Representing Volumetric Videos as Dynamic MLP Maps

Sida Peng* Yunzhi Yan* Qing Shuai Hujun Bao Xiaowei Zhou†
State Key Lab of CAD&CG, Zhejiang University

Abstract

This paper introduces a novel representation of volumetric videos for real-time view synthesis of dynamic scenes. Recent advances in neural scene representations demonstrate their remarkable capability to model and render complex static scenes, but extending them to represent dynamic scenes is not straightforward due to their slow rendering speed or high storage cost. To solve this problem, our key idea is to represent the radiance field of each frame as a set of shallow MLP networks whose parameters are stored in 2D grids, called MLP maps, and dynamically predicted by a 2D CNN decoder shared by all frames. Representing 3D scenes with shallow MLPs significantly improves the rendering speed, while dynamically predicting MLP parameters with a shared 2D CNN instead of explicitly storing them leads to low storage cost. Experiments show that the proposed approach achieves state-of-the-art rendering quality on the NHR and ZJU-MoCap datasets, while being efficient for real-time rendering with a speed of 41.7 fps for 512×512 *images on an RTX 3090 GPU. The code is available at https://zju3dv.github.io/mlp_maps/.*

1. Introduction

Volumetric video captures a dynamic scene in 3D which allows users to watch from arbitrary viewpoints with immersive experience. It is a cornerstone for the next generation media and has many important applications such as video conferencing, sport broadcasting, and remote learning. The same as 2D video, volumetric video should be capable of high-quality and real-time rendering as well as being compressed for efficient storage and transmission. Designing a proper representation for volumetric video to satisfy these requirements remains an open problem.

Traditional image-based rendering methods [2,13,26,77] build free-viewpoint video systems based on dense camera arrays. They record dynamic scenes with many cameras and then synthesize novel views by interpolation from input nearby views. For these methods, the underlying scene representation is the original multi-view video. While there have been many multi-view video coding techniques, the storage and transmission cost is still huge which cannot satisfy real-time video applications. Another line of work [12, 14] utilizes RGB-D sensors to reconstruct textured meshes as the scene representation. With mesh compression techniques, this representation can be very compact and enable streamable volumetric videos, but these methods can only capture humans and objects in constrained environments as reconstructing a high-quality renderable mesh for general dynamic scenes is still a very challenging problem.

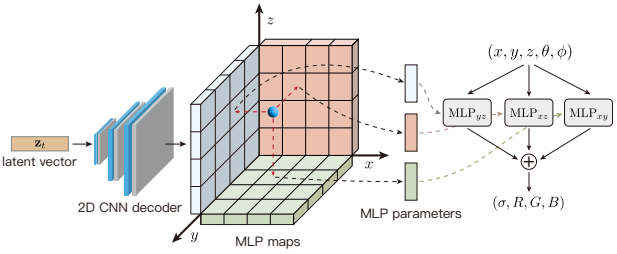

Figure 1. **The basic idea of dynamic MLP maps.** Instead of modeling the volumetric video with a big MLP network [28], we exploit a 2D convolutional neural network to dynamically generate 2D MLP maps at each video frame, where each pixel storing the parameter vector of a small MLP network. This enables us to represent volumetric videos with a set of small MLP networks, thus significantly improving the rendering speed.

Recent advances in neural scene representations [28, 35, 64] provide a promising solution for this problem. They represent 3D scene with neural networks, which can be effectively learned from multi-view images through differentiable renderers. For instance, Neural Volumes [35] represents volumetric videos with a set of RGB-density volumes predicted by 3D CNNs. Since the volume prediction easily consumes large amount of GPU memory, it struggles to model high-resolution 3D scenes. NeRF [39] instead represent 3D scenes with MLP networks regressing density and color for any 3D point, thereby enabling it to synthesize high-resolution images. DyNeRF [28] extends NeRF to model volumetric videos by introducing a temporal latent code as additional input of the MLP network. A major issue of NeRF models is that their rendering is gen-

*Equal contribution. †Corresponding author.

erally quite slow due to the costly network evaluation. To increase the rendering speed, some methods [18, 64, 78] utilize caching techniques to pre-compute a discrete radiance volume. This strategy typically leads to high storage cost, which is acceptable for a static scene, but not scalable to render a volumetric video of dynamic scenes.

In this paper, we propose a novel representation of volumetric video, named dynamic MLP maps, for efficient view synthesis of dynamic scenes. The basic idea is illustrated in Figure 1. Instead of modeling a volumetric video with a single MLP network, we represent each video frame as a set of small MLP networks whose parameters are predicted by a per-scene trained 2D CNN decoder with a per-frame latent code. Specifically, given a multi-view video, we choose a subset of views and feed them into a CNN encoder to obtain a latent code for each frame. Then, a 2D CNN decoder regresses from the latent code to 2D maps, where each pixel in the maps stores a vector of MLP parameters. We call these 2D maps as MLP maps. To model a 3D scene with the MLP maps, we project a query point in 3D space onto the MLP maps and use the corresponding MLP networks to infer its density and color values.

Representing 3D scenes with many small MLP networks decreases the cost of network evaluation and increases the rendering speed. This strategy has been proposed in previous works [51, 52], but their networks need to be stored for each static scene, which easily consumes a lot of storage to represent a dynamic scene. In contrast to them, we use shared 2D CNN encoder and decoder to predict MLP parameters on the fly for each video frame, thereby effectively compressing the storage along the temporal domain. Another advantage of the proposed representation is that MLP maps represent 3D scenes with 2D maps, enabling us to adopt 2D CNNs as the decoder instead of 3D CNNs in Neural Volumes [35]. This strategy leverages the fast inference speed of 2D CNNs and further decreases the memory requirement.

