Gear-NeRF: Free-Viewpoint Rendering and Tracking with Motion-aware Spatio-Temporal Sampling

Xinhang Liu\textsuperscript{1}\textsuperscript{*} Yu-Wing Tai\textsuperscript{2} Chi-Keung Tang\textsuperscript{1}\textsuperscript{†} Pedro Mirdal\textsuperscript{3} Suhas Lohit\textsuperscript{3} Moitrey Chatterjee\textsuperscript{3}

\textsuperscript{1}HKUST \textsuperscript{2}Dartmouth College \textsuperscript{3}Mitsubishi Electric Research Laboratories (MERL)
xliufe@connect.ust.hk, yu-wing.tai@dartmouth.edu, cktang@cse.ust.hk, miraldo@merl.com, slohit@merl.com, metro.smiles@gmail.com

Abstract

Extensions of Neural Radiance Fields (NeRFs) to model dynamic scenes have enabled their near photo-realistic, free-viewpoint rendering. Although these methods have shown some potential in creating immersive experiences, two drawbacks limit their ubiquity: (i) a significant reduction in reconstruction quality when the computing budget is limited, and (ii) a lack of semantic understanding of the underlying scenes. To address these issues, we introduce Gear-NeRF, which leverages semantic information from powerful image segmentation models. Our approach presents a principled way for learning a spatio-temporal (4D) semantic embedding, based on which we introduce the concept of gears to allow for stratified modeling of dynamic regions of the scene based on the extent of their motion. Such differentiation allows us to adjust the spatio-temporal sampling resolution for each region in proportion to its motion scale, achieving more photo-realistic dynamic novel view synthesis. At the same time, almost for free, our approach enables free-viewpoint tracking of objects of interest—a functionality not yet achieved by existing NeRF-based methods. Empirical studies validate the effectiveness of our method, where we achieve state-of-the-art rendering and tracking performance on multiple challenging datasets. The project page is available at: https://merl.com/research/highlights/gear-nerf.

1. Introduction

Reconstructing 3D scenes has a broad range of applications, including Virtual Reality/Augmented Reality (VR/AR), 3D animation, game production, and film creation which allow users to observe scenes from any desired viewpoint. While it is crucial to reconstruct static scenes, towards which significant progress has been made, it is even more crucial to reconstruct dynamic scenes, as the world around us is characterized by a constant state of flux, with many objects in it - in a state of motion.

Recent advances in novel view synthesis, such as Neural Radiance Fields (NeRFs) [56] have inspired numerous studies to extend them to dynamic 3D scenes. Existing approaches either employ a deformation field to map neural fields from a given time to a canonical space [22, 45, 60, 61, 65, 99], or directly model dynamic scenes as a 4D space-time grid [1, 6, 26]. Though these methods offer improved rendering quality by utilizing more accessible inputs compared to previous solutions [21, 42, 43, 77], they still struggle to ensure rendering quality in low-resource settings, requiring carefully engineered efforts. Further, most dynamic radiance field approaches adopt a naive spatio-temporal sampling strategy, without discerning the different scales of motion across different regions in the scene.

We propose to fix this issue by leveraging a semantic understanding of dynamic scenes. Intuitively, a reconstruction system aware of the distinction between static and dynamic regions in a scene can perform more focused
sampling in the dynamic regions, which inherently require more resources per unit volume than static regions, due to their time-evolving nature. Accordingly, dynamic regions can be further stratified according to their scale of motion. To this end, this paper presents Gear-NeRF, a framework that leverages semantic embedding from powerful image segmentation models [38] for stratified modeling of 4D scenes. Gear-NeRF optimizes for a 4D semantic embedding, based on which we introduce the concept of gear to smartly determine the appropriate region-specific resolution of spatio-temporal sampling in the NeRF. Regions with larger motion scales are automatically assigned higher gears, through our gear determination scheme and we accordingly perform higher-resolution spatio-temporal sampling. Empirical studies reveal that this motion-aware sampling strategy improves the quality of synthesized images, over competing approaches. As a by-product of our semantically embedded representation, we achieve free-viewpoint object tracking, given user prompts. Figure 1 presents an overview of the capabilities of our method.

