Semantic Attention Flow Fields for Monocular Dynamic Scene Decomposition

Yiqing Liang  Eliot Laidlaw  Alexander Meyerowitz  Srinath Sridhar  James Tompkin
Brown University

Abstract

From video, we reconstruct a neural volume that captures time-varying color, density, scene flow, semantics, and attention information. The semantics and attention let us identify salient foreground objects separately from the background across spacetime. To mitigate low resolution semantic and attention features, we compute pyramids that trade detail with whole-image context. After optimization, we perform a saliency-aware clustering to decompose the scene. To evaluate real-world scenes, we annotate object masks in the NVIDIA Dynamic Scene and DyCheck datasets. We demonstrate that this method can decompose dynamic scenes in an unsupervised way with competitive performance to a supervised method, and that it improves foreground/background segmentation over recent static/dynamic split methods.

Project webpage: https://visual.cs.brown.edu/saff

1. Introduction

Given a casually-captured monocular RGB video of a dynamic scene, decomposing it into background and foreground is an important task in computer vision, with downstream applications in segmentation and video editing. An ideal method would also reconstruct the geometry and appearance over time including frame correspondences.

Previous methods have made great progress but there are many limitations. Some works assume that there is no object motion in the scene [32, 46, 31, 16, 4], take input from multi-view cameras [16, 46], or do not explicitly reconstruct the underlying 3D structure of the scene [4, 7]. For objects, some works rely on masks or user input to aid segmentation [42, 15, 7], or use task-specific training datasets [7]. Sometimes, works assume the number of foreground objects [4, 20]. Given the challenges, many works train and test on synthetic data [14, 38].

We present Semantic Attention Flow Fields (SAFF): A method to overcome these limitations by integrating low-level reconstruction cues with high-level pretrained information—both bottom-up and top-down—into a neural volume. With this, we demonstrate that embedded semantic and saliency (attention) information is useful for unsupervised dynamic scene decomposition. SAFF builds upon neural scene flow fields [18], an approach that reconstructs appearance, geometry, and motion. This uses frame interpolation rather than explicit canonicalization [35] or a latent hyperspace [26], which lets it more easily apply to casual videos. For optimization, we supervise two network heads with pretrained DINO-ViT [6] semantic features and attention. Naively supervising high-resolution SAFF with low-resolution DINO-ViT output reduces reconstruction quality. To mitigate the mismatch, we build a semantic attention pyramid that trades detail with whole-image context. Having optimized a SAFF representation for a dynamic scene, we perform a saliency-aware clustering both in 3D and on rendered feature images to describe objects and their background. Given the volume reconstruction, the clustering generalizes to novel spacetime views.

To evaluate SAFF’s dynamic scene decomposition capacity, we expand the NVIDIA Dynamic Scene [45] and DyCheck [10] datasets by manually annotating object masks across input and hold-out views. We demonstrate that SAFF outperforms 2D DINO-ViT baselines and is comparable to a state-of-the-art video segmentation method ProposeReduce [19] on our data. Existing monocular video dynamic volume reconstruction methods typically separate static and dynamic parts, but these often do not represent meaningful foreground. We show improved foreground segmentation over NSFF and the current D²NeRF [40] method for downstream tasks like editing.
2. Related Work

Decomposing a scene into regions of interest is a long-studied task in computer vision [5], including class, instance, and panoptic segmentation in high-level or top-down vision, and use of low-level or bottom-up cues like motion. Recent progress has considered images [12, 4, 20], videos [14, 15], layer decomposition [44, 24], and in-the-wild databases using deep generative models [25]. One example, SAVi++ [7], uses slot attention [20] to define 2D objects in real-world videos. Providing first-frame segmentation masks achieves stable performance, with validation on the driving Waymo Open Dataset [33]. Our work attempts 3D scene decomposition for a casual monocular video without initial masks.

