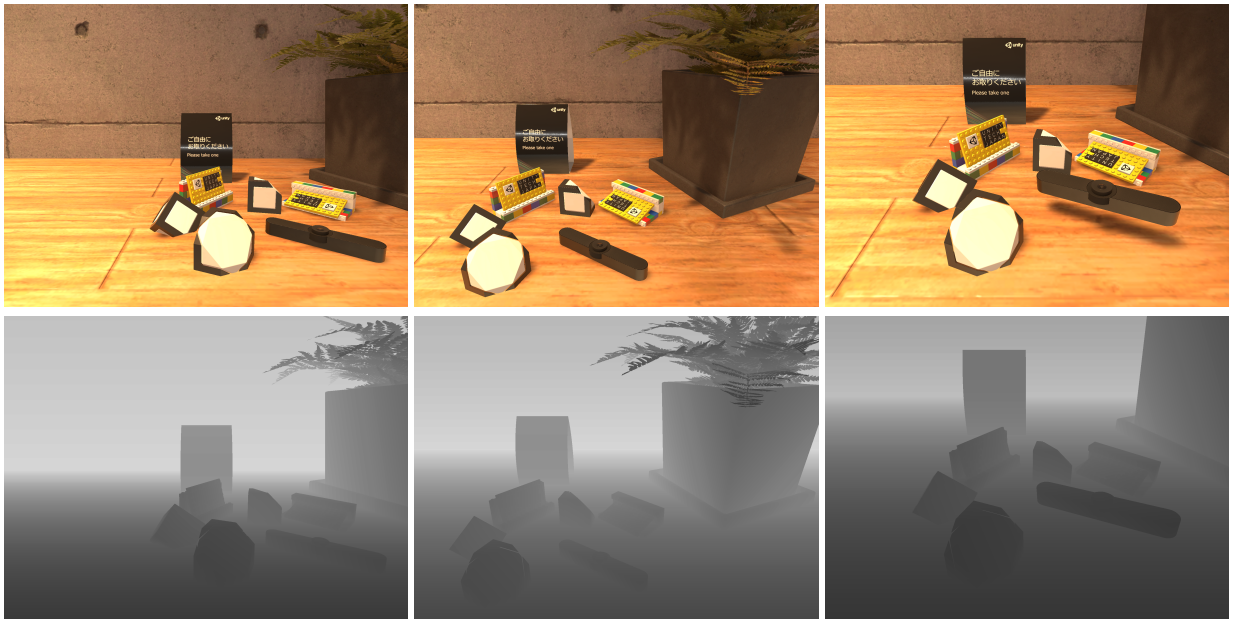

Figure 3.1: **Sample images and depth maps from our Unity dataset.** Images of the scene from the canonical frame (left, center) and frame 6 (right).

we ultimately render from is the point cloud $\mathcal{P}$, we then sample the optimized NeRF along a 3D grid, with the the point p at position $\mathbf{x}$'s (r, g, b, a) values corresponding to the color $\mathbf{c}$ and density σ obtained from evaluating $F^*(\mathbf{x})$.

To get our cluster assignment vector $\mathbf{a}$ for each point, we first add a feature reconstruction head to our NeRF model. For each input image, we compute a semantic feature map using the pretrained self-supervised transformer model DINO-ViT [4]. We then add an additional output to our NeRF to reconstruct these semantic features: $F^* : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma, \mathbf{f})$. At our point cloud sampling step we can then append a feature vector $\mathbf{f}$ to each point p.

To turn these features into cluster assignment vectors, we perform k-means clustering on a weighted combination of 3D position and feature vectors. We use this clustering to define $\mathbf{a}$ for each point as a one-hot vector assigning the point fully to the cluster centroid it lies closest to.

The final piece of our representation to build from data is the rigid cluster transformations. To this point we have only considered images (and corresponding semantic feature maps) from the canonical frame. Now we consider images from subsequent timesteps, and assume that we have camera parameters and at least rough depth maps for each frame. For each pair of adjacent frames in the video sequence, we estimate optical flow using RAFT [42].

To compute the transformation from a frame i where we have the transformations T_j^i to frame $i+1$ where we do not, we use the optical flow, camera parameters, and depths from the adjacent frames. We backproject the pixels from each frame to 3D points using the camera parameters and depth maps and then compute the corresponding 3D scene flow from timestep i to timestep $i+1$ using the estimated optical flow. These back-projected points can then be assigned to clusters using the centroids from our k-means clustering (transformed by the known T_j^i). For each cluster j, we fit a transform $T_j^{i \rightarrow i+1}$ that best represents the scene flow of the backprojected points assigned to cluster j from timstep i to timestep $i+1$. The transform for cluster j from

Figure 3.2: **Sample results from the base representation** Ground truth rendering of the scene from frame 11 (left) and the rendering from the point cloud (center). There are significant errors around the black object that has moved off of the desk. The cluster for the object (right) includes a large piece of the desk, which then moves based on the estimated object transforms.

the canonical frame to timestep $i+1$, T_j^{i+1}, is then simply $T_j^{i \to i+1} \circ T_j^i$.

Determining the cluster transforms for timestep $i+1$ relies on knowing the transforms for timestamp i, so we build the transforms iteratively, starting with the canonical frame for which we constrain the transforms to be the identity. Because the optical flow estimates are not perfect, if the estimated transform for a cluster j from frame i to $i+1$ is small enough, we assume that the cluster has not in fact moved and set $T_j^{i+1} = T_j^i$.

These transforms are based on estimated optical flow and error can be accumulated over the video sequence because of the iterative computation process. This means that the transforms may require finetuning. We optimize the transforms using gradient descent, minimizing the RGB reconstruction loss when rendering frames of the video sequence using our scene representation and PyTorch3D's differentiable point cloud renderer.

3.3 Experiments

We perform a number of exploratory experiments to investigate the capabilities of our dynamic scene representation and possible additions to the model.

3.3.1 Unity dataset

To evaluate our representation, we first created a dataset of images and depth maps using Unity, shown in Figure 3.1. The sequence contains 24 images of the scene in the canonical frame followed by 1 image of the scene for each of 12 subsequent timesteps in which one of the objects in the scene moves. We also render depth maps and save camera parameters for each of the 36 images. The dataset capture script can be used with any Unity scene.

3.3.2 Model investigation results

We perform a number of investigations to attempt to improve performance of the base representation as previously defined. We begin with results using the base representation, point out limitations, and describe our attempts to ameliorate the limitations.

Results from the base model. Example results using the method for building the representation defined

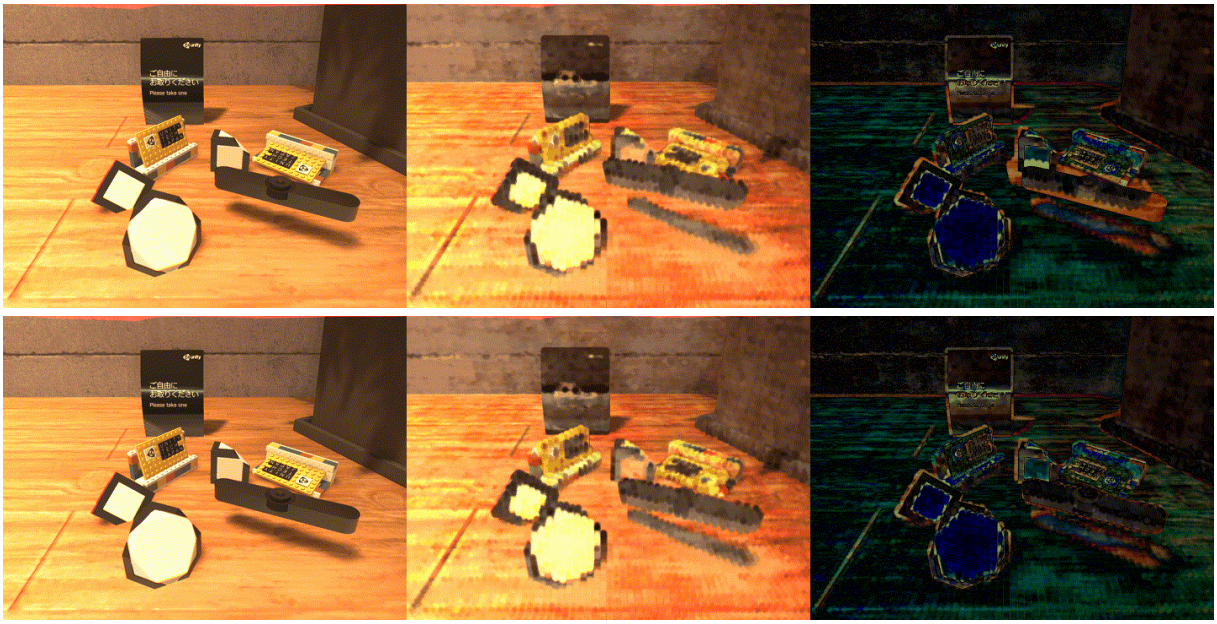

Figure 3.3: **Cluster transform optimization using PyTorch3D** Ground truth rendering (left), our base model rendering (center), and error (right) before (top) and after (bottom) optimizing cluster transforms using PyTorch3D's differentiable point cloud renderer. The black moving object is better aligned to the ground truth after optimization.