Gear-NeRF makes two primary advancements: (i) enhanced dynamic novel view synthesis by resorting to smarter spatio-temporal sampling, and (ii) the ability for free-viewpoint tracking of objects of interest. The latter is a capability not yet realized by existing NeRF methods for dynamic scenes. We perform extensive experiments on multiple datasets to validate the generalizability and robustness of our method, which shows state-of-the-art performances for both tasks across all datasets.

2. Related Work

Neural Radiance Fields: NeRF [56] is a recent breakthrough among novel view synthesis methods that uses multilayer perceptrons (MLPs) to parameterize the appearance and density for each point in 3D space, given any viewing direction of the scene. Researchers have extended NeRF along various dimensions [78], including improving rendering quality [2–4, 12, 18, 31], handling challenging conditions such as large scenes [55, 68, 75], view-dependent appearances [30, 52, 80], and sparse inputs [34, 51, 59, 86, 87, 93]. NeRF-like neural representations have also found applications in semantic segmentation [40, 50, 102] and 3D content generation [11, 25, 58, 74]. Recent work has shown that replacing the deep MLPs with a feature voxel grid can significantly improve training and inference speed [11, 25, 58, 74]. On the other hand, a more recent approach to further improve visual quality, rendering time, and performance entails representing the scene with 3D Gaussians [36]. Our approach, while drawing upon many of these approaches, deals with dynamic scenes which is beyond the scope of these methods.

Neural Representations for Dynamic Scenes: NeRF-like representations have recently been extended to model dynamic scenes in high fidelity [17, 28, 33, 39, 46–48, 53, 63, 66, 84, 89, 90, 97]. One straightforward approach to do this is to directly condition the radiance field on time [27, 44, 45, 88]. Alternatively, several methods model a deformation field to map coordinates from different time stamps to a common canonical space [22, 24, 60, 61, 65, 79, 98, 99]. Some recent approaches [6, 26, 71, 81] represent the scene using a 4D space-time grid, which is decomposed into sets of planar representations for training efficiency. Other techniques for improving rendering fidelity and frame rate include Fourier PlenOctrees [82], ray-conditioned sample prediction networks [1], 4D space decomposition (static/dynamic/newly appeared regions) [73], and explicit voxel grids [24]. 3D Gaussians have also been adapted to model dynamic scenes [54, 85, 95, 96]. While these approaches paved the initial path for rendering dy-
dynamic scenes, semantically aware modeling of the scene is absent, a caveat that our proposed method seeks to address. **Segment Anything Model (SAM):** SAM [38] is a powerful promptable image segmentation model, which showcases remarkable zero-shot generalization abilities and can produce semantically consistent masks, given a single foreground point on the image. HQ-SAM [35] is an improvement on SAM that enhances the quality of masks, especially on objects with intricate boundaries and structures. Recent works [20, 92] have extended SAM to perform interactive video object segmentation. These methods utilize SAM for mask initialization or correction and then employ state-of-the-art mask trackers [19, 94] for mask tracking and prediction [14]. Recent methods have also leveraged SAM for tracking multiple reference objects in a video [13, 101]. This work uses SAM to profile the scene into semantic regions, which are then grouped based on motion scales.

**3D Semantic Understanding:** Existing methods for 3D visual understanding [15, 32, 76, 91] mainly focus on closed set segmentation of point clouds or voxels. NeRF’s ability to integrate information across multiple views has led to its applications in 3D semantic segmentation [102], object segmentation [23, 50, 57], panoptic segmentation [72], and interactive segmentation [29, 69]. Kobayashi et al. [40] explored the effectiveness of embedding pixel-aligned features [7, 41] into NeRFs for 3D manipulations. LERF [37] fuses CLIP embeddings [67] and NeRFs to enable language-based localization in 3D NeRF scenes. Recent work has enabled click/text-based 3D segmentation by learning a 3D SAM embedding [16] or inverse rendering of SAM-generated masks [8]. We, on the other hand, seek to utilize the synergy of dynamic NeRFs and SAM segments to derive a semantic understanding of a dynamic 3D scene for tracking objects of interest in novel views—a first of its kind effort.