Scene Decomposition with NeRFs Neural Radiance Fields (NeRF) [41] have spurred new scene decomposition research through volumes. ObSuRF [32] and uORF [46] are unsupervised slot attention works that bind a latent code to each object. Unsupervised decomposition is also possible on light fields [31]. For dynamic scenes, works like NeuralDiff [37] and D^2-NeRF [40] focus on foreground separation, where foreground is defined to contain moving objects. Other works like N3F [36] and occlusions-4d [13] 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. Our work attempts to recover a segmented dynamic scene from a single RGB video.

Neural Fields Beyond Radiance Research has begun to add non-color information to neural volumes to aid decomposition via additional feature heads on the MLP. iLabel [47] adds a semantic head to propagates user-provided segmentations in the volume. PNF [17] and Panoptic-NeRF [9] attempt panoptic segmentation within neural fields, and Object-NeRF integrates instance segmentation masks into the field during optimization [42]. Research also investigates how to apply generic pretrained features to neural fields, like DINO-ViT.

DINO-ViT for Semantics and Saliency DINO-ViT is a self-supervised transformer that, after pretraining, extracts generic semantic information [6]. Amir et al. [1] use DINO-ViT features with k-means clustering to achieve co-segmentation across a video. Seitzer et al. [29] combine slot attention and DINO-ViT features for object-centric learning on real-world 2D data. TokenCut [39] performs normalized cuts on DINO-ViT features for foreground segmentation on natural images. Deep Spectral Segmentation [22] show that graph Laplacian processing of DINO-ViT features provides unsupervised foreground segmentation, and Selfmask [30] shows that these features can provide object saliency masks. Our approach considers these clustering and saliency findings for the setting of 3D decomposition from monocular video.

DINO-ViT Fields Concurrent works have integrated DINO-ViT features into neural fields. DFF [16] distills features for dense multi-view static scenes with user input for segmentation. N3F [36] expands NeuralDiff to dynamic scenes, and relies on user input for segmentation. AutoLabel [3] uses DINO-ViT features to accelerate segmentation in static scenes given a ground truth segmentation mask. Other works use DINO-ViT differently. FeatureRealistic-Fusion [21] uses DINO-ViT in an online feature fusion task, focusing on propagating user input, and NeRF-SOS [8] uses 2D DINO-ViT to process NeRF-rendered multi-view RGB images of a static scene. In contrast to these works, we consider real-world casual monocular videos, recover and decompose a 3D scene, then explore whether saliency can avoid the need for masks or user input in segmenting objects.

3. Method

For a baseline dynamic scene reconstruction method, we begin with NSFF from Li et al. [18] (Sec. 3.1), which builds upon NeRF [23]. NSFF’s low-level scene flow frame-to-frame approach provides better reconstructions for real-world casual monocular videos than deformation-based methods [35, 26]. We modify the architecture to integrate higher-level semantic and saliency (or attention) features (Sec. 3.2). After optimizing a SAFF for each scene, we perform saliency-aware clustering of the field (Sec. 3.4). All implementation details are in our supplemental material.

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 [28]. From all poses, we define an NDC-like space that bounds the scene, and a set of rays \(r \in \mathbb{R}\), one per image pixel with color \(\hat{c}^\dagger\). Here, \(\dagger\) denotes a 2D pixel value in contrast to a 3D field value, and \(\dagger\) denotes an input value in contrast to an estimated value.

From pretrained networks, we estimate single-frame monocular depth \(d^\dagger\) (MiDaSv2 [27]), optical flow \(p^\dagger\) (RAFT [34]), and semantic features \(s^\dagger\) and attention \(\alpha^\dagger\) (DINO-ViT [6]) after important preprocessing (Sec. 3.3).

3.1. Initial Dynamic Neural Volume Representation

The initial representation comprises a static NeRF \(F^s_{\theta}\) and a dynamic NeRF \(F^d_{\theta}\). The static network predicts a color \(c\), density \(\sigma\), and blending weight \(v\) (Eq. (1)), and the dynamic network predicts time-varying color \(c_t\), density \(\sigma_t\), scene flow \(f_t\), and occlusion weights \(w_t\) (Eq. (2)). In both network architectures, other than position \(x\), we add direction \(\omega\) to a late separate head to only condition the estimation of color \(c\).