previously can be found in Figure 3.2. The biggest limitation of the base method is that the k-means clustering is inherently naive and does not accurately capture object boundaries. This is especially apparent when objects are in contact with each other in the canonical frame, like the moving black object and desk in our Unity dataset. The DINO-ViT supervised feature vectors assigned to each point are by definition unconstrained in regions of the volume that are not directly projected into any of the input images. This means that the interior regions of the desk and black object have unconstrained feature vectors which results in the poor clustering performance we see in Figure 3.2, right.

Figure 3.3 shows the benefits of using gradient descent to optimize the cluster transforms that are initially estimated from optical flow. For these results, the moving cluster is manually adjusted to not include points from the desk.

Results after culling erroneous cluster points using projection. We now attempt to remove erroneous points from the moving cluster to improve reconstruction performance. The key insight is that when reconstructing an input image, the points that are erroneously included with a moving cluster will project into a region of the image where the optical flow is not consistent with the motion of the cluster.

More specifically, for an input image and corresponding optical flow map from a given frame i, we first render a corresponding image using our representation. We also render an image mapping each pixel to the point in $\mathcal{P}$ that it was rendered from. Given a cluster j that is moving at time i, we find all pixels that were rendered using points from the cluster. For each of the pixels that have magnitude of optical flow below a certain threshold, we remove the points from j and merge them into the next closest cluster. By repeating this process, we remove all points that are visible that do not project into a region of non-negligible optical flow. Figure 3.4, bottom

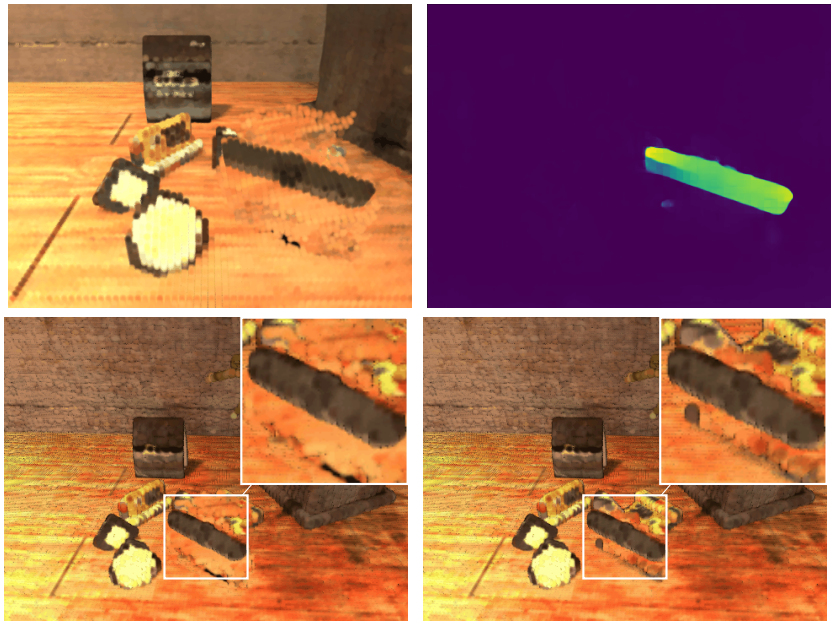

Figure 3.4: **Optical flow offers an opportunity to remove erroneous points from moving clusters** In the rendering from our base model, the moving cluster includes a number of erroneous points (top left). The magnitude of optical flow (top right) from this frame gives us a mask to cull away the erroneous points from the cluster. After culling away the points that project into regions of the flow that are inconsistent with the motion of the cluster, we see improved reconstruction (bottom right).

right, shows the cluster for the black object before and after this cluster culling.

Results after culling erroneous cluster points using fitted planes. As can be seen in Figure 3.4, even after culling points using projection and optical flow, there are still erroneous points in the cluster. The next proposal is to fit planes to the boundaries between objects and use the planes to remove points from the clusters. For example, in our Unity dataset, the black object initially lies on the desk. The boundary between the desk and object is the plane defined by the surface of the desk. From the canonical frame NeRF, we have a good model of the scene in the canonical frame. Points that are part of the contact area between two objects in the canonical frame will not be visible until one of the objects move. Therefore, in the canonical frame, they are "interior" points of the point cloud, meaning all adjacent points are not transparent. If "interior" points of the point cloud are revealed in subsequent frames, they lay on an object boundary. We accumulate these points over several frames and fit planes using RANSAC. We then use these planes to adjust the clustering.

Direct optimization of cluster assignment vectors. We also tried using gradient descent to optimize each point's cluster assignment vector directly using image reconstruction loss. This was largely unsuccessful, as the reconstruction signal was not strong or directed enought to force erroneous points into adjacent clusters.

3.4 Discussion

The explicit clustering dynamic scene representation shows some promise for representing scenes with rigid objects, but has clustering limitations that bring its viability into question. Setting aside the intentionally

imposed assumptions, the reconstruction quality on an applicable scene, our Unity dataset, is poor because of the underlying point cloud representation and the lackluster clustering performance. While we can improve the initial clustering with our projection culling and plane culling approaches, they are more of hacks than real solutions to the underlying problem. During NeRF optimization, gradient descent by definition continually improves reconstruction quality. Here, however, there is no feedback loop to improve clustering other than our culling methods. The clusters we have are taken mostly "as is" and therefore limit model performance. To combat these issues, we turn instead to a model that uses NeRF as its core representation and clusters objects using a neural field.

Chapter 4

ClusterNeRF

4.1 Motivation

Neural fields have shown a remarkable ability to learn complex functions. Rather than extract information from an initial NeRF representation into a point cloud-based one one, we now attempt to morph a standard NeRF-based representation into one that resembles our explicit one. One problem that plagues most neural fields-based methods is optimization time. While Instant NGP [24] and other methods have shown significant improvements in this area, dynamic NeRFs are still relatively slow to optimize. We investigate a neural field model with this in mind. Even if optimization is slower in this setting, rendering and incorporating information from subsequent timesteps could theoretically still happen at interactive speeds.

4.2 Method

4.2.1 ClusterNeRF Representation

Again, we wish to represent a 3D scene at every frame $i \in \{1, \ldots, t\}$, where t is the number of observed frames in time. The first piece of our representation is still a NeRF that represents the scene in the canonical frame, i^*. The modified NeRF is a function $F^* : (\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$ which maps a 3D position $\mathbf{x} = (x, y, z)$ and viewing direction $\mathbf{d} = (\theta, \phi)$ to a color $\mathbf{c} = (r, g, b)$ and density σ. The * superscript denotes that a function (F^* in this example) represents the scene in the canonical frame.

We also use the same notion of clusters–sets of points in the canonical frame that move together over time. Given a scene with c clusters, we now define an assignment function $A^* : \mathbf{x} \rightarrow \mathbf{a}$ that maps a 3D position $\mathbf{x} = (x, y, z)$ to a cluster assignment vector $\mathbf{a} = (a_1, \ldots, a_c)$ where $a_j \in [0, 1]$ is the weight with which the point $\mathbf{x}$ is assigned to cluster j.

Finally, we use the same transforms as our explicit model, with $T_j^i : \mathbf{x}_j^i \rightarrow \mathbf{x}_j^*$ being the function that transforms a 3D position $\mathbf{x}_j^i$ in frame i to its corresponding 3D position in the canonical frame if it were part of cluster j. T_j^*, the transforms for clusters in the canonical frame, are constrained to be the identity.