### 3. Preliminaries

**Neural Radiance Fields (NeRFs):** Vanilla NeRFs [56] employ a multi-layer perceptron (MLP) with sinusoidal positional encoding to map a 3D-spatial coordinate \( x = (x, y, z) \) and a viewing direction \( d = (\theta, \phi) \) to a volume density \( \sigma \in [0, 1] \) and an emitted RGB, \( c \in \mathbb{R}^3 \). Rendering each image pixel involves casting a ray \( r(t) = o + td \) from the camera center \( o \) through the pixel along direction \( d \). The predicted color for the corresponding pixel is computed as:

\[
\hat{C}(r) = \sum_{i=1}^{N} T_i \alpha_i c_i,
\]

where \( T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right), \alpha_i = 1 - \exp(-\sigma_i \delta_i), \) and \( \delta_j = t_{j+1} - t_j \). A vanilla NeRF is trained by minimizing the mean squared error between the input images and the predicted images, obtained by rendering the scene from the viewpoints from which the input images have been captured, with the training loss given by:

\[
\mathcal{L}_{\text{pho}} = \sum_{r \in \mathcal{R}} \| \hat{C}(r) - C(r) \|^2.
\]

where \( \mathcal{R} \) is the set of all rays projected from the input image. **Planar-Factorized 4D Volumes:** A recent emerging trend of handling dynamics using radiance field representations is to directly adapt them to be conditioned on a frame index \( t \) (denoting time) in addition to \( x \) and \( d \). This can be accomplished by learning a mapping from \((x, d, t) \) to \((\sigma, c)\) using planar-factorized 4D volumes [1, 6, 26, 71]. These methods attempt to learn a 4D feature vector for every \((x, t)\), by projecting it to a set of 2D-planes. Embeddings of these projections on these planes can then be integrated to obtain the embedding for the 4D point. This can be mathematically represented as follows:

\[
f(x, t) = B_1(h_1(x, y) \odot k_1(z, t)) + B_2(h_2(x, z) \odot k_2(y, t)) + B_3(h_3(y, z) \odot k_3(x, t)).
\]

where \( h_i(\cdot, \cdot) \) and \( k_i(\cdot, \cdot) \) are functions (evaluated by bilinear interpolation on regularly spaced 2D feature grids) embedding coordinate tuples to features of dimension \( M \), “\( \odot \)” denotes an element-wise product, and \( B_i(\cdot) \) denotes a linear transform which maps the products to feature vectors. Subsequently, a tiny MLP can map the feature vector \( f(\cdot, \cdot) \) to the volume density, \( \sigma \), and the view-dependent emitted color, \( c \), given the viewing direction \( d \).

### 4. Proposed Method

Given a set of \( W \) input videos, \( V = \{V_1, V_2, \ldots, V_W\} \) of a dynamic scene, with calibrated camera poses, our approach represents the scene using a series of 4D feature volumes (Section 4.1) along with 4D semantic embeddings (Section 4.2).

Analogous to multiple gears in motor vehicles for optimizing engine performance, Gear-NeRF stratifies this semantically embedded scene representation into \( N_{\text{gear}} \) levels, based on the motion scales. Each of these levels is called a gear. Through our training scheme, regions with larger motion are assigned higher gear levels (Section 4.3) and as a result, are more densely sampled (Section 4.4) for improved dynamic novel view synthesis. Our 4D semantic embedding also enables a new functionality, almost for free—free-viewpoint tracking of target objects, given simple user prompts like clicks (Section 4.5). Figure 2 shows the overall pipeline of Gear-NeRF.