\[
F^s_{\theta} : (x, \omega) \rightarrow (c^s, \sigma^s, v) \quad (1)
\]
\[
F^d_{\theta} : (x, \omega, i) \rightarrow (c^{d_i}, \sigma^{d_i}, f_i, w_i) \quad (2)
\]

To produce a pixel’s color, we sample points at distances \(t\) along the ray \(x_i = x - \omega t\) between near and far planes \(t_n\),
to \( t_f \), query each network, then integrate transmittance \( T \), density \( \sigma \), and color along the ray from these samples [23]. For brevity, we omit evaluating at \( x_i \), \( \omega \), e.g., \( \sigma^s(x_i) \) is simply \( \sigma^s(x_i, \omega) \) is simply \( c^s \).

We produce a combined color from the static and dynamic colors by multiplication with their densities:

\[
\sigma_i c_i = v \sigma^s c^s + (1-v) \sigma^d c^d
\]

(3)

Given that transmittance integrates density up to the current sampled point under Beer-Lambert volume attenuation, the rendered pixel color for the ray is computed as:

\[
\hat{c}_i = \int_{t_n}^{t_f} T_i \sigma_i c_i \, dt \quad \text{where} \quad T_i = \exp \left( - \int_{t_n}^{t_f} \sigma_i \, dt \right)
\]

(4)

To optimize the volume to reconstruct input images, we compute a photometric loss \( \mathcal{L}_c \) between rendered and input colors:

\[
\mathcal{L}_c = \frac{1}{|R|} \sum_{r_i \in R} \| \hat{c}_i(r_i) - c^r_i(r_i) \|^2_2
\]

(5)

Scene Flow Defining correspondence over time is important for monocular input as it lets us penalize a reprojection error with neighboring frames \( j \in \mathcal{N} \), e.g., where \( j = i + 1 \) or \( j = i - 1 \). We denote \( i \rightarrow j \) for the projection of frame \( i \) onto frame \( j \) by scene flow. \( F^d_{\theta} \) estimates both forwards and backwards scene flow at every point to penalize a bi-directional loss.

Thus, we approximate color output at time step \( i \) by flowing the queried network values at time step \( i \):

\[
\hat{c}_{i \rightarrow j} = \int_{t_n}^{t_f} T_{i \rightarrow j} \sigma_{i \rightarrow j} c_{i \rightarrow j} \, dt
\]

(6)

Reprojection must account for occlusion and disocclusion by motion. As such, \( F^d_{\theta} \) also predicts forwards and backwards scene occlusion weights \( w_{i+1} \) and \( w_{i-1} \in [0, 1] \), where a point with \( w_{i+1} = 0 \) means that occlusion status is changed one step forwards in time. We can integrate \( w \) to a pixel:

\[
\hat{w}_{i \rightarrow j} = \int_{t_n}^{t_f} T_{i \rightarrow j} \sigma_{i \rightarrow j} w_j \, dt
\]

(7)

Then, this pixel weight modulates the color loss such that occluded pixels are ignored.

\[
\mathcal{L}_{c_{i \rightarrow j}} = \frac{1}{|R||\mathcal{N}|} \sum_{r_i \in R} \sum_{j \in \mathcal{N}} \hat{w}_{i \rightarrow j}(r_i) \| \hat{c}_i(r_i) - c^r_j(r_j) \|^2_2
\]

(8)

Prior losses We use the pretrained depth and optical flow map losses to help overcome the ill-posed monocular reconstruction problem. These losses decay as optimization progresses to rely more and more on the optimized self-consistent geometry and scene flow. For geometry, we estimate a depth \( d_i \) for each ray \( r_i \) by replacing \( \sigma_i \) in Eq. (4) by the distance \( t \) along the ray. Transform \( z \) estimates a scale and shift as the pretrained network produces only relative depth.

\[
\mathcal{L}_d = \frac{1}{|R|} \sum_{r_i \in R} \| \tilde{d}_i - z(d^r_i) \|_1
\]

(9)