Rendering an image with this formulation is similar to rendering a NeRF. Given a camera pose and a frame i,

points are sampled and accumulated along rays emanating from the camera center. For a NeRF, at each sample point $\mathbf{x}$ and corresponding viewing direction $\mathbf{d}$, the color and density are determined by a single evaluation of F^*. In our method, however, we must account for the possibility that at frame i, any cluster j may have moved to occupy space containing the sample point $\mathbf{x}$. Therefore, for each cluster, we query the network at $\mathbf{x}$'s position in the canonical frame if it were part of cluster j, with this position being represented by $T_j^i(\mathbf{x})$. We take the weighted average of F^* evaluated at each of these c positions based on the corresponding evaluation of A^*:

$$F^i(\mathbf{x}, \mathbf{d}) = \frac{\sum_{j=1}^{c} A^*(T_j^i(\mathbf{x}))_j F^*(T_j^i(\mathbf{x}), T_j^i(\mathbf{d})))}{\sum_{j=1}^{c} A^*(T_j^i(\mathbf{x}))_j} \tag{4.1}$$

$A^*(\cdots)_j$ denotes the j^{th} value in the resulting $\mathbf{a}$ vector when evaluating A^*. These $F^i(\mathbf{x})$ values are then accumulated along each ray using the same method as standard NeRF to render images from any frame.

4.2.2 Building the representation

Unlike NeRF, optimizing our representation cannot be achieved using just reconstruction loss over a set of RGB images. Therefore, we propose the following method for optimizing F^*, A^*, and the parameterized values of T_j^i for each cluster and frame.

Similar to building our explicit representation, we take as input a set of images of the scene with corresponding depth maps, and estimate optical flow between adjacent frames. We then use existing methods to estimate a segmentation of each image into the objects we would like to be represented by different clusters. This can be done using classical methods similar to the ones we use in our explicit model or deep learning approaches. For our initial experiments on basic data, we simply use k-means clustering on the RGB images of the scene. We could also use the same method we use to estimate T_j^i in our explicit model for our NeRF-based model, but for initial experiments we just initialize all T_j^i to be the identity.

Learning the representation is more tractible now with supervision for A^*. We optimize the model using reconstruction loss over the RGB images and clustering masks. Each T_j^i function is parameterized by a cluster center, a rotation, and a translation vector, which are directly optimized using gradient descent. The canonical NeRF F^* and assignment function A^* are hybrid models based on the approach of Instant NGP.

To avoid baking error from the estimated cluster masks, we decay the loss from reconstructing the masks over time.

4.3 Experiments

4.3.1 D-NeRF dataset

For our initial experiments, we use a simple bouncing ball example provided by the authors of D-NeRF [29]. The sequence consists of three balls that bounce on a surface and is captured from several viewing angles over 150 input images, a few of which are shown in Figure 4.1.

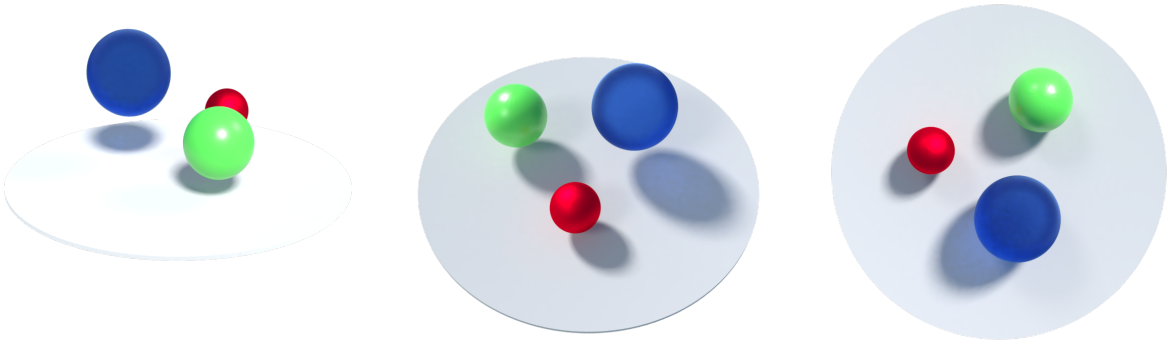

Figure 4.1: **Sample images from the D-NeRF bouncing balls dataset.** The figure contains 150 images of the scene as the balls bounce on the white surface.

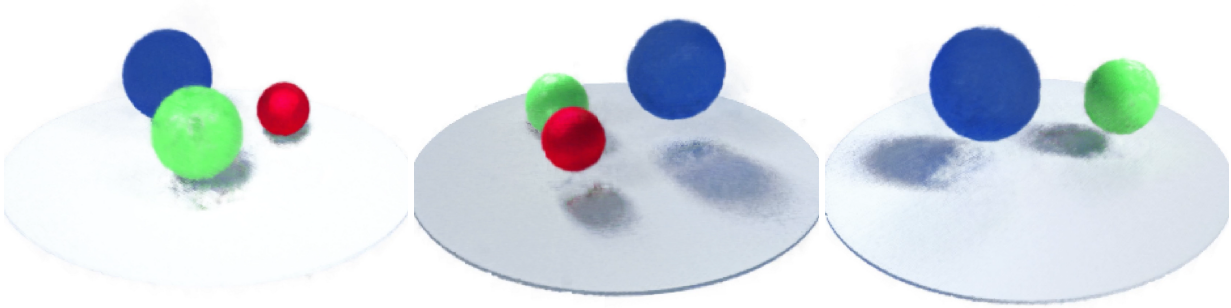

Figure 4.2: **Sample results from our implicit model.** These images are rendered from novel viewpoints at three different timesteps.

4.3.2 Results

We optimize the model as described and show key results that suggest the assignment function is performing relatively well. Figure 4.2 shows renderings from our model of the scene at novel views and different times. The results are slightly fuzzy and there are errors in the shadows, but the model's renderings are fairly accurate. Importantly, the motion of the balls is correct in the reconstruction.

To investigate the quality of the assignment function A^*, we produce two additional sets of results. In the first set, shown in Figure 4.3, we use A^* to render the scene without one of the balls. The blue ball is removed from the scene, but there are some ghosting artifacts. The shadow cast by the blue ball is not part of its cluster, which leads to an inconsistent scene when deleting the ball.

Results from our second investigation are shown in Figure 4.4. Here, we render the scene with just one of the clusters moving. To do this, we fix all other clusters' transforms to be T_j^* while cycling through the transforms for the cluster containing the green ball. We see similar results to the removal experiment. There is some ghosting where the green ball was in the canonical frame, and the shadow does not move to reflect the motion of the ball. The model does successfully reconstruct the deformation of the ball when it bounces. We hypothesize that the model assigns parts of the ball "softly" to multiple clusters, meaning that the assignment vector $\mathbf{a}$

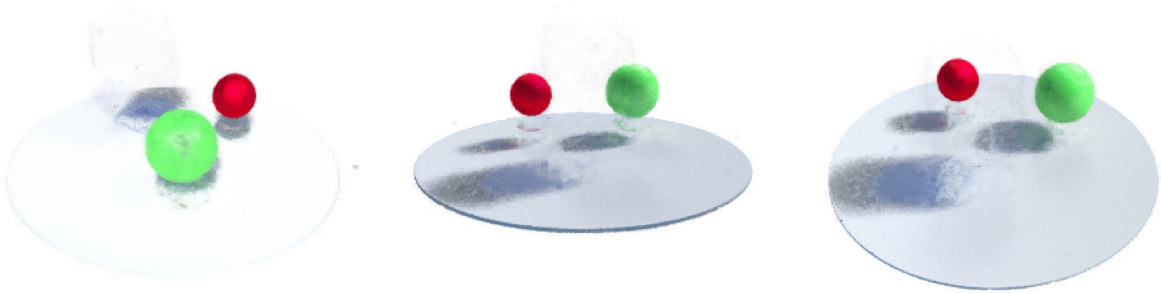

Figure 4.3: **Implicit model results without a cluster.** Points assigned to the cluster for the blue ball are rendered as transparent.