We evaluate our approach on the NHR and ZJU-MoCap datasets, which present dynamic scenes with complex motions. Across all datasets, our approach exhibits state-of-the-art performance in terms of rendering quality and speed, while taking up low storage. Experiments demonstrate that our approach is over 100 times faster than DyNeRF [28].

In summary, this work has the following contributions:

- A novel representation of volumetric video named dynamic MLP maps, which achieves compact representation and fast inference.
- A new pipeline for real-time rendering of dynamic scenes based on dynamic MLP maps.
- State-of-the-art performance in terms of the rendering quality, speed, and storage on the NHR and ZJU-MoCap datasets.

2. Related work

Traditional methods. Early works generally represent 3D scenes with multi-view images [13, 26] or explicit surface models [12, 14]. Light field-based methods [13, 19, 26] build up a dense camera array to capture the target scene and synthesize novel views via light field interpolation techniques. To reduce the number of input camera views, [2, 8, 23, 30, 34, 56, 65, 77] estimate the scene geometry from input views, then use the geometry to warp input views to the target view, and finally blend warped images to the target image. These methods require storing captured RGB images for latter prediction, which could cause high storage costs. Surface-based methods [12, 14, 41, 42, 79] leverage RGB-D sensors to reconstruct textured meshes to represent target scenes. They utilize the depth sensors and multi-view stereo methods [53, 82] to obtain per-view depth images, and employ the depth fusion to obtain the scene geometry. Although these methods are able to create high-quality streamable volumetric videos, the need of depth sensors limits them to work in constrained environments.

Neural scene representations. These methods [39, 62, 69] propose to represent 3D scenes with neural networks, which can be effectively learned from images through differentiable renderers. Some surveys [58, 59, 72] have summarized the neural scene representation methods. NeRF [3, 4, 39] represents continuous volumetric scenes with MLP networks and achieves impressive rendering results. To further improve the performance, some methods [51, 52, 57, 63] decompose the scene into a set of spatial cells, each of which is represented by a NeRF network. Block-NeRF [57] demonstrates that this strategy enables high-quality rendering for large-scale scenes. KiloNeRF [52] represents each subregion with a smaller NeRF network and thus accelerates the rendering process. Other methods speed up the rendering via caching techniques [22, 78] or explicit representations [11, 32]. Explicit representations are also used to increase the rendering quality, such as voxels [21, 32, 76], point clouds [75], and planes [6, 7, 16, 48]. Previous methods [6] have used tri-plane representations, where they leverage tri-plane feature maps to increase the model capacity. Our model differs from them in that we predict tri-plane MLP maps instead of feature maps, and we can model the 3D scene with a single MLP map instead of three maps.

Recent methods [31, 35, 67, 73] investigate the potential of neural 3D representations in representing volumetric videos. To model high-resolution scenes, [15, 17, 33, 44, 47, 49, 50, 74] extend NeRF to represent dynamic scenes. [17, 28, 29, 55, 71] introduce temporal embedding as an additional input of NeRF network to describe dynamic scenes. They typically adopt deep networks to achieve photorealistic rendering, resulting in slow rendering. Another line of works [44, 45, 50, 61, 80] introduce a deformation field which

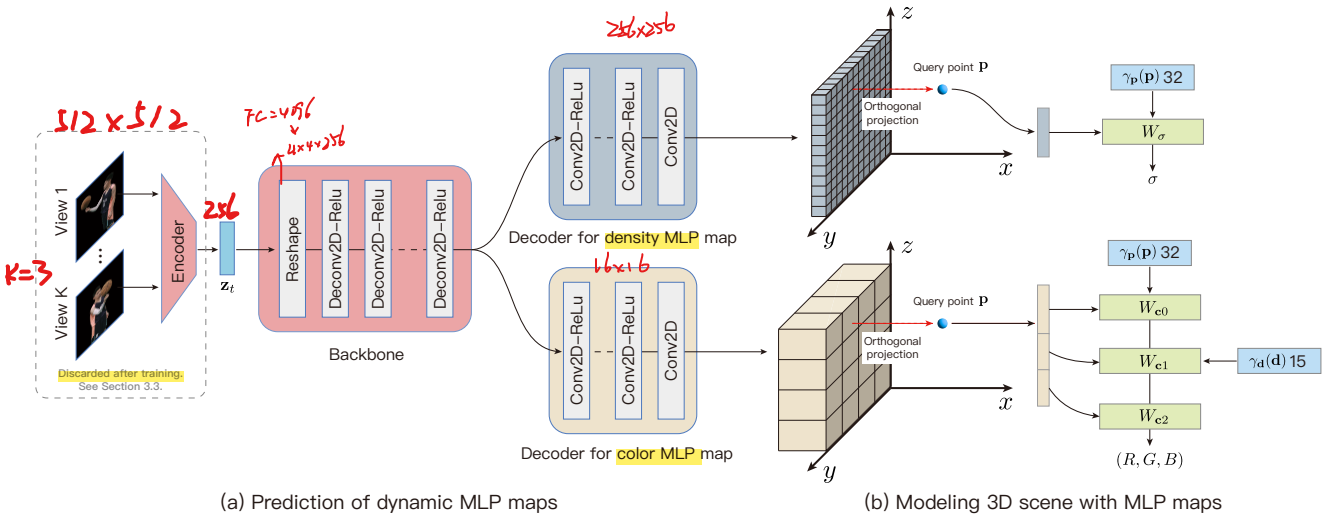

Figure 2. **Illustration of dynamic MLP maps on the YZ plane.** To model the volumetric video, a shared auto-encoder network predicts MLP maps to represent the 3D scene at each video frame. (a) Specifically, for a particular video frame, an encoder network embeds a subset of input views into a latent vector, which is subsequentially processed by a decoder network to output density and color MLP maps. (b) For any 3D query point, it is projected onto the 2D plane defined by the MLP map and retrieve the corresponding network parameters to construct the density and color head. Finally, the resulted MLP network predicts the density and color for this point.

maps points into a canonical space. However, as discussed in [29, 44], deformation is difficult to optimize from images and thus struggle to represent scenes with large motions.