For motion, projecting scene flow to a camera lets us compare to the estimated optical flow. Each sample point along a ray \( x_i \) is advected to a point in the neighboring frame \( x_{i \rightarrow j} \), then integrated to the neighboring camera plane to produce a 2D point offset \( \tilde{p}_i(r_i) \). Then, we expect the difference in the start and end positions to match the prior:

\[
\mathcal{L}_p = \frac{1}{|\mathcal{R}||\mathcal{N}|} \sum_{r_i \in R} \sum_{j \in \mathcal{N}(i)} \| \tilde{p}_i(r_i) - \tilde{p}_j^r(r_i) \|_1
\]

(10)

Additional regularizations encourage occlusion weights to be close to one, scene flow to be small, locally constant, and cyclically consistent, and blending weight \( v \) to be sparse.
3.2. Semantic Attention Flow Fields

Beyond low-level or bottom-up features, high-level or top-down features are also useful to define objects and help down-stream tasks like segmentation. For example, methods like NSFF or D^2NeRF struggle to provide useful separation of static and dynamic parts because blend weight $v$ estimates whether the volume appears to be occupied by some moving entity. This is not the same as objectness; tasks like video editing could benefit from accurate dynamic object masks.

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. As semantics/attention are integrated into the 4D volume, we can render them from novel spacetime views without further DINO-ViT computation.

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

$$F_{s}^{\text{st}} : (x, \omega) \rightarrow (c^{\text{st}}, \sigma^{\text{st}}, v, s^{\text{st}}, a^{\text{st}})$$
$$F_{a}^{\text{dy}} : (x, \omega, i) \rightarrow (c_{i}^{\text{dy}}, \sigma_{i}^{\text{dy}}, f_{i}, w_{i}, s_{i}^{\text{dy}}, a_{i}^{\text{dy}})$$

As semantic features have been demonstrated to be somewhat important over scales and have limited resolution, e.g., DINO-ViT [6] produces one output for each $8 \times 8$ patch. But, from this, we want semantic features and saliency for every RGB pixel that still respects scene boundaries.

$$\sigma_{i}s_{i} = v\sigma^{\text{st}}s^{\text{st}} + (1 - v)s_{i}^{\text{dy}}, \hat{s}_{i} = \int_{t_{s}}^{t_{f}} T_{i} \sigma_{i}s_{i} dt$$

To encourage scene flow to respect semantics over time, we penalize complementary losses on $s$ and $a$ (showing $s$ only):

$$L_{\hat{s}_{i} \rightarrow s_{j}} = \frac{1}{|R||\mathcal{N}|} \sum_{r_{i} \in R_{j}, r_{j} \in \mathcal{N}} ||\hat{s}_{i} - \hat{s}_{j}(r_{j})||^{2}_{2}$$

Finally, as supervision, we add respective losses on the reconstruction of the 2D semantic and attention features from projected 3D volume points (showing $s$ only):

$$L_{s} = \frac{1}{|R|} \sum_{r_{i} \in R} ||\hat{s}_{i} - \hat{s}_{i}(r_{i})||^{2}_{2}$$

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 (please see supplemental).

Thus our final loss becomes:

$$L_{\text{SAFF}} = L_{c} + \lambda s_{i} \rightarrow L_{s_{i} \rightarrow j} + \lambda \hat{s}_{j} \rightarrow \mathcal{L}_{d} + \lambda \hat{s}_{i} \rightarrow \mathcal{L}_{\tilde{d}} + \lambda \hat{s}_{a} \rightarrow \mathcal{L}_{\tilde{a}} + \lambda \hat{s}_{a} \rightarrow \mathcal{L}_{\tilde{a}}$$

3.3. 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 [6] produces one output for each $8 \times 8$ 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 [16] as multi-view constraints provide improved localization within the volume. For monocular video, this approach has limitations [36]. 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 [2], saliency changes significantly over scales and does not preserve energy.

Consider a feature pyramid $P$ with loss weights per level:

$$L_{P_{s}} = \sum_{i \in P} \lambda_{s_{i}}^{i} L_{s_{i}}, L_{P_{a}} = \sum_{i \in P} \lambda_{a_{i}}^{i} L_{a_{i}}$$

Naively encouraging scale-consistent semantics and whole-image saliency, e.g., $\lambda_{s} = \{1/3, 1/3, 1/3\}$ with $\lambda_{a} = \{1, 0, 0\}$, 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 $\lambda_{s} = \lambda_{a} = \{1/3, 1/3, 1/3\}$. 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. 3).