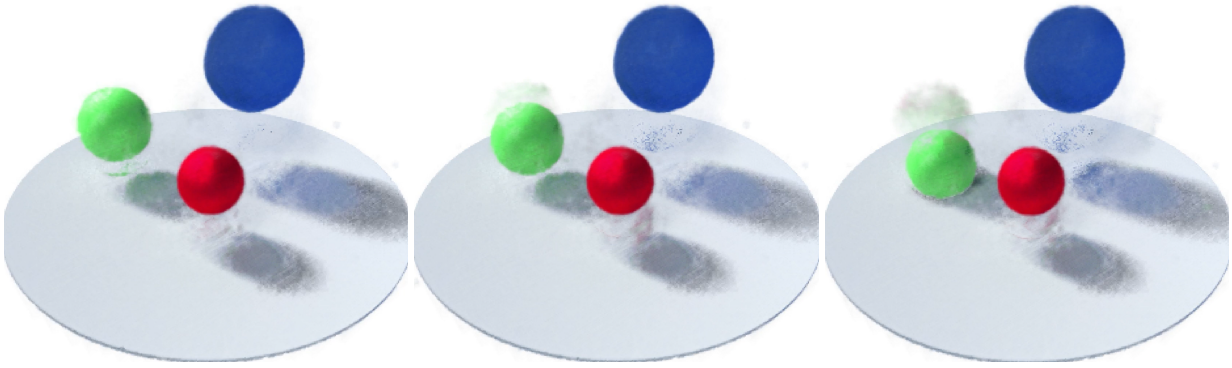

Figure 4.4: **Results from our implicit model moving one cluster individually.** The transforms for the red and blue ball clusters are held constant while the green ball's transform is varied over time. These images are rendered from novel viewpoints.

would have multiple non-zero components as opposed to being a "one-hot" vector. This could theoretically allow our model to represent more complex non-rigid objects.

4.4 Discussion

Our ClusterNeRF model produces generally higher quality results than the explicit one, albeit with a few caveats. First, the dataset is different in the two sets of experiments. While the increase in quality from using a NeRF-based method vs. a point cloud is expected and would likely carry over to different datasets, it is possible that our model would fail to estimate the motion or clustering of the scene.

For the D-NeRF dataset we use for experiments, estimating cluster masks is easy, given that the objects in the scene are monochromatic and distinctly colored. Initial experiments suggest that cluster masks can be estimated for other scenes like our Unity dataset, but we have not conducted enough testing to suggest this is broadly true. The model does have some ability to overcome error in the masks by using RGB reconstruction loss as a stronger signal.

Overall, ClusterNeRF seems promising for representing dynamic scenes with rigid objects. It is also easy to

update with information from new frames. By simply estimating a new transform for each cluster, the scene could be rendered from the new timestep without any gradient descent updates. The representation can also be easily converted into an explicit model that is easy to render at interactive framerates. For example, each cluster could be converted to a separate mesh that would be rendered using the transforms and standard rasterization techniques.

Chapter 5

Semantic Attention Flow Fields

5.1 Motivation

Thus far, we have designed methods that rely heavily on a clustering of the scene into objects that move together, but we have yet to focus on dynamic scene decomposition as the core problem. We wish to discern the salient objects in a dynamic scene from the background. Existing methods make limiting assumptions about the input data, but we choose to work with casually captured monocular video to support the broadest spectrum of possible applications. This problem is inherently tied to the efficient scene reconstruction problem our explicit and implicit cluster-based methods have attempted to solve. A good dynamic scene decomposition would be invaluable in optimizing our ClusterNeRF representation.

5.2 Method

For a baseline dynamic scene reconstruction method, we begin with NSFF from Li et al. [19] (**??**). Its low-level scene flow frame-to-frame approach provides better reconstructions for real-world casual monocular videos than deformation-based methods [43, 27]. We modify the architecture to integrate higher-level semantic and attention (or saliency) features (section 5.2.1). After optimizing a SAFF for each scene, we can extract 3D object representations and perform saliency-aware clustering of the field (section 5.2.3).

Input Our method takes in a single RGB video over time i as an ordered set of images $I \in \mathcal{I}$ and camera poses. We use COLMAP to recover camera poses [35]. From all poses, we define an NDC-like space that bounds the scene, and a set of rays $\mathbf{r} \in \mathcal{R}$, one per image pixel with color $\hat{\mathbf{c}}^\dagger$. Here, $\hat{\cdot}$ denotes a 2D pixel value in contrast to a 3D field value, and $\cdot^\dagger$ denotes an input value in contrast to an estimated value.

From pretrained networks, we estimate single-frame monocular depth $\hat{d}^\dagger$ (MiDaSv2 [31]), optical flow $\hat{\mathbf{p}}_i^\dagger$ (RAFT [42]), and semantic features $\hat{\mathbf{s}}^\dagger$ and attention $\hat{\mathbf{a}}^\dagger$ (DINO-ViT [4]) after important preprocessing (section 5.2.2).

5.2.1 Semantic Attention Flow Fields

NSFF has only integrated low-level or *bottom-up* features into the field to represent a video. However, high-level or *top-down* features are also useful in defining objects and helping down-stream tasks like segmentation. For example, static/dynamic blend v estimates whether the volume appears to be occupied by some moving entity, but this is not the same as objectness.

As such, we extract 2D semantic features and attention (or saliency) values from a pretrained DINO-ViT network, then optimize the SAFF such that unknown 3D semantic and attention features over time can be projected to recreate their 2D complements. This helps us to ascribe semantic meaning to the volume and to identify objects.

To estimate semantic features $\mathbf{s}$ and attention $\mathbf{a}$ at 3D points in the volume at time i, we add two new heads to both the static $F_{\boldsymbol{\theta}}^{\text{st}}$ and the dynamic $F_{\boldsymbol{\theta}}^{\text{dy}}$ networks:

$$F_{\boldsymbol{\theta}}^{\text{st}} : (\mathbf{x}, \boldsymbol{\omega}) \quad \rightarrow (\ldots, \mathbf{s}^{\text{st}}, \mathbf{a}^{\text{st}}) \tag{5.1}$$

$$F_{\boldsymbol{\theta}}^{\text{dy}} : (\mathbf{x}, \boldsymbol{\omega}, i) \rightarrow (\ldots, \mathbf{s}_i^{\text{dy}}, \mathbf{a}_i^{\text{dy}}) \tag{5.2}$$

Here, $\ldots$ is used as a placeholder for the outputs from the original NSFF static and dynamic networks. i is the timestep we are querying. As semantic features have been demonstrated to be somewhat robust to view dependence [1], in our architectures both heads for $\mathbf{s}, \mathbf{a}$ are taken off the backbone before $\boldsymbol{\omega}$ is injected.

To render semantics and attention from the volume, we replace the color term while volume rendering with $\mathbf{s}, \mathbf{a}$. To encourage the flow to respect semantic features and attention over time, we penalize flow losses complementary to the flow loss on color reconstruction from NSFF. Finally, to supervise the two extra heads, we add respective losses on the reconstruction of the 2D semantic and attention features from projected 3D volume points ($\mathcal{R}$ is the set of rays we render):

$$\mathcal{L}_{\hat{\mathbf{s}}} = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r}_i \in \mathcal{R}} ||\hat{\mathbf{s}}_i(\mathbf{r}_i) - \hat{\mathbf{s}}_i^{\dagger}(\mathbf{r}_i)||_2^2 \tag{5.3}$$

$$\mathcal{L}_{\hat{\mathbf{a}}} = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r}_i \in \mathcal{R}} ||\hat{\mathbf{a}}_i(\mathbf{r}_i) - \hat{\mathbf{a}}_i^{\dagger}(\mathbf{r}_i)||_2^2 \tag{5.4}$$

Unlike depth and scene flow priors, these are not priors—there is no self-consistency for semantics to constrain their values. Thus, we *do not* decay their contribution. While decaying avoids disagreements between semantic and attention features and color-enforced scene geometry, it also leads to a loss of useful meaning.

These additional loss terms based on semantics and attention are added to all loss terms from NSFF.

5.2.2 Semantic Attention Pyramids

When thinking about scenes, we might argue that semantics from an ideal extractor should be scale invariant, as distant objects have the same class as close objects. We might also argue that saliency (or attention features) may not be scale invariant, as small details in a scene should only be salient when viewed close up. In practice, both extracted features vary across scale and have limited resolution, e.g., DINO-ViT [4] produces one output for each 8×8 patch. But, from this, we want semantic features and saliency for every RGB pixel that still respects scene boundaries.