#### 4.1. Serial 4D Feature Volumes

Instead of using a unified 4D volume to represent a dynamic scene [1, 6, 26, 71], our representation consists of a series of feature volumes, each corresponding to a gear level, \( G \). Specifically, for any space-time coordinate \((x, t)\), its feature
The vector-valued functions $\mathbf{h}_j(\cdot, \cdot)$ and linear transforms $\mathbf{B}_j(\cdot)$ are shared by all gears, while each gear has its own spatio-temporal embedding $\mathbf{k}_j^g(\cdot, \cdot)$, in $M$-dimensional space. Therefore, each gear describes regions of a certain scale of motion while the purely spatial features can be shared among all gears.

We obtain the gear level at any spatio-temporal coordinate also from a planar-factorized 4D feature volume. Specifically, the gear level at $(x, t)$ is computed as:

$$g(x, t) = 1^T \mathbf{h}_1^g(x, y) \odot \mathbf{k}_1^g(z, t) + 1^T \mathbf{h}_2^g(x, z) \odot \mathbf{k}_2^g(y, t) + 1^T \mathbf{h}_3^g(y, z) \odot \mathbf{k}_3^g(x, t),$$  \hspace{1cm} (5)

where $1$ is a vector of ones, $\mathbf{h}_i^g(\cdot, \cdot)$ and $\mathbf{k}_i^g(\cdot, \cdot)$ are $M$-dimensional embedding functions. This however defines a continuous feature volume. To map it to the gear level integers, we apply the following projection operation:

$$p(x, t) = \begin{cases} 1, & \text{if } g(x, t) < 1, \\ N_{\text{gear}}, & \text{if } g(x, t) \geq N_{\text{gear}}, \\ \lfloor g(x, t) \rfloor, & \text{otherwise}. \end{cases}$$  \hspace{1cm} (6)

Based on this gear level volume, we define a 4D mask for a region at gear level $\mathcal{G}$ as:

$$m_\mathcal{G}(x, t) = \begin{cases} 1, & \text{if } p(x, t) = \mathcal{G}, \\ 0, & \text{otherwise}. \end{cases}$$  \hspace{1cm} (7)

The final feature vector at $(x, t)$ is computed as:

$$\mathbf{f}(x, t) = \sum_{\mathcal{G} = 1}^{N_{\text{gear}}} m_\mathcal{G}(x, t) \mathbf{f}_\mathcal{G}(x, t).$$  \hspace{1cm} (8)

Subsequently, a tiny MLP, $F_\theta$, maps these feature vectors $\mathbf{f}(\cdot, \cdot)$ as well as the viewing direction $\mathbf{d}$ to the volume density $\sigma$ and radiance color $c$. This allows us to obtain a photometric rendering of the scene.

### 4.2. 4D Semantic Embedding

Gear-NeRF leverages the strong object priors of the SAM [38] model to acquire a semantic understanding of the scene, for improved photometric rendering (Section 4.3 and Section 4.4) as well as free-viewpoint object tracking (Section 4.5). Toward this end, we utilize the SAM encoder to obtain 2D feature maps from the frames of each video. We then optimize a 4D SAM embedding field by supervising it with these 2D feature maps. In particular, the MLP above, $F_\theta$, is configured to output a 4D semantic embedding $s$ for a given space-time coordinate in addition to the density, $\sigma$, and color, $c$. To render 2D semantic feature maps in a given view, we compute the semantic feature of a pixel in the feature map by tracing a ray through it and perform volume rendering analogous to Equation 1, as follows:

$$\hat{S}(\mathbf{r}) = \sum_{i=1}^{N} T_i \alpha_i s_i.$$

This SAM embedding is supervised by minimizing the mean squared error between the prediction and the ground truth features ($S(\mathbf{r})$) from the SAM encoder, as shown:

$$\mathcal{L}_{\text{SAM}} = \sum_{\mathbf{r} \in \mathcal{R}} \|\hat{S}(\mathbf{r}) - S(\mathbf{r})\|_2^2.$$  \hspace{1cm} (10)

### 4.3. Training Scheme with Gear Assignment

With gear initialization $g(x, t) = 1, \forall x, t$, the (semantically embedded) radiance field optimization and gear assignment updating take place in an alternating fashion.