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 $\lambda_{a} = \lambda_{a} = \{1/3, 1/3, 1/3\}$. 
3.4. Using SAFF for Saliency-aware Clustering

We now wish to isolate salient objects. Even in dynamic scenes, relevant objects may not move, so analyzing dynamic elements is insufficient (cf. [40]). One approach predicts segmentation end-to-end [19]. However, end-to-end learning requires priors about the scene provided by supervision, and even large-scale pretraining might fail given unseen test scenes. To achieve scene-specific decompositions from per-video semantics, inspired by Amir et al. [1], we design clustering for spatiotemporal volumes. While DINO-ViT is trained on images, its features are loosely temporally consistent [6].

Some works optimize a representation with a fixed number of clusters, e.g., via slot attention [20] in NeRFs [32, 46]. Instead, we cluster 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.

We demonstrate clustering results in both 3D over time and on rendered volume 2D projections over time. Given the volume reconstruction, we might think that clustering directly in 3D would be better. But, monocular input with narrow baselines makes it challenging to precisely reconstruct geometries: Consider that depth integrated a long a ray can still be accurate even though geometry at specific 3D points may be inaccurate or ‘fluffy’. As such, we use the 2D volume projection clustering results in 2D comparisons.

**Method** For 3D over time, we sample points from the SAFF uniformly along input rays \((128 \times H \times W)\) and treat each pixel as a feature point, e.g., semantics are 64 dim. and saliency is 1 dim. For volume projection, we render to \(N\) input poses \((N \times H \times W)\), and treat each pixel as a feature point instead. In either case, we cluster all feature points 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 feature points within the cluster \(\bar{a}_c\). If \(\bar{a}_c > 0.07\), then this cluster is salient for this image. Finally, all feature points 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. 4, the person's head/body are separated. Plus, unwanted background saliency may exist, e.g., input \(\hat{a}_i\) 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. 4).

To extract an object from the 3D volume, we sample 3D points along each input ray, then ascribe the label from the semantically-closest centroid. All clusters not similar to the stored salient clusters are marked as background with zero density. 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 input view centroids.

4. Experiments

We show the impact of adding semantic and saliency features through scene decomposition and foreground experiments. Our website contains supplemental videos.

**Data: NVIDIA Dynamic Scene Dataset (Masked)** This data [45] has 8 sequences of 24 time steps formed from 12 cameras simultaneously capturing video. We manually annotate object masks for view and time step splits; we will release this data publicly. 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 [45].

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. \(\{(\text{cam}_i, \text{time}_0), i \in [1, 2, ..., 11]\}\).
Figure 4: **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 extract a time-varying 3D field of the object.

Figure 5: **SAFF object segmentations show balanced quality while recovering a volumetric scene representation (e).** Basic DINO-ViT produces low-quality segmentations and misses objects. A state-of-the-art 2D video learning method [19] sometimes has edge detail (*Umbrella*, legs) but otherwise misses detail and objects (*Balloon NBoard*).

---

**Data: DyCheck Dataset (Masked)** For single-camera casual monocular data, we select a subset of Gao et al.’s DyCheck dataset [10]. We remove scenes without an obvious subject, e.g., where telling foreground from background is hard even for a human. We select *haru-sit, space-out, spin, paper-windmill, and mochi-high-five*. We uniformly sample 48 frames and manually annotate object masks. We take even frames as the input set and odd frames as the hold out set. We use the same hyperparameters between both datasets.

**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.

### 4.1. Comparisons including ablations

**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.

**SAFF (3D)** The same as above, but processed in 3D rather than on 2D volume projections.

- **w/ pyr $\lambda_\alpha = \{1, 0, 0\}** Pyramid with only coarse saliency (Sec. 3.3) and balanced semantic weight across levels.