Thus far, work on *static* scenes has ignored the input/feature resolution mismatch [17] as multi-view constraints provide improved localization within the volume. For monocular video, this approach has limitations

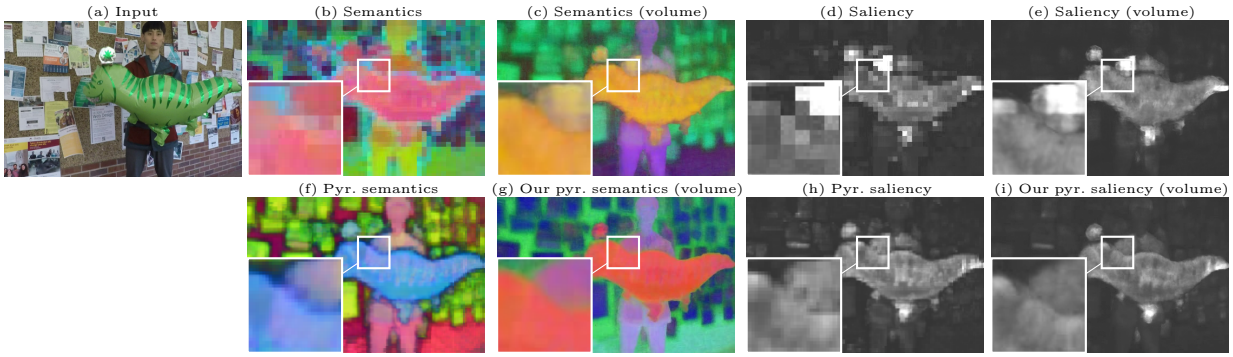

Figure 5.1: **Semantics and saliency improve by both volume integration and by our pyramid.** On *Balloon NBoard*, resolution is increased and unwanted saliency is softened. Semantics are visualized as most significant three PCA dimensions; specific colors are less meaningful.

[44]. Forming many constraints on dynamic objects requires long-term motion correspondence—a tricky task—and so we want to maximize the resolution of any input features where possible without changing their meaning.

One way may be through a pyramid of semantic and attention features that uses a sliding window approach at finer resolutions. Averaging features could increase detail around edges, but we must overcome the practical limit that these features are not stable across scales. This is especially important for saliency: unlike typical RGB pyramids that must preserve energy in an alias-free way [3], saliency changes significantly over scales and does not preserve energy.

Consider a feature pyramid $\mathcal{P}$ with loss weights per level:

$$\mathcal{L}_{\mathcal{P}\hat{\mathbf{s}}} = \sum_{i \in \mathcal{P}} \lambda^i_{\hat{\mathbf{s}}} \mathcal{L}^i_{\hat{\mathbf{s}}} \qquad \mathcal{L}_{\mathcal{P}\hat{\mathbf{a}}} = \sum_{i \in \mathcal{P}} \lambda^i_{\hat{\mathbf{a}}} \mathcal{L}^i_{\hat{\mathbf{a}}} \tag{5.5}$$

Naively encouraging scale-consistent semantics and whole-image saliency, e.g., $\lambda_{\hat{\mathbf{s}}} = \{1/3, 1/3, 1/3\}$ with $\lambda_{\hat{\mathbf{a}}} = \{1, 0, 0\}$, empirically leads to poor recovered object edges because the balanced semantics and coarse saliency compete over where the underlying geometry is. Instead, we weight both equally $\lambda_{\hat{\mathbf{s}}} = \lambda_{\hat{\mathbf{a}}} = \{1/9, 4/9, 4/9\}$. Even though the coarse layer has smaller weight, it is sufficient to guide the overall result. This balances high resolution edges from fine layers and whole object features from coarse layers while reducing geometry conflicts, and leads to improved features (fig. 5.1).

Of course, any sliding window must contain an object to extract reliable features for that object. At coarse levels, an object is always in view. At fine levels, an object is only captured in *some* windows. Objects of interest tend to be near the middle of the frame, meaning that boundary windows at finer pyramid levels contain features that less reliably capture those objects. This can cause spurious connections in clustering. To cope with this, we relatively decrease finer level boundary window weights: We upsample all levels to the finest level, then increase the coarsest level weight towards the frame boundary to $\lambda_{\hat{\mathbf{s}}} = \lambda_{\hat{\mathbf{a}}} = \{1/3, 1/3, 1/3\}$.

5.2.3 Using SAFF for Saliency-aware Clustering

We now wish to isolate salient objects. Even in dynamic scenes, relevant objects may or many not move, meaning that analysis of dynamic elements is not sufficient (cf. [49]). We expand the 2D clustering of Amir

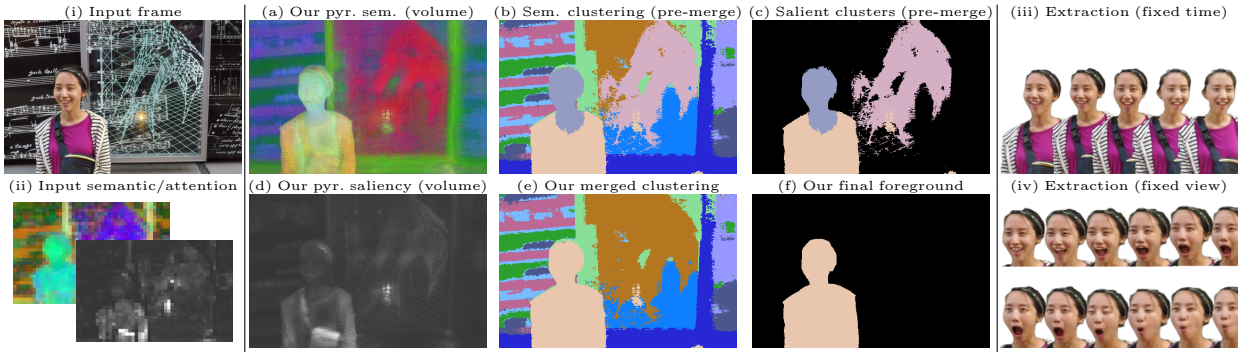

Figure 5.2: **Saliency-aware clustering improves decomposition.** On *Dynamic Face*, the head and body are semantically and saliently different, but are mutually different from the background. This allows us to segment objects cleanly and extract a time-varying 3D field of the object.

et al. [1] to cope with volumes; this allows segmenting novel spatio-temporal views. Even though DINO-ViT features are trained on images, they have been shown to represent saliency over time in videos [4].

Some works optimize a representation with a fixed number of clusters, e.g., via slot attention [22] in NeRFs [38, 56]. Instead, we cluster on 2D volume projections using elbow k-means, letting us adaptively find the number of clusters after optimization. This is more flexible than baking an anticipated number of slots (sometimes with fixed semantics), and lets us cluster and segment in novel spatio-temporal viewpoints.

For N input poses, we render semantics ($N \times H \times W \times 64$) and saliency ($N \times H \times W \times 1$) from the SAFF, and treat each pixel as a feature point. Then, we cluster all pixels together using elbow k-means to produce an initial set of separate regions. For each cluster c, for each image, we calculate the mean attention of all pixels within the cluster $\bar{\mathbf{a}}_c$. If $\bar{\mathbf{a}}_c > 0.07$, then this cluster is salient for this image. Finally, all images vote on saliency: if more than 70% agree, the cluster is salient.

Salient objects may still be split into semantic parts: e.g., in fig. 5.2, the person's head/body are separated. Plus, unwanted background saliency may exist, e.g., input $\hat{\mathbf{a}}^\dagger$ is high for the teal graphic on the wall. As such, before saliency voting, we merge clusters whose centroids have a cosine similarity > 0.5. This reduces the first problem as heads and bodies are similar, and reduces the second problem as merging the graphic cluster into the background reduces its *average* saliency (fig. 5.2).

For novel space-time views, we render feature images from the volume, then assign cluster labels to each pixel according to its similarity with stored centroids from the input views. All clusters not similar to the stored salient clusters are marked as background. To extract an object from the volume, we sample 3D points along each input ray, then ascribe the label from the semantically-closest centroid. We set non-salient label points to have zero density.

5.3 Experiments

We show the impact of adding semantic and saliency features through reconstruction, scene decomposition, and foreground experiments.