Some methods [18, 35, 36, 64, 68] have explored the task of fast rendering of dynamic scenes. Wang *et al*. [64] utilize a generalized NeRF to build up a PlenOctree [78] for each video frame and compress these PlenOctrees into a Fourier PlenOctree with discrete Fourier transform. To achieve high-quality compression, Fourier transform requires high dimensions of Fourier coefficients, which increases the storage cost. As shown in [64], Fourier PlenOctree models a volumetric video of 60 frames with more than 7 GB storage cost. In contrast, our approach represents dynamic scenes with 2D CNNs, resulting in a compact scene representation for real-time rendering.

Hypernetworks. As discussed in Ha *et al*. [20], a hypernetwork generates the parameters for another network. There have been some works attempting to apply hypernetworks to computer vision tasks, such as semantic perception [9, 60], image generation [24, 25], view synthesis [10, 54], human modeling [66], and inverse rendering [38]. In neural field-based methods [54, 66], hypernetworks are generally utilized to generalize neural representations across a set of scenes. To encode a set of scenes with a single network, SRN [54] defines a set of latent vectors and uses an MLP network to map the latent vector to parameters of an MLP representing a specific scene. Instead of regressing network weights with MLP networks, our approach proposes to use fully convolutional neural network to generates MLP maps, which efficiently produces a set of networks.

3. Proposed approach

Given a multi-view video captured by synchronized and calibrated cameras, our goal is to produce a volumetric video that takes up low disk storage and supports fast rendering. In this paper, we propose a novel representation called dynamic MLP maps for volumetric videos and develop a pipeline for real-time view synthesis of dynamic scenes. In this section, we first describe how to model 3D scenes with MLP maps (Section 3.1). Then, Section 3.2 discusses how to represent volumetric videos with dynamic MLP maps. Finally, we introduce some strategies to speed up the rendering process (Section 3.3).

3.1. Modeling 3D scenes with MLP maps

An MLP map is a 2D grid map with each pixel storing the parameters of an MLP network. To represent 3D scene with MLP maps, we project any 3D point $\mathbf{p}$ to the 2D planes defined by MLP maps for querying the corresponding MLP parameters. In practice, we align MLP maps with the axes of the coordinate frame, and point $\mathbf{p}$ is orthographically projected onto a canonical plane. See Figure 2(b) as an example of MLP maps on the YZ plane. The projected query point is assigned to a parameter vector through spatial binning, which is dynamically loaded to the MLP network. Here we adopt a small NeRF network to predict the density and color for the query point $\mathbf{p}$, which consists of one-layer density head and three-layer color head. The networks encoded by the 2D maps together represent the whole scene. Since each NeRF only describes a fraction of the target scene, our model can reach high-quality rendering

with small MLPs. In contrast to network sets used in previous methods [51, 52, 57, 63], the proposed MLP maps are in the format of 2D planes, enabling us to effectively and efficiently generate MLP parameters with 2D convolutional networks, which will be described latter.

When predicting the density and color for point $\mathbf{p}$, instead of directly passing the spatial coordinate into the network, we embed the input coordinate to a high-dimensional feature vector for better performance. Specifically, we define three multi-level hash tables [40]: $\mathbf{h}_{xy}$, $\mathbf{h}_{xz}$, $\mathbf{h}_{yz}$. Each hash table has a resolution of $L \times T \times F$, where L is the number of hash table's levels, T is the table size, and F is the feature dimension. To embed the input point $\mathbf{p}$, we project it onto three axis-aligned orthogonal planes, transform projected points to feature vectors using three multi-level hash tables, and aggregate the three feature vectors via summation. The embedding process is formulated as:

$$\gamma^h_{\mathbf{p}}(\mathbf{p}) = \eta(x, y, t; \mathbf{h}_{xy}) + \eta(x, z, t; \mathbf{h}_{xz}) + \eta(y, z, t; \mathbf{h}_{yz}), \quad (1)$$

where (x, y, z) is the coordinate of point $\mathbf{p}$, t is the video frame, and η is the encoding function, which obtains the feature vector from the hash table according to the input point (please refer to [40] for more details). In addition, we follow EG3D [6] to predict tri-plane feature maps with a 2D CNN and use them to assign a feature vector $\gamma^t_{\mathbf{p}}(\mathbf{p})$ to each 3D point. We add up the hash table feature $\gamma^h_{\mathbf{p}}(\mathbf{p})$ and the tri-plane feature $\gamma^t_{\mathbf{p}}(\mathbf{p})$ to obtain the final feature vector $\gamma_{\mathbf{p}}(\mathbf{p})$. The density σ and color $\mathbf{c}$ is predicted via:

$$(\sigma, \mathbf{c}) = M(\gamma_{\mathbf{p}}(\mathbf{p}), \gamma_{\mathbf{d}}(\mathbf{d})), \quad (2)$$

where M means the MLP network, and $\gamma_{\mathbf{d}}(\mathbf{d}))$ is the encoded viewing direction. Figure 2(b) visualizes the MLP network. In practice, we implement $\gamma_{\mathbf{d}}$ as a positional encoding function [39], and the 2D CNN shares parameters with the one used to dynamic MLP maps, whose detailed architecture is presented in Section 4.