- **w/o pyr** No pyramid (Sec. 3.3); 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 $\nu$** 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% $\nu$ 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 using a conditional random field (please see supplemental material for details). This gains significantly from the depth estimated via volume reconstruction, producing sharp and detailed edges.
Saliency and blend weight are applied to the state-of-the-art 2D video segmentation method. We compare our approach to NSFF, which cannot apply to novel viewpoints. Instead, we evaluate this on synthetic data. With our added pyramid processing (Sec. 3.3), ProposeReduce (2D) produces good results (Fig. 5), but sometimes misses salient objects or fails to join them behind occluding objects, and only works for unsupervised foreground segmentation. As shown in Table 1, our method outperforms DINO-ViT (2D) [1] and Naive 2D NeRF [40] in terms of Adjusted Rand Index (ARI) on all splits.

Dynamic scene decomposition We separate the back- and foreground object individually (Tab. 1). The baseline 2D DINO-ViT method is improved by our pyramid approach. But, being only 2D, this fails to produce a consistent decomposition across novel spacetime views even when given ground truth hold-out images. This shows the value of a volume integration. Next, supervised ProposeReduce produces good results (Fig. 5), but sometimes misses salient objects or fails to join them behind occluding objects, and only sometimes produces better edges than our method without post-processing as it can oversmooth edges. ProposeReduce also receives ground truth images in hold-out sets.

Oracle saliency We also include an oracle experiment where saliency voting is replaced by cluster selection based on ground truth masks. This experiment tells us what part of the performance gap lies with saliency itself, and what remains due to volume integration and cluster boundary issues. With oracle clusters, our decomposition performance is 0.8 ARI (Tab. 1) even in hold-out views. This shows that our existing cluster boundaries are accurate, and that accurate saliency for object selection is the larger remaining problem.

### Figure 6: Saliency improves foreground segmentation
Static/dynamic separations are not foreground segmentations, leading to limited use of dynamic NeRF models for downstream tasks. Minor improvements to dynamic blending weight $v$ are seen in some sequences (Jumping) by adding the saliency head to the shared backbone.

**Table 1: Spacetime volume integration improves dynamic scene decomposition.** Pyramid construction and cluster merging help quantitatively, and ours is comparable to SOTA supervised 2D video segmentation network ProposeReduce. Metric: Adjusted Rand Index ($[-1, 1]$, higher is better).

<table>
<thead>
<tr>
<th>Method</th>
<th>Input Fix Cam 0</th>
<th>Fix Cam 0</th>
<th>Fix Time 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProposeReduce (2D)</td>
<td>0.725</td>
<td>0.736</td>
<td>0.742</td>
</tr>
<tr>
<td>DINO-ViT (2D) w/o pyr</td>
<td>0.501</td>
<td>0.495</td>
<td>0.321</td>
</tr>
<tr>
<td>w/o oracle + post process</td>
<td>0.759</td>
<td>0.733</td>
<td>0.735</td>
</tr>
<tr>
<td>w/ merge cluster</td>
<td>0.593</td>
<td>0.574</td>
<td>0.563</td>
</tr>
<tr>
<td>w/ pyr $\lambda_s$ = {1, 0.0}</td>
<td>0.620</td>
<td>0.598</td>
<td>0.592</td>
</tr>
<tr>
<td>w/o merge cluster</td>
<td>0.545</td>
<td>0.532</td>
<td>0.521</td>
</tr>
<tr>
<td>w/ post process</td>
<td>0.759</td>
<td>0.733</td>
<td>0.735</td>
</tr>
<tr>
<td>w/ oracle</td>
<td>0.834</td>
<td>0.806</td>
<td>0.800</td>
</tr>
<tr>
<td>w/ oracle + post process</td>
<td>0.922</td>
<td>0.890</td>
<td>0.880</td>
</tr>
</tbody>
</table>
Table 2: **Saliency improves foreground segmentation.**
Adding saliency also slightly aids how much static/dynamic blend weight $v$ represents the foreground (cf. NSFF blend $v$ to SAFF’s). Here, ProposeReduce uses unsupervised training and data [5]. Metric: IoU/Jaccard ($[0, 1]$, higher is better).