**Gear Assignment Update**: As illustrated in Figure 3, when updating the gear assignment after a period of radiance field
optimization, we find the regions rendered most poorly from the rendering loss maps and increment their gears for denser spatio-temporal sampling. The following steps lay out the process for updating gear assignments to regions:

- We sample a number of viewpoints and time steps and render 2D-images/SAM features for it. For every rendered RGB-SAM map, we compute a rendering loss map. Each pixel of the rendering loss map is computed as: \( L(r) := L_{rgb}(r) + \lambda L_{SAM}(r) \). See Figure 3 for example loss maps.

- Next, we patchify each rendering loss map to find patches with the top-\(k\) largest/smallest average loss. The center coordinate of these patches serve as positive/negative prompts for the next step. See Figure 3 for example positive (red) / negative (green) prompts.

- We then feed the ground truth RGB images together with positive and negative prompts into the SAM decoder [38] to estimate an upshift mask. These masks tend to cover regions that have motions and are not satisfactorily rendered with the current sampling resolution. We have multiple upshift masks at different viewpoints and time steps.

- For every pixel of an upshift mask, we trace a ray and sample points along it and update the gear assignment by pushing \( g(x, t) \) towards incremented values.

In particular, in the last step, for each pixel within an upshift mask, a corresponding ray is traced that connects it with the camera center \( c \), along direction \( d \). Next, a set of points are sampled along this ray. The collection of sampled points, that lie on the rays emanating from within the masked region constitutes the set \( S_{\text{upshift}} = \{(x_i^{\text{upshift}}, t_i^{\text{upshift}})\}_{i=1}^{N_{\text{upshift}}} \), where \( N_{\text{upshift}} \) represents the total count of points sampled from rays pertaining to the masked region. We then follow a similar procedure to sample a set of points pertaining to the unmasked region. We denote this set as: \( S_{\text{stay}} = \{(x_i^{\text{stay}}, t_i^{\text{stay}})\}_{i=1}^{N_{\text{stay}}} \), with \( N_{\text{stay}} \) indicating the total number of points sampled from rays that pertain to the unmasked area. Next, for each sample point in each of \( S_{\text{upshift}} \) and \( S_{\text{stay}} \), we query its current gear level \( p(x, t) \). In order to assign new gear levels, we need to update the gear assignment function \( g(\cdot, \cdot) \), which proceeds with the objective function:

\[
L_{\text{upshift}} = \frac{1}{N_{\text{upshift}}} \sum_{(x, t) \in S_{\text{upshift}}} \| g(x, t; \Theta) - (p(x, t) + 1) \|^2 + \frac{\lambda_{\text{stay}}}{N_{\text{stay}}} \sum_{(x, t) \in S_{\text{stay}}} \| g(x, t; \Theta) - p(x, t) \|^2,
\]

where \( \Theta \) denotes the set of optimizable parameters for \( g(\cdot, \cdot) \). The minimization of the first term encourages sample points within the masked region \( S_{\text{upshift}} \) to have a gear level equal to their current gear incremented by one, resulting in an upshift of gear. Conversely, minimizing the second term encourages the points in \( S_{\text{stay}} \) to maintain their gear values, thereby encouraging the remaining regions to keep their current gear levels and avoiding unwanted upshifts. We update \( \Theta \) via a single step of gradient descent as follows:

\[
\Theta \leftarrow \Theta - \alpha \nabla_{\Theta} L_{\text{upshift}},
\]

where \( \alpha \) is the learning rate. Since, \( g(\cdot, \cdot) \) is essentially derived from the embedding functions, \( h_i(\cdot, \cdot) \) and \( k_i(\cdot, \cdot) \) for \( i \in \{1, 2, 3\} \), the aforementioned optimization step amounts to updating these embedding functions. Once updated, we proceed with a fresh round of gear assignment to increase the spatio-temporal sampling resolution for the regions that end up at a higher gear level than before.

**Radiance Field Optimization:** With the updated gear assignment, we increase the resolution of spatio-temporal sampling for the gear-shifted regions (Section 4.4) and then resume optimizing the radiance field. We alternate between the two processes: radiance field optimization (each time for \( L \) epochs), and gear assignment updates until the average variance of each rendering loss map is below a predetermined threshold. After this, we optimize the radiance field for an additional \( L' \) epochs without further gear assignment updates.