Orthogonal MLP maps. Experiments demonstrate that modeling scenes with MLP maps defined on one canonical plane struggles to give good rendering quality. A reason is that scene content could be a high-frequency signal function along an axis, which makes the MLP map difficult to fit the scene content. In such case, scene content may have lower frequency along another axes. Inspired by [6, 7, 48], we define three MLP maps on three axis-aligned orthogonal planes. For a query point, we first use the MLP maps to predict densities and colors $\{(\sigma_i, \mathbf{c}_i) \mid i = 1, 2, 3\}$ and then aggregate them via summation. Figure 1 illustrates the idea of summing the outputs of orthogonal signal functions. The experimental results in Section 5.2 show that three MLP maps defined on orthogonal planes perform better than three ones defined on the same planes.

3.2. Volumetric videos as dynamic MLP maps

Based on the MLP map, we are able to use a 2D CNN to represent the volumetric video. Given a multi-view video, we leverage a 2D CNN to dynamically regress 2D maps containing a set of MLP parameters for each video frame, which model the geometry and appearance of 3D scene at corresponding time step. As illustrated in Figure 2(a), the network architecture is implemented as an encoder-decoder, where the encoder regresses latent codes from camera views and the decoder produces MLP maps based on latent codes.

Specifically, for a particular video frame, we select a subset of camera views and utilize 2D CNN encoder to convert them into a latent code following Neural Volumes [35]. The latent code is designed to encode the state of the scene at the video frame and used for the prediction of MLP maps. An alternative way to obtain latent codes is pre-defining learnable feature vectors for each video frame as in [5, 37, 43]. The advantage of learning with an encoder network is that it implicitly shares the information across video frames, enabling joint reconstruction of video sequences rather than just per-frame learning.

Given the encoded latent vector, a 2D CNN decoder is employed to predict MLP maps. Figure 2(a) presents the schematic architecture of 2D CNN decoder. Denote the latent vector as $\mathbf{z} \in \mathbb{R}^D$. We first use a fully-connected network to map $\mathbf{z}$ to a 4096-dimensional feature vector and reshape the resulting vector as a 4×4 feature map with 256 channels. Then, a network with a series of deconvolutional layers upsamples it to a feature map of higher resolution $D \times D$. Based on the feature map, two subsequent convolutional networks are used to predict the density and color MLP maps, respectively. The convolutional network consists of several convolutional layers. By controlling the number and stride of convolutional layers, we control the resolution of the predicted MLP map. Since the density MLP has fewer parameters than the color MLP, we can predict a higher resolution for the density MLP map, which leads to better performance, as demonstrated by experimental results in Section 5.2.

3.3. Accelerating the rendering

Our approach represents 3D scenes with a set of small MLP networks. Since these MLP networks are much smaller than that of DyNeRF [28], our network evaluation takes less time, enabling our approach to be much faster than DyNeRF. To further improve the rendering speed, we introduce two additional strategies.

First, the encoder network can be discarded after training. We use the trained encoder to compute the latent vector for each video frame, and store the resulted latent vectors instead of forwarding the encoder network every time, which save the inference time of the encoder.

Second, the number of network queries is reduced by skipping the empty space. To this end, we compute a low-resolution 3D occupancy volume per video frame, where each volume voxel stores a binary value indicating whether the voxel is occupied. The occupancy volume is extracted from the learned scene representation. We mark a voxel as occupied when its evaluated density is higher than a threshold. Since the occupancy volume has a low resolution and is stored in the binary format, the occupancy volumes of a 300-frame video only take up about 8 MB storage. During inference, the network evaluation is only performed on occupied regions indicated by the occupancy volume. In practice, we will run the density evaluation before the color evaluation to further reduce the number of color evaluations. Specifically, we first evaluate the densities of sampled points within occupied voxels, which are used to calculate the composition weights as defined in the volume rendering [27, 39]. If the weight of a point is higher than a threshold, we then predict its color value, otherwise we skip this point during the volume rendering.

The complete rendering pipeline and the corresponding implementation details are presented in the supplementary material. The code will be released for the reproducibility.

4. Implementation details

Network architecture. Our approach adopts an encoder with seven convolutional layers, which takes three 512×512 images as input and converts them into a 256-dimensional latent vector following [35]. The decoder network consists of a backbone network and two prediction heads. The backbone network has six deconvolutional layers and upsamples the input 4×4 feature map to a 256×256 backbone feature map. We use a convolutional layer to regress from the backbone feature map to a 96-channel feature map and reshape it to three 32-channel planes. The prediction head for density MLP is implemented as a convolutional layer with a stride of 1 and outputs a 256×256 MLP map, each having 32 parameters (32×1). The prediction head for color MLP applies four convolutional layers with a stride of 2 and one convolutional layer with a stride of 1 to the input feature map, resulting in a 16×16 MLP map, each having 2624 parameters ($32 \times 32 + (32 + 15) \times 32 + 32 \times 3$) with ReLU internal activation. For each multi-level hash table, we follow [40] to set $L = 19$, $T = 16$ and $F = 2$.

Storage cost. For an input video of 300 frames, the storage cost of latent vectors, MLP maps decoders, multi-level hash tables and occupancy volumes in our model are 300 KB, 103MB, 131MB and 8MB, respectively. Note that, the encoder network are not stored thanks to our accelerating strategies in Section 3.3.

Training. The networks are optimized to minimize the MSE loss that calculates the difference between rendered and observed images. The Kullback-Leibler divergence loss in [35] is also used for training. More details of loss functions can be found in the supplementary material. During optimization, we set the batch size as eight and train the model on one RTX 3090 GPU, which takes about 16 hours. The learning rate is set as $5e^{-4}$ and $5e^{-3}$ for auto-encoder network and hashtable respectively and decays exponentially by 0.1 every 400 epochs.