<table>
<thead>
<tr>
<th></th>
<th>Input Fix Cam 0</th>
<th>Fix Time 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProposeReduce (2D)</td>
<td>0.609</td>
<td>0.464</td>
</tr>
<tr>
<td>DINO-ViT (2D)</td>
<td>0.381</td>
<td>0.382</td>
</tr>
<tr>
<td>NSFF — blend $v$</td>
<td>0.322</td>
<td>0.309</td>
</tr>
<tr>
<td>D$^2$NeRF— blend $v$</td>
<td>0.470</td>
<td>0.334</td>
</tr>
<tr>
<td>SAFF — saliency</td>
<td>0.609</td>
<td>0.589</td>
</tr>
<tr>
<td>— blend $v$</td>
<td>0.388</td>
<td>0.380</td>
</tr>
<tr>
<td>— post process $v$</td>
<td>0.720</td>
<td>0.694</td>
</tr>
</tbody>
</table>

Table 3: **SAFF semantics generalize to DyCheck Dataset (Masked)** Oracle saliency. ProposeReduce is given ground truth test frames, while SAFF must render them. SAFF is comparable to SOTA supervised 2D video segmentation network ProposeReduce. Metric: Adjusted Rand Index ($[-1, 1]$, higher is better); IoU/Jaccard index ($[-1, 1]$, higher is better).

<table>
<thead>
<tr>
<th></th>
<th>ARI</th>
<th>IoU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input</td>
<td>Test</td>
</tr>
<tr>
<td>ProposeReduce (2D)</td>
<td>0.761</td>
<td>0.762</td>
</tr>
<tr>
<td>DINO-ViT (2D)</td>
<td>0.801</td>
<td>0.800</td>
</tr>
<tr>
<td>SAFF (3D)</td>
<td>0.902</td>
<td>0.820</td>
</tr>
</tbody>
</table>

**Foreground segmentation** 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 (Tab. 2). We note more subtle improvements to static/dynamic blending weights when adding our additional feature heads 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. 6).

**DyCheck evaluation** DyCheck often has close-up objects and, even with our selected sequences, saliency struggles to find foreground objects. Thus, we use oracle saliency. DINO-ViT and ProposeReduce are given test views while SAFF must render them. Quantitative (Tab. 3) and qualitative (Fig. 7) experiments show similar trends as before: ProposeReduce is good but may miss objects and fine details; SAFF may produce finer details and is more geometrically consistent.

### 5. Discussion and Limitations

DINO-ViT features are not instance-aware (Fig. 8). This is in contrast to object-centric learning approaches that aim to identify individual objects. To represent these different approaches, we compare to a result from slot-attention based SA Vi++ [7]. This method trains on thousands of supervised MOVi [11] sequences with per-object masks, whereas we use generic pre-trained features and gain better edges from volume integration. Combining these two approaches could give accurate instance-level scene objects.

DINO-ViT saliency may attend to unwanted regions. In Figure 8d–e, the static pillars could be isolated using scene flow. But, often our desired subjects do not move (cf. people in *Umbrella or Balloon NBoard*). For tasks or data that can assume that salient objects are dynamic, we use SAFF’s 4D scene reconstruction to reject static-but-salient objects by merging clusters via scene flow: First, we project $f$ 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 $|p| > 0.07$ and mean attention $\bar{a}_c > 0.07$. Finally, as before, a cluster is globally salient if 70% of images agree (Fig. 8f).

### Acknowledgements
The CV community in New England for feedback, and funding from NSF CNS-2038897 and an Amazon Research Award. Eliot thanks a Randy F. Pausch ’82 Undergraduate Summer Research Award at Brown CS.
References


[31] Cameron Smith, Hong-Xing Yu, Sergey Zakharov, Frédéric Durand, Joshua B. Tenenbaum, Jiajun Wu, and Vincent Sitzmann. Unsupervised discovery and composition of object light fields. ArXiv, abs/2205.03923, 2022. 1, 2


[34] 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. 2


[42] 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. 1, 2