### 4.4. Motion-aware Spatio-Temporal Sampling

In this subsection, we explain our motion-aware spatio-temporal sampling strategy based on assigned gears, permitting differential processing of regions at different gear levels. By temporal sampling, we imply the choice of temporal resolution for planar-factorized 4D feature volumes, and by spatial sampling, we mean the strategy used to choose sampling points along each ray for volume rendering.

**Motion-aware Temporal Sampling:** To handle the increasing intensity of object motion, as reflected by their growing gear levels, we increment the temporal resolution for voxel grids. Specifically, \( k_j^G \) in Equation 4 has increasing resolution along the time axis, thereby empowering the 4D feature volumes to better model the dynamics along the temporal axis. This ensures fast-moving objects can be more faithfully modeled without unsightly blurring. The temporal resolution for each gear’s feature volume is determined by linear interpolation between 1 (for \( G = 1 \)) and the total number of frames (for \( G = N_{\text{gear}} \)).

**Motion-aware Spatial Sampling:** While denser sampling of points can improve reconstruction accuracy, increasing the number of sampling points throughout the scene can lead to prohibitive computational costs. Therefore, we propose a 3D point-splitting strategy as illustrated in Figure 4. We begin by sampling a relatively small number, \( n \), of samples along each ray, assuming it is at the lowest gear level. If a sampled point belongs to a region with a higher gear, as determined by \( p(x, t) \), we then sample more densely in that region. For every sampled point in that region, we split it into \( 2^g(x, t) - 1 \) points, equally spaced within the corresponding ray segment (at that gear level).
4.5. Free-Viewpoint Tracking with User Prompts

Our 4D SAM embedding enables another useful functionality, almost for free – free-viewpoint object tracking, where the user only needs to provide as few as one click to extract the target object based on the 4D embedding. Next, we describe how, given a user-supplied point click at any arbitrary viewpoint and time step, we obtain the corresponding object mask at a novel viewpoint and time step.

Masks for Novel Viewpoints: The first step for this task entails finding the 3D correspondence of the user click. We trace a ray through the selected pixel, and by utilizing the volume density, we determine the depth at which the ray intersects with the first object surface it encounters. This yields the 3D coordinates of the point of intersection. Subsequently, the 3D coordinates of this intersection can be easily mapped into a 2D coordinate within any novel viewpoint image, using the camera pose of the new viewpoint. Alongside the rendered SAM feature map of the novel view, we feed this coordinate into the SAM decoder to generate the object mask for the novel viewpoint.

Masks for Novel Time Steps: For this task, we propagate an object mask to its neighboring time step. Specifically, with an object mask for a specific frame $t$, we calculate the bounding box of this mask and use this bounding box as a prompt to SAM for neighboring frames $t' = t + 1$ or $t' = t - 1$. By inputting this prompt along with the rendered SAM feature map at $t'$ into the SAM decoder, we can obtain the object mask for $t'$. Combining the above two processes, we can start from a single click and get the object mask in any viewpoint and time step.

5. Experiments

We assess the performance of our proposed Gear-NeRF for dynamic novel view synthesis and free-viewpoint object tracking across a range of challenging datasets, comparing it with state-of-the-art methods. Through ablation studies, we provide empirical evidence of the effectiveness of its fundamental components. We kindly refer the reader to our supplementary material for additional experimental details and results, including videos for free-viewpoint rendering and tracking.
method against a range of recent NeRF-based baseline methods: (i) ST-NeRF [99], (ii) HexPlane [6], (iii) HyperReel [1], and (iv) MixVoxels [81]. As the first method to enable promptable free-viewpoint object tracking under the NeRF setting, there is no established direct baseline for this specific task. However, SA3D [8], a recent method for segmenting static scenes, is treated as a baseline for predicting masks at novel viewpoints corresponding to the prompted time step.