Methods	Baseline1	C-NeRF	Ours-Single	Ours
Description	feature tri-planes + single NeRF	feature volumes + single NeRF	feature tri-planes + hash tables + single NeRF	feature tri-planes + hash tables + MLP maps
LPIPS	0.072	0.076	0.065	**0.058**

Table 1. **Analyzing what contributes to the rendering quality**. Metrics are averaged over two scenes from ZJU-MoCap and NHR.

5. Experiments

5.1. Datasets

To evaluate the performance of our approach, we conduct experiments of the ZJU-MoCap [49] and NHR [70] datasets. Both two datasets capture foreground dynamic scenes within bounded regions using multiple synchronized cameras, and the recorded scenes exhibit large motions. On ZJU-MoCap dataset, we uniformly select 11 cameras for training and use the remaining views for evaluation. All video sequences have a length of 300 frames. On NHR dataset, 90 percent of cameras are taken as training views and the other views are kept for evaluation. We select 100 frames from captured videos to reconstructing volumetric videos for each scene. The two datasets provide segmentation masks for foreground dynamic scenes. Based on foreground masks, we produce visual hulls of foreground scenes and calculate axis-aligned bounding boxes enclosing scenes, which are used in our approach.

5.2. Ablation studies

We validate our algorithm's design choices on two scenes from the ZJU-MoCap and NHR datasets. Tables 1 and 2 present the quantitative results.

What contributes to our rendering quality. Table 1 lists the results of ablation studies. "Baseline1" consists of tri-plane feature maps and a single shared MLP for all video frames. The MLP has the same network as the original NeRF. "Ours-Single" is the model in Table 2 (5). Table 1 indicates two components important to our rendering quality: 1) Hash tables (Ours-Single vs. C-NeRF [67]). 2) MLP maps (Ours vs. Ours-Single). Using MLP maps allows each MLP to represent a small region, in contrast to the single NeRF modeling the whole volumetric video.

	LPIPS ↓	Rendering time (milliseconds) ↓	Storage (MB) ↓
(1) Color map size 1×1	0.069	22	489
(2) Color map size 4×4	0.065	23	324
(3) Color map size 32×32	0.060	27	208
(4) Density map Size 16×16	0.064	25	250
(5) Single shared MLP	0.065	90	133
(6) Single shared MLP w/o ESS	0.065	1345	**125**
(7) XY MLP map	0.072	**19**	206
(8) w/o orthogonal signal	0.067	24	242
(9) w/o ESS	**0.058**	144	237
(10) Complete model	**0.058**	24	242

Table 2. **Ablation studies of our model.** Metrics are averaged over two scenes from the ZJU-MoCap and NHR datasets. We test the time of rendering a 512×512 image on one RTX 3090 GPU. See Section 5.2 for detailed descriptions.

Figure 3. **Qualitative results of models with different color MLP map resolutions** on the ZJU-MoCap dataset.

The resolution of MLP maps. Experimental results in rows 1, 2, and 4 of Table 2 show that decreasing the resolution undermines the rendering performance. Note that, the storage cost increases when the resolution decreases. This is because that we add more convolutional layers for the downsampling, as described in Section 4.

Results of row 3 show that increasing the resolution does not necessarily improves the rendering quality. A plausible explanation is that the increased resolution make the model more difficult to train for two factors. First, the higher resolution of MLP map means more parameters to be optimized. Second, increasing the resolution need to decrease the number of query points fed into each color MLP during each training iteration due to memory limit, thereby decreasing the batch size. Figure 3 shows some visual results.

It is interesting to analyze why the high-resolution density map is trainable while the color map is not. The reason is that density MLP has 32 parameters, and color MLP has 2624 parameters, which means that 256×256 density MLP map has similar number of parameters to 28×28 color MLP map. In addition, the scene geometry generally has lower frequency than the appearance. Therefore, the high-resolution density MLP maps are easier to train.

The results in rows 1-4 of Table 2 also indicate that the resolution of MLP maps does not affect the rendering speed much. This can be attributed to that the number of MLP evaluations is not related to the map resolution.

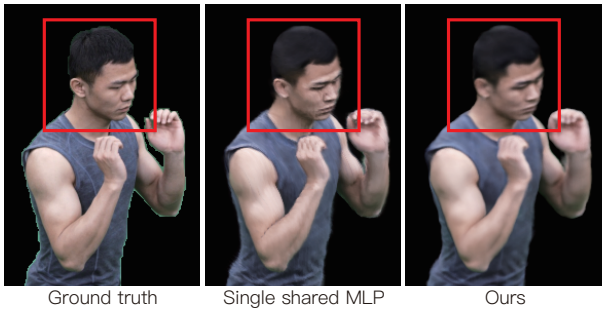

Figure 4. **Comparisons between dynamic MLP maps and a single shared MLP** on the NHR dataset.

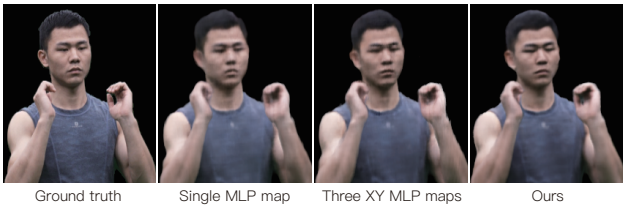

Figure 5. **Ablation studies on the orthogonal MLP maps.** The results indicate that orthogonal MLP maps improves the quality.