Data: Dynamic Scene Dataset (Masked) We use NVIDIA's Dynamic Scene Dataset [54] of eight sequences. Each sequence comprises of 12 cameras simultaneously capturing video at 24 time steps. We manually annotate object masks for view and time step splits; this data will be released.

Then, we define three data splits per sequence:

1. *Input*: A monocular camera that moves position for every timestep is simulated from the input sequences; we use Yoon et al.'s input sequences [54].
2. *Fix Cam 0* (hold out): We fix the camera at position 0 as time plays, requiring novel view and time synthesis. $\{(\text{cam}_0, \text{time}_i), i \in [1, 2, ...23]\}$.
3. *Fix Time 0* (hold out): We fix time at step 0 as the camera moves, requiring novel view and time synthesis. $\{(\text{cam}_i, \text{time}_0), i \in [1, 2, ...11]\}$.

Metrics To assess clustering performance, we use the Adjusted Rand Index (ARI; $[-1, 1]$). This compares the similarity of two assignments without label matching, where random assignment would score $\approx$0. For foreground segmentation, we compute IoU (Jaccard), and for RGB quality we use PSNR, SSIM, and LPIPS.

5.3.1 Comparisons including ablations

We compare to methods that operate on monocular videos and do not require user input or initial masks. While very recent neural volume works use features, none meet these conditions, e.g., related N3F [44] requires user input.

SAFF (ours) We optimize upon the input split of each scene, and perform clustering to obtain object segmentations. To produce a foreground, we merge all salient objects.

— **w/ pyr** $\lambda_{\hat{\mathbf{a}}} = \{1, 0, 0\}$ Pyramid with only coarse saliency (section 5.2.2) and balanced semantic weight across levels.

— **w/o pyr** No pyramid (section 5.2.2); we optimize with features and saliency extracted from the input image only.

— **w/o merge** With pyramid, but we remove cluster merging inside the saliency-aware clustering algorithm.

— **w/ blend** v To compare generic dynamic segmentation to saliency segmentation, we use the static/dynamic weight instead of volume saliency to segment foreground objects. We set every pixel below the 80% v quantile in each image to be background, or otherwise foreground.

— **w/ post process** We add a step after the saliency-aware clustering to refine edges.

NSFF[19] This method cannot produce semantic clusterings. While saliency and blend weight v have different meanings, if we compare our v to NSFF's, then we can see any impact of a shared backbone with attention heads upon the static/dynamic separation. We disable the supervised motion mask initialization in NSFF as our method does not use such information.

D^{2}NeRF[49] This method also cannot produce semantic clusterings. Over HyperNeRF [27], it adds a shadow field network and further losses to try to isolate objects into the dynamic NeRF over the separate static one. The paper also compares to NSFF without motion mask initialization.

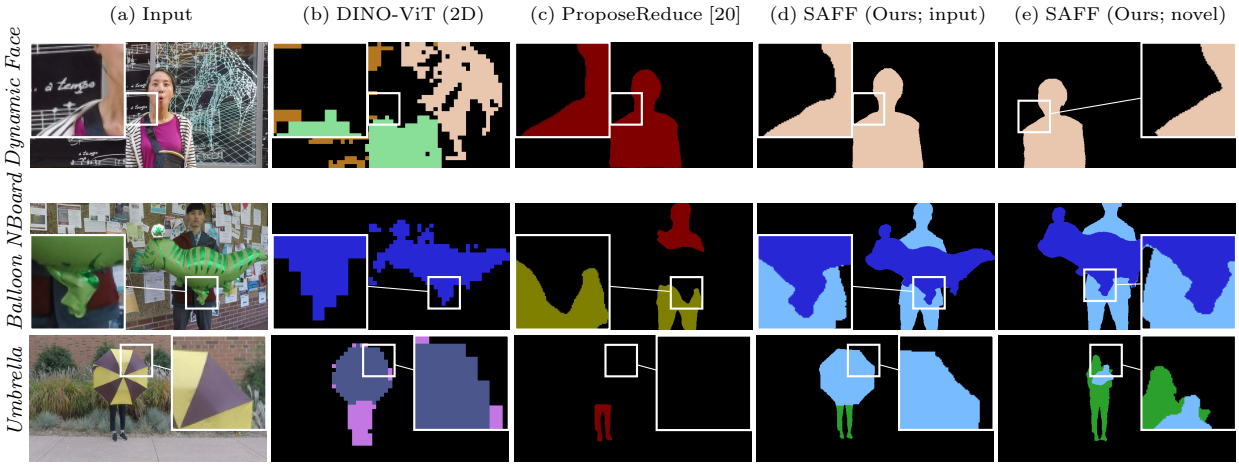

Figure 5.3: **SAFF object segmentations show balanced quality and apply to novel spacetime views (e).** Basic DINO-ViT produces low-quality segmentations and misses objects. A state-of-the-art 2D video learning method [20] sometimes has edge detail (*Umbrella*, legs) but othertimes misses detail and objects (*Balloon NBoard*). Our approach balances these while recovering a 3D scene representation (table 5.2).

Table 5.1: **SAFF does not hurt image quality.** Adding semantics and attention on the same backbone produces the same image quality as NSFF [19]. Metrics: L is LPIPS ($[0, 1]$, lower is better), S is SSIM ($[0, 1]$, higher is better), P is PSNR ($[0, \infty]$, higher is better).

	Input			Fix Cam 0			Fix Time 0		
	L ▼	S ▲	P ▲	L	S	P	L	S	P
D^2NeRF	0.115	0.790	23.91	0.228	0.565	18.04	0.344	0.309	13.85
NSFF w/o masks	0.068	0.804	24.11	0.107	0.752	21.90	0.342	0.313	13.70
SAFF (ours)	0.069	0.804	24.12	0.104	0.754	22.08	0.342	0.313	13.69

DINO-ViT (2D) [1] We ignore the volume and pass 2D semantic and attention features into the clustering algorithm. This cannot apply to novel viewpoints. Instead, we evaluate the approach upon *all* multi-view color images—input and hold-out—whereas other methods must render hold-out views. With pyramid processing (section 5.2.2).

— **w/o pyr** No pyramid; upsample to input RGB size.

ProposeReduce (2D) [20] We apply this state-of-the-art 2D video segmentation method that was pretrained on YouTube-VIS 2019 [53]. As above, we provide hold-out images for splits with novel views. For foreground segmentation, if a pixel belongs to any SAFF saliency object clusters, the pixel is labeled as foreground.

5.3.2 Findings

View synthesis and depth First, we evaluate whether RGB view synthesis performance is affected by adding more heads to the MLP. We find that it is not affected (table 5.1). D^2NeRF's hyper-spacetime deformation has trouble reconstructing images on this dataset, producing distorted dynamic objects or failing to freeze time. For scene geometry over time (depth), we produce similar results to NSFF.

Table 5.2: **Spacetime volume integration improves dynamic scene decomposition.** Our pyramid construction and cluster merging also improve quantitative performance, and our approach is comparable to SOTA 2D video segmenter ProposeReduce on our data. Metric: Adjusted Rand Index ($[-1, 1]$, higher is better).

	Input	Fix Cam 0	Fix Time 0
ProposeReduce (2D)	**0.725**	**0.736**	**0.742**
DINO-ViT (2D)	0.501	0.495	0.321
w/o pyr $\hat{\mathbf{s}}, \hat{\mathbf{a}}$	0.470	0.464	0.346
SAFF (ours)	**0.653**	**0.634**	**0.625**
w/ pyr $\lambda_{\hat{\mathbf{a}}} = \{1, 0, 0\}$	0.620	0.598	0.592
w/o pyr $\hat{\mathbf{s}}, \hat{\mathbf{a}}$	0.545	0.532	0.521
w/o merge cluster	0.593	0.574	0.563
w/ post process	0.759	0.733	0.735
w/ oracle	0.834	0.806	0.800
w/ oracle + post process	0.922	0.890	0.880

Table 5.3: **Foreground segmentation improves by saliency.** Adding saliency also slightly aids how much static/dynamic blend weight v represents the foreground (cf. NSFF blend v to SAFF's). No post process. Metric: IoU/Jaccard index ($[0, 1]$, higher is better).