5.2. Results

Dynamic Novel View Synthesis: As shown in Figure 5, Gear-NeRF produces high-quality novel view synthesis of dynamic scenes, accurately modeling real-world scenes with intricate motions and fine details. For example, patterns on the toys or stripes on the shirt, are faithfully rendered, resulting in more photo-realistic images compared to all of the baselines. Table 1, presents quantitative comparisons of our method against the baselines. While Gear-NeRF has longer training (Tr. Time) / inference times (FPS) compared to some baselines, it almost always achieves the best performance in terms of rendering quality.

Free-Viewpoint Tracking: In Figure 6, we present the object masks obtained by our method based on the user prompts provided at a specified viewpoint and time step. Specifically, the first row displays masked objects at the prompted viewpoint and time step. The second row shows novel view masks at the prompted time step. The third row shows novel view masks at novel time steps. We see that masks obtained from Gear-NeRF show precise boundaries, compared to SA3D. Table 2 presents quantitative assessment of the quality of the masks generated of our method versus SA3D. Our method exceeds 90% across all metrics and datasets, demonstrating the effectiveness of our approach. Our gains over SA3D can be attributed to the fact that SA3D, as a static scene segmentation method, does not utilize information across all time frames, whereas our approach does. SA3D is incapable of predicting masks for novel time steps, and as such, the corresponding entries are marked as not applicable, a shortcoming which our method does not have.

5.3. Evaluations

To verify the effectiveness of the design choices of Gear-NeRF, we perform extensive ablation studies on the Truck scene of the Google Immersive dataset [5].

Motion-aware Temporal Sampling: An intuitive strategy for adjusting the temporal sampling is to directly modify

Table 1. Quantitative comparisons for dynamic novel view synthesis: Our method outperforms all baselines across all datasets on all metrics. We report means over all scenes for each dataset. Best and second best results are highlighted.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>PSNR (↑)</th>
<th>SSIM (↑)</th>
<th>LPIPS (↓)</th>
<th>Tr. Time (↑)</th>
<th>FPS (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technicolor [70]</td>
<td>ST-NeRF [99]</td>
<td>30.86</td>
<td>0.883</td>
<td>0.101</td>
<td>344 min</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td>HyperReel [1]</td>
<td>31.04</td>
<td>0.887</td>
<td>0.092</td>
<td>97 min</td>
<td>7.7</td>
</tr>
<tr>
<td></td>
<td>MixVoxels [81]</td>
<td>28.52</td>
<td>0.873</td>
<td>0.107</td>
<td>14 min</td>
<td>18.9</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>32.21</td>
<td>0.919</td>
<td>0.058</td>
<td>148 min</td>
<td>7.9</td>
</tr>
<tr>
<td>Google Immersive [5]</td>
<td>HexPlane [6]</td>
<td>27.67</td>
<td>0.808</td>
<td>0.188</td>
<td>527 min</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>HyperReel [1]</td>
<td>28.32</td>
<td>0.862</td>
<td>0.145</td>
<td>117 min</td>
<td>7.4</td>
</tr>
<tr>
<td></td>
<td>MixVoxels [81]</td>
<td>27.74</td>
<td>0.875</td>
<td>0.207</td>
<td>17 min</td>
<td>5.8</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>28.74</td>
<td>0.876</td>
<td>0.122</td>
<td>189 min</td>
<td>7.0</td>
</tr>
<tr>
<td>Neural 3D Video [44]</td>
<td>ST-NeRF [99]</td>
<td>31.03</td>
<td>0.890</td>
<td>0.084</td>
<td>367 min</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td>HyperReel [1]</td>
<td>31.12</td>
<td>0.928</td>
<td>0.065</td>
<td>129 min</td>
<td>6.1</td>
</tr>
<tr>
<td></td>
<td>MixVoxels [81]</td>
<td>30.69</td>
<td>0.944</td>
<td>0.179</td>
<td>15 min</td>
<td>16.3</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>31.80</td>
<td>0.936</td>
<td>0.058</td>
<td>204 min</td>
<td>6.8</td>
</tr>
</tbody>
</table>


Figure 6. Qualitative comparisons of free-viewpoint object tracking on Technicolor [70] and Neural 3D Video [44] datasets: Our method can obtain desirable object masks, with clear edges, from prompting points provided by users at desired time steps and viewpoints.