Our model with a shared MLP. Rows 5-6 of Table 2 means that we replace the dynamic MLP maps with a single shared MLP for all video frames. This MLP network is implemented as the original NeRF network [39] except that the input is the high-dimensional feature vector $\gamma_{\mathbf{p}}(\mathbf{p})$ in Section 3.2. Our complete model has better rendering quality and faster rendering speed than the model with a single shared MLP network. The results of row 6 indicate that, with the empty space skipping, the rendering speed of model "Single shared MLP" increases much. But it is still much slower than our complete model.

The influence of using orthogonal MLP maps. Row "XY MLP map" in Table 2 means that we represent the scene with a single dynamic MLP map defined on the XY plane. The results show the rendering quality degrades much. Row "w/o orthogonal signal" indicates that we adopts three dynamic MLP maps, which are all defined on the XY plane. The resulting model also shows a bad rendering quality.

Analysis of our rendering time. Row "w/o ESS" in Table 2 investigates the effect of the empty space skipping. The results show that this technique improves the rendering speed 6 times while only adding 5 MB on the storage cost. Here we analyze the running time of each model component. The inference time of MLP maps decoder and querying multi-level hash tables are about 4ms and 7ms, respectively. Our color and density MLP maps take only 12ms in total, in comparison with over 80ms taken by a single shared MLP with empty space skipping.

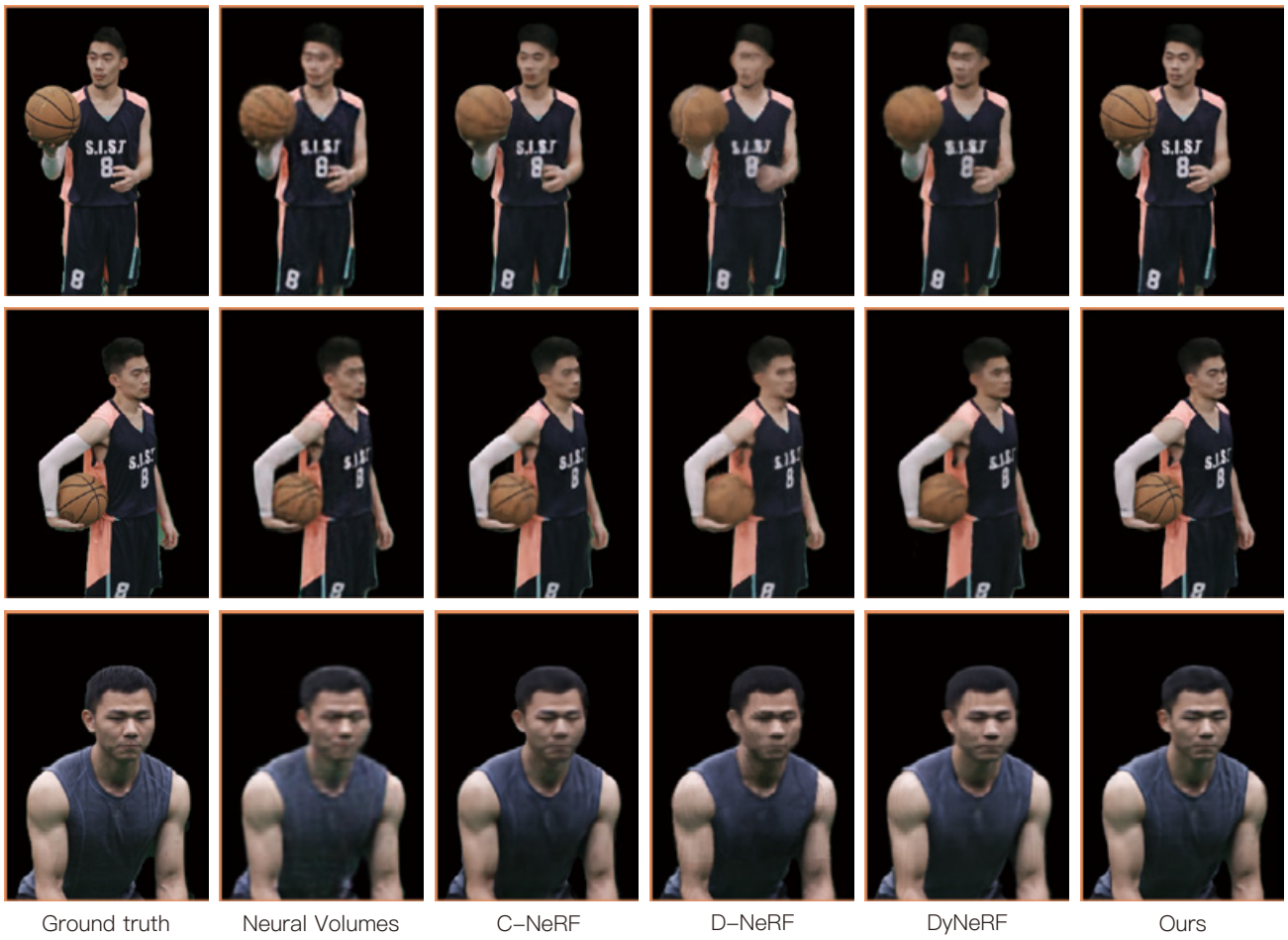

Figure 6. **Qualitative comparisons on the NHR dataset.** Our approach significantly outperforms other methods. The results in the first rows indicates that we can synthesize detailed texture of basketballs, while the rendering results of other methods are blurry.