	Input	Fix Cam 0	Fix Time 0
ProposeReduce (2D)	**0.646**	**0.651**	**0.654**
DINO-ViT (2D)	0.381	0.382	0.357
NSFF — blend v	0.322	0.309	0.268
D^2NeRF— blend v	0.470	0.334	0.269
SAFF — saliency	**0.609**	**0.589**	**0.572**
— blend v	0.388	0.380	0.329

Dynamic scene decomposition Second, we ask the relevant methods to separate the background and each foreground object individually (table 5.2). The baseline 2D DINO-ViT method produces reasonable results, with our pyramid approach in 2D increasing performance. But, being only 2D, this fails to produce a consistent decomposition across novel spacetime views *even* when given ground truth input RGB images. This shows the value of the volume integration for constraining the solution. Next, ProposeReduce can produce good results (fig. 5.3), but sometimes misses salient objects and only sometimes produces better edges than our method, tending to be oversmooth. It is still a 2D representation, and so benefits from being given ground truth images in hold-out sets.

Our approach more reliably identifies objects, with more consistent segmentations through spacetime manipulations—this is the added value of volume integration through learned saliency and attention heads. Ablated components show the value of our pyramid step, its coarse-saliency-only variant, and the cluster merge and image post processing steps. Qualitatively, we see a balance of detail and more often correct object identification (fig. 5.3).

Foreground segmentation Third, we simplify the problem and consider all objects as simply 'foreground' to compare to methods that do not produce per object masks. Here, the same trend continues (table 5.3). We note more subtle improvements to static/dynamic blending weights when adding our additional feature heads

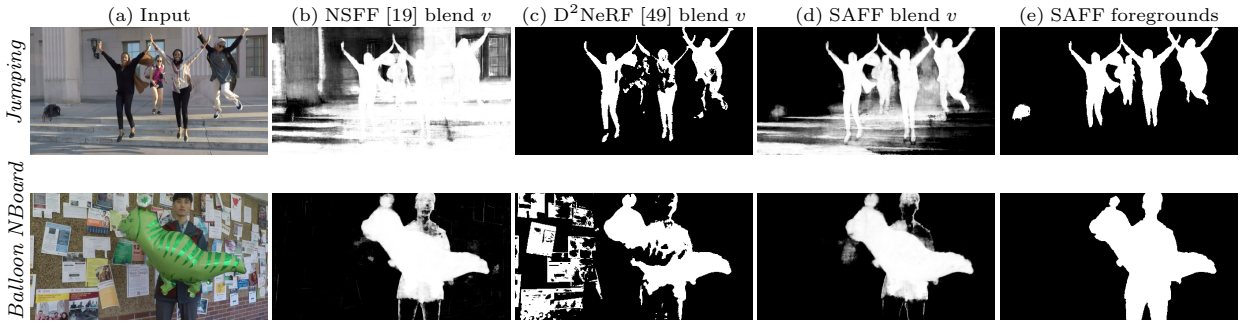

Figure 5.4: **Saliency improves foreground segmentation over static/dynamic weights.** Static/dynamic separations are not foreground segmentations, leading to limited use of dynamic NeRF models for downstream tasks. Our approach produces more useful segmentations. Minor improvements to dynamic blending weight v are seen in some sequences (*Jumping*) by adding the saliency head to the shared backbone.

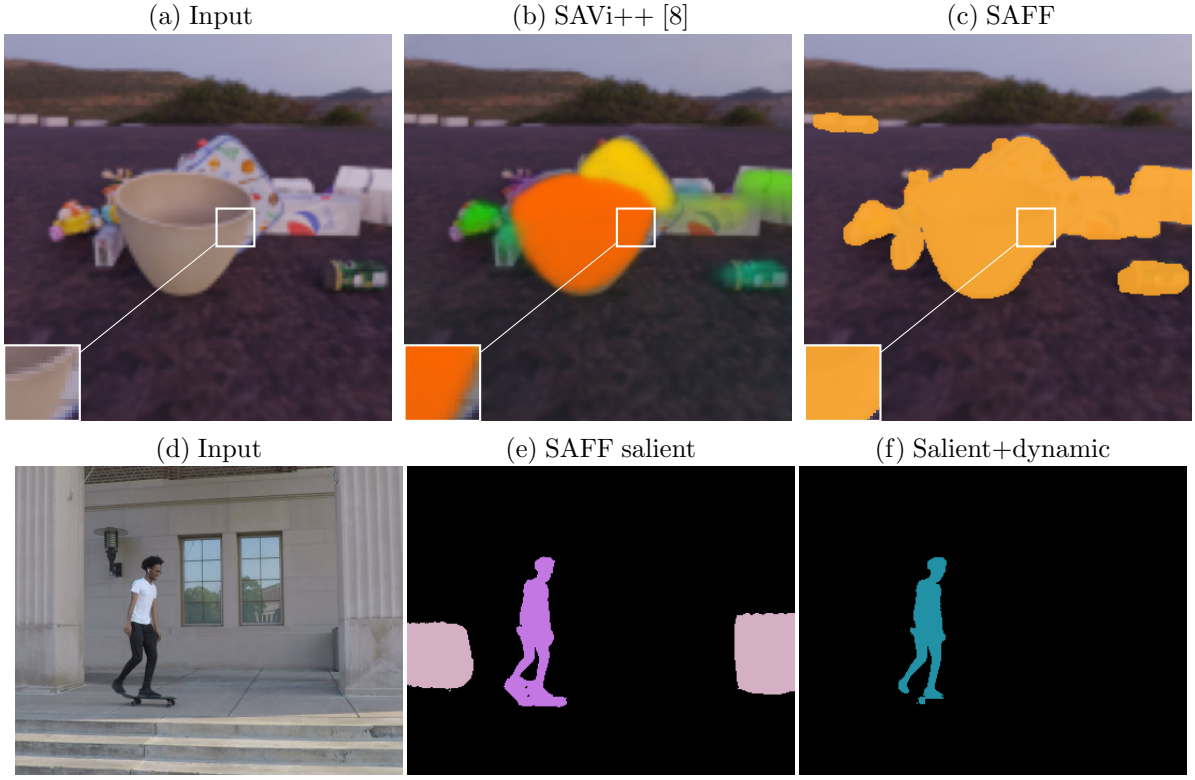

Figure 5.5: **Limitations.** DINO-ViT features are not instance-aware, causing groups of multiple objects (top). Unwanted scene parts may appear salient; (f) assuming salient objects are dynamic fixes this.

to the backbone MLP, and overall show that adding top-down information helps produce more useful object masks. Qualitative results show whole objects in the foreground rather than broad regions of possible dynamics (NSFF) or broken objects (D^{2}NeRF; fig. 5.4).

5.4 Discussion

One question is why we cluster in projected 2D and not in 3D. We demonstrate that our approach still benefits from volume integration without explicit 3D clustering. While volume clustering is possible, the scene reconstruction is only of surfaces and, in the monocular case, the geometry can be noisy. Thus, taking advantage of 3D separation is harder than it appears. It is difficult to collect ground truth segmented 3D data for dynamic real world scenes (none exist to our knowledge); this is an area for future work.

Further, we manage varying saliency over scales through the pyramid and volume approach. Given the concept of saliency, ideally it could be requested from the volume at different scales. This also is an area for future consideration.

Limitations First, as DINO-ViT features are not instance-level, then the clustering is not instance-aware either (fig. 5.5). This is in contrast to object-centric learning approaches, which aim to identify individual objects more effectively. To represent these different approaches, we compare to a result from slot-attention based SAVi++ [8]. This method trains on thousands of supervised MOVi [13] sequences with per-object masks, whereas we tangentially use generic pre-trained features and gain better edges from volume integration. While expensive, combining these two approaches could give accurate instance-level scene objects.

The MOVi sequences are similar to the data that we evaluate our methods with in chapter 3 and chapter 4. The grouping of all objects into one cluster would be particularly bad for these methods, as the objects in the scene do not follow the same motion trajectories. Given that many of the objects do not move, and the ambiguity across object boundaries is propagated from DINO-ViT, additional investigation is needed. This could take the form of improvements to the underlying feature network or improvements to the volume optimization or post-processing methods.