Table 2. Quantitative comparisons for free-viewpoint tracking: t-mIoU and t-Acc are metrics used for evaluating novel view masks at novel time steps, not applicable to SA3D. Reported metrics are averages over all scenes for each dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>mIoU (%)</th>
<th>Acc. (%)</th>
<th>t-mIoU (%)</th>
<th>t-Acc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technicolor [70]</td>
<td>SA3D [8]</td>
<td>96.4</td>
<td>97.1</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>97.4</td>
<td>97.6</td>
<td>92.1</td>
<td>93.3</td>
</tr>
<tr>
<td>Google Immersive [3]</td>
<td>SA3D [8]</td>
<td>94.1</td>
<td>94.8</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>94.3</td>
<td>95.0</td>
<td>91.5</td>
<td>92.8</td>
</tr>
<tr>
<td>Neural 3D Video [44]</td>
<td>SA3D [8]</td>
<td>93.1</td>
<td>94.0</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>93.4</td>
<td>94.3</td>
<td>90.6</td>
<td>92.3</td>
</tr>
</tbody>
</table>

The temporal resolution of the 4D feature volume and make it uniform across all regions of the scene. In Table 3, we demonstrate the performance of this naive temporal sampling strategy using medium (25) or dense (100) temporal resolutions on 100-frame input videos. The results show that increasing the temporal resolution universally does not necessarily yield better performance. In contrast, our proposed method, which utilizes motion-aware temporal sampling, achieves the best results. We surmise that this may be attributed to the model distributing the volume’s capacity sparsely across a large number of time steps, which is not the case for our method.

Motion-aware Spatial Sampling: To validate the effectiveness of our motion-aware spatial sampling strategy, we compare it against other sampling strategies. The first variant involves uniform sampling along the ray, and the second uses a sample prediction network (SPN) like Attal et al. [1] for sampling. These variants sample 128 points on each ray. Results in Table 3 show that our full model employing motion-aware spatial sampling outperforms these variants, perhaps by having a better sense of where to sample more from, derived from its semantic-aware embedding.

SAM embedding: To verify that introducing SAM embedding can improve rendering quality (rather than just enabling segmentation or tracking), we tested a variant of Gear-NeRF without SAM embedding (it instead thresholds loss maps of rendered RGB frames to obtain upshift masks). As shown in Table 3, the absence of SAM embedding for guiding gear assignment reduces the model’s rendering quality.

Number of Gears: We ablate on the numbers of gears. Increasing the number of gears allows for more fine-grained motion-aware spatio-temporal sampling, while increasing the computational cost. As shown in Table 3, a choice of up to 4 gear levels seems optimal, further increasing the number of gears does not result in significant improvements.

6. Conclusions

In this work, we introduced Gear-NeRF, an extension of dynamic NeRFs that leverages semantic information from powerful segmentation models for stratified modeling of dynamic scenes. Our approach leverages a 4D (spatio-temporal) semantic embedding and introduces the concept of “gears” for differentiated modeling of scene regions based on their motion intensity. With determined gear assignments, Gear-NeRF adaptively adjusts its spatio-temporal sampling resolution to improve the photo-realism of rendered views. At the same time, Gear-NeRF provides the new functionality of free-viewpoint object tracking with user prompts as simple as a click. Our empirical studies underscore the effectiveness of Gear-NeRF, showcasing state-of-the-art performance in both rendering quality and object tracking across multiple challenging datasets.
References


[76] Jiaxiang Tang, Xiaokang Chen, Jingbo Wang, and Gang Zeng. Point scene understanding via disentangled instance
mesh reconstruction. European Conference on Computer Vision (ECCV), 2022. 3


[91] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning object bounding boxes for 3d instance segmentation on point clouds, 2019. 3


[102] Shaufeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. In IEEE/CVF International Conference on Computer Vision (ICCV), 2023. 2