5.3. Comparisons with the state-of-the-art methods

Baseline methods. We compare our approach with four recent methods. (1) Neural Volumes (NV) [35] uses a 3D deconvolutional network to produce RGB-Alpha volumes as the volumetric video. (2) C-NeRF [67] is a follow-up work of Neural Volumes, which adopts a 3D CNN to produce feature volumes and appends a NeRF-like MLP to predict radiance fields. (3) D-NeRF [50] decomposes the dynamic scene into a static scene in the canonical space and a deformation field. (4) DyNeRF [28] represents the volumetric video with a NeRF-like MLP, which takes spatial coordinate and temporal embedding as input and predict the radiance field at a particular video frame. Since C-NeRF and DyNeRF do not release the code, we re-implement them for the comparison. We do not compare with Neural Body [49] and MVP [36], because their algorithms take tracked meshes as input, making the comparison unfair.

Tables 3, 4 list the comparison of our method with [28,35,50,67] in terms of rendering quality, rendering speed and storage. We adopt PSNR, SSIM and LPIPS [81] as metrics to evaluate rendering quality. For all metrics, our method achieves the best performance among all the methods. Moreover, our model renders two magnitude faster than C-NeRF, D-NeRF and DyNeRF while maintaining reasonable storage cost. Although our approach is only twice as fast as Neural Volumes due to its explicit scene represen-

	PSNR ↑	SSIM ↑	LPIPS ↓	Rendering time (milliseconds) ↓	Storage (MB) ↓
NV [35]	30.86	0.941	0.130	73	658
C-NeRF [67]	31.32	0.949	0.102	1969	1019
D-NeRF [50]	29.25	0.920	0.150	2303	**4**
DyNeRF [28]	30.87	0.943	0.118	5195	12
Ours	**32.20**	**0.953**	**0.080**	**33**	239

Table 3. **Quantitative results on the NHR dataset.** Metrics are averaged over all scenes. This dataset includes 512×612 images and 384×512 images.

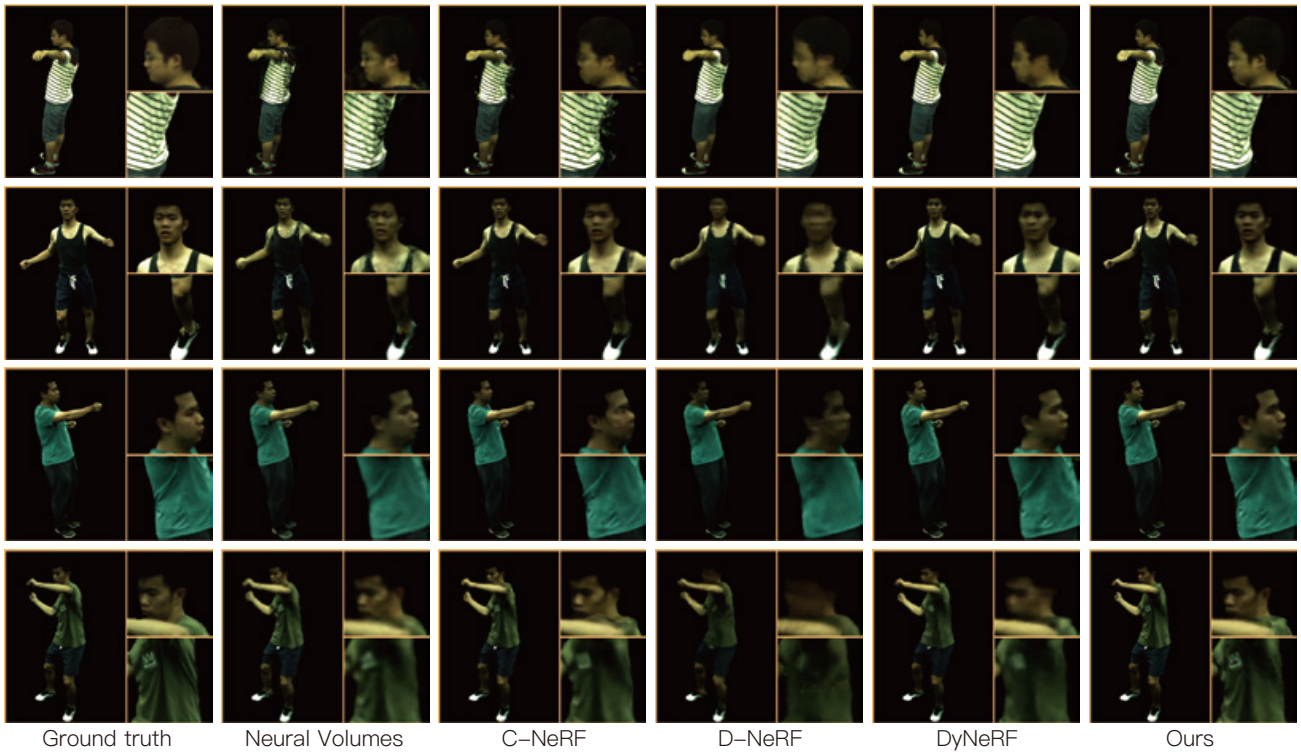

Figure 7. **Qualitative comparisons in the ZJU-MoCap dataset.** Our proposed model can generate sharp details, as shown by the results of last two rows. In contrast, other methods tend to render smooth images.

	PSNR ↑	SSIM ↑	LPIPS ↓	Rendering time (milliseconds) ↓	Storage (MB) ↓
NV [35]	28.12	0.934	0.131	49	658
C-NeRF [67]	29.79	0.959	0.077	1313	1019
D-NeRF [50]	27.08	0.922	0.139	1534	**4**
DyNeRF [28]	29.88	0.959	0.087	3452	12
Ours	**30.17**	**0.963**	**0.068**	**24**	245

Table 4. **Comparison on the ZJU-MoCap dataset.** Metrics are averaged over all scenes. The image resolution is 512×512.

tation, our rendering accuracy is significantly better.