Secondly, DINO-ViT saliency may attend to unwanted regions. In Figure 5.5, bottom, we might think that the static pillars could be isolated using scene flow information. But, often our desired subjects do not move (cf. people in *Umbrella* or *Balloon NBoard*). In applications or data that can assume that salient objects are dynamic, we can exploit SAFF's 4D scene reconstruction to reject static-but-salient objects by also merging clusters via scene flow: First, we project $\mathbf{f}$ over each timestep into each input camera pose—this simulates optical flow with a static camera. Clusters are marked as salient per image if mean flow magnitude per cluster $|\bar{\mathbf{p}}| > 0.07$ *and* mean attention $\bar{\mathbf{a}}_c > 0.07$. Finally, as before, a cluster is globally salient if 70% of images agree (fig. 5.5f).

Chapter 6

Conclusions

We set out to explore the possibilities of a more object-centric model for dynamic scene reconstruction. Our first approach was a point cloud-based approach that used an explicit cluster and scene motion model. We then moved to a version that formulated the clustering as a neural field. Finally, we explored the task of dynamic scene decomposition with SAFF.

For the first two methods, future work will focus on evaluation on a more expansive dataset to get a better sense of the capabilities and limitations of the models. Our ClusterNeRF model shows promise as a dynamic scene representation, but we need a way to construct initial cluster masks for it to optimize the cluster assignment network against. Adapting the model to work with scenes that contain non-rigid subjects (like people) is also a future direction. The model does not inherently fail in this case; the soft cluster assignment should allow for some level of non-rigid scene motion. There is a question of whether the desired effect can be achieved just through the reconstruction losses we have proposed.

There is also potential to merge SAFF with the ClusterNeRF model and use the salient SAFF clusters as a basis for our motion clusters in the NeRF-based model. SAFF would have to be adapted to support clustering in the volume as opposed to in projected 2D images for this to be possible.

A final future research direction is investigating these methods in the online setting. The methods examined in chapter 3 and chapter 4 should have the ability to consume a new frame of image data and quickly estimate object transforms to represent it. This could allow for applications like robot teleoperation, where real-time information is critical. This has been a limitation of existing dynamic NeRF works and would be a big breakthrough.

Dynamic scene reconstruction is a complex 4D problem. Naively optimizing a neural field to represent the scene in 4D has been shown to fail, so we and other researchers have worked to massage the optimization problem into something tractible. In particular, our methods and others leverage object permanence. In previous deformation-based works like Nerfies [26], a network has been required to learn this object performance at every timestep, mapping the points in that frame to the canonical frame. Our methods require this only in the canonical frame, but this adds the limitation of only working for rigid objects.

Neural fields have shown a powerful ability to reconstruct *input signal*, which is evident in our case as well. The input signal reconstruction, however, does not always lead to high quality reconstructions where supervision was not present. This is because neural fields can "cheat" and reconstruct the input data in a way that is

not physically "correct". For NeRF-based methods like ours, this occurs more often when strong multi-view correspondences are not present. For our ClusterNeRF method in chapter 4, we use a dataset with several viewpoints. This is what allows our model to learn the appropriate clustering of the scene, and we hypothesize that without multi-view correspondences it would struggle to learn an accurate segmentation. Even with multi-view data, our model still "cheats." The moving shadows, for instance are somehow reconstructed despite not having a cluster assigned to them. Striking a balance between a model that produces desirable results and one that approximates a physically correct reconstruction of the scene is important and something that future work could continue to investigate. Our previous work on TöRF [2] made progress in this area using time-of-flight data to physically ground scene geometry, and a combination of these two methods could yield dividends.

Bibliography

[1] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *arXiv preprint arXiv:2112.05814*, 2021.

[2] Benjamin Attal, Eliot Laidlaw, Aaron Gokaslan, Changil Kim, Christian Richardt, James Tompkin, and Matthew O'Toole. Törf: Time-of-flight radiance fields for dynamic scene view synthesis. *Advances in Neural Information Processing Systems*, 34, 2021.

[3] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022.

[6] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures, 2022.

[7] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. *CoRR*, abs/2012.09790, 2020.

[8] Gamaleldin F. Elsayed, Aravindh Mahendran, Sjoerd van Steenkiste, Klaus Greff, Michael C. Mozer, and Thomas Kipf. SAVi++: Towards end-to-end object-centric learning from real-world videos. In *Advances in Neural Information Processing Systems*, 2022.

[9] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, 2022.

[10] Xiao Fu, Shangzhan Zhang, Tianrun Chen, Yichong Lu, Lanyun Zhu, Xiaowei Zhou, Andreas Geiger, and Yiyi Liao. Panoptic nerf: 3d-to-2d label transfer for panoptic urban scene segmentation. In *International Conference on 3D Vision (3DV)*, 2022.

[11] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. *CoRR*, abs/2012.03065, 2020.

[12] Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien P. C. Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *CoRR*, abs/2103.10380, 2021.

[13] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J. Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam H. Laradji, Hsueh-Ti Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: A scalable dataset generator. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3739–3751, 2022.

[14] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. *ICCV*, 2021.

[15] Basile Van Hoorick, Purva Tendulka, Dídac Surís, Dennis Park, Simon Stent, and Carl Vondrick. Revealing occlusions with 4d neural fields. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3001–3011, 2022.

[16] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[17] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[18] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas J. Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12871–12881, June 2022.

[19] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[20] Huaijia Lin, Ruizheng Wu, Shu Liu, Jiangbo Lu, and Jiaya Jia. Video instance segmentation with a propose-reduce paradigm. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1719–1728, 2021.

[21] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020.

[22] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*, 2020.

[23] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[24] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022.

[25] Atsuhiro Noguchi, Xiao Sun, Stephen Lin, and Tatsuya Harada. Neural articulated radiance field. In *International Conference on Computer Vision*, 2021.

[26] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021.

[27] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6), dec 2021.

[28] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. *CoRR*, abs/2012.15838, 2020.

[29] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *arXiv preprint arXiv:2011.13961*, 2020.

[30] Amit Raj, Julian Tanke, James Hays, Minh Vo, Carsten Stoll, and Christoph Lassner. Anr: Articulated neural rendering for virtual avatars, 2020.

[31] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022.

[32] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[33] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *CoRR*, abs/2103.13744, 2021.

[34] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022.

[35] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[36] Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *Proc. NeurIPS*, 2021.

[37] Cameron Smith, Hong-Xing Yu, Sergey Zakharov, Frédo Durand, Joshua B. Tenenbaum, Jiajun Wu, and Vincent Sitzmann. Unsupervised discovery and composition of object light fields. *ArXiv*, abs/2205.03923, 2022.

[38] Karl Stelzner, Kristian Kersting, and Adam R. Kosiorek. Decomposing 3d scenes into objects via unsupervised volume segmentation. *ArXiv*, abs/2104.01148, 2021.

[39] Shih-Yang Su, Frank Yu, Michael Zollhöfer, and Helge Rhodin. A-nerf: Articulated neural radiance fields for learning human shape, appearance, and pose. In *Advances in Neural Information Processing Systems*, 2021.

[40] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022.

[41] Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. Variable bitrate neural fields. In *ACM SIGGRAPH 2022 Conference Proceedings*, SIGGRAPH '22, New York, NY, USA, 2022. Association for Computing Machinery.

[42] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020.

[43] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2021.

[44] Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural Feature Fusion Fields: 3D distillation of self-supervised 2D image representations. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2022.

[45] Vadim Tschernezki, Diane Larlus, and Andrea Vedaldi. NeuralDiff: Segmenting 3D objects that move in egocentric videos. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2021.

[46] Chung-Yi Weng, Brian Curless, and Ira Kemelmacher-Shlizerman. Vid2actor: Free-viewpoint animatable person synthesis from video in the wild. *CoRR*, abs/2012.12884, 2020.

[47] Chung-Yi Weng, Brian Curless, Pratul P. Srinivasan, Jonathan T. Barron, and Ira Kemelmacher-Shlizerman. HumanNeRF: Free-viewpoint rendering of moving people from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16210–16220, June 2022.

[48] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[49] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Öztireli. D2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. *ArXiv*, abs/2205.15838, 2022.

[50] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9421–9431, 2021.

[51] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022.

[52] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *International Conference on Computer Vision (ICCV)*, October 2021.

[53] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019.

[54] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[55] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021.

[56] Hong-Xing Yu, Leonidas J. Guibas, and Jiajun Wu. Unsupervised discovery of object radiance fields. In *International Conference on Learning Representations*, 2022.

[57] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew Davison. In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.