Figures 6, 7 show the qualitative results of our method and baselines. Limited by low volume resolution, Neural Volumes gives blurry results. D-NeRF has similar problems as deformation fields are difficult to learn when complex scene motions exist. C-NeRF and DyNeRF tend to lose details in high frequency regions of the image because of their single shared MLP model structures. In contrast, our method generates photorealistic novel view results.

6. Conclusion

We proposed a novel neural representation named dynamic MLP maps for volumetric videos. The key idea is utilizing a shared 2D CNN to predict 2D MLP maps of each video frame, which store the parameters of MLP networks at pixels. To model 3D scene with MLP maps, our approach regresses the density and color of any 3D point by projecting it onto the 2D planes defined by MLP maps and queries the corresponding the MLP networks for the prediction. Experiments demonstrated that our approach achieves competitive rendering quality and low storage costs, while being efficient for real-time rendering.

Limitations. 1) Common videos are more than a few minutes. However, this work only deals with videos of 100 to 300 frames, which are relatively short, thus limiting the applications. How to model a long volumetric video remains an interesting problem. 2) Since we use multi-level hash tables, the storage cost of our model will increase as the video length increases. 3) The proposed representaion requires dense camera views for training, similar to [28, 35, 67]. Recovering free-viewpoint videos from sparse-view videos is also an open problem.

Acknowledgements. This work was supported by the Key Research Project of Zhejiang Lab (No. K2022PG1BB01), NSFC (No. 62172364), and Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

References

[1] Ahmad Abdelfattah, Azzam Haidar, Stanimire Tomov, and Jack Dongarra. Novel hpc techniques to batch execution of many variable size blas computations on gpus. In *International Conference on Supercomputing*, 2017. 12

[2] Aayush Bansal, Minh Vo, Yaser Sheikh, Deva Ramanan, and Srinivasa Narasimhan. 4d visualization of dynamic events from unconstrained multi-view videos. In *CVPR*, 2020. 1, 2

[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 *ICCV*, 2021. 2

[4] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 2

[5] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776*, 2017. 4

[6] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*, 2022. 2, 4

[7] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *arXiv preprint arXiv:2203.09517*, 2022. 2, 4

[8] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, 2021. 2

[9] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *CVPR*, 2020. 3

[10] Yinbo Chen and Xiaolong Wang. Transformers as meta-learners for implicit neural representations. In *ECCV*, 2022. 3

[11] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. *arXiv preprint arXiv:2208.00277*, 2022. 2

[12] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. High-quality streamable free-viewpoint video. *ACM TOG*, 2015. 1, 2

[13] Abe Davis, Marc Levoy, and Fredo Durand. Unstructured light fields. In *CGF*. Wiley Online Library, 2012. 1, 2

[14] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. Fusion4d: Real-time performance capture of challenging scenes. *ACM TOG*, 2016. 1, 2

[15] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *ICCV*, 2021. 2

[16] Qiao Feng, Yebin Liu, Yu-Kun Lai, Jingyu Yang, and Kun Li. Fof: Learning fourier occupancy field for monocular real-time human reconstruction. In *NeurIPS*, 2022. 2

[17] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *ICCV*, 2021. 2

[18] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *ICCV*, 2021. 2, 3

[19] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *SIGGRAPH*, 1996. 2

[20] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 3

[21] Zekun Hao, Arun Mallya, Serge Belongie, and Ming-Yu Liu. Gancraft: Unsupervised 3d neural rendering of minecraft worlds. In *ICCV*, 2021. 2

[22] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *ICCV*, 2021. 2

[23] Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. Geonerf: Generalizing nerf with geometry priors. In *CVPR*, 2022. 2

[24] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 3

[25] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020. 3

[26] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH*, 1996. 1, 2

[27] Ruilong Li, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: A general nerf acceleration toolbox. *arXiv preprint arXiv:2210.04847*, 2022. 5, 12

[28] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *CVPR*, 2022. 1, 2, 4, 7, 8

[29] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 2, 3

[30] Haotong Lin, Sida Peng, Zhen Xu, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields with learned depth-guided sampling. *arXiv preprint arXiv:2112.01517*, 2021. 2

[31] Kai-En Lin, Lei Xiao, Feng Liu, Guowei Yang, and Ravi Ramamoorthi. Deep 3d mask volume for view synthesis of dynamic scenes. In *ICCV*, 2021. 2

[32] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020. 2

[33] Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural actor: Neural free-view synthesis of human actors with pose control. *ACM TOG*, 2021. 2

[34] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Christian Theobalt, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based rendering. In *CVPR*, 2022. 2

[35] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019. 1, 2, 3, 4, 5, 7, 8, 12, 13

[36] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM TOG*, 2021. 3, 7

[37] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, 2021. 4

[38] Maxim Maximov, Laura Leal-Taixé, Mario Fritz, and Tobias Ritschel. Deep appearance maps. In *ICCV*, 2019. 3

[39] 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. 1, 2, 4, 5, 6, 12

[40] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 2022. 4, 5

[41] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *CVPR*, 2015. 2

[42] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L Davidson, Sameh Khamis, Mingsong Dou, et al. Holoportation: Virtual 3d teleportation in real-time. In *UIST*, 2016. 2

[43] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 4

[44] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *ICCV*, 2021. 2, 3

[45] 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. *arXiv preprint arXiv:2106.13228*, 2021. 2

[46] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019. 12

[47] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Animatable neural radiance fields for modeling dynamic human bodies. In *ICCV*, 2021. 2

[48] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020. 2, 4

[49] 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. In *CVPR*, 2021. 2, 5, 7

[50] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021. 2, 7, 8

[51] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. In *CVPR*, 2021. 2, 4

[52] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *ICCV*, 2021. 2, 4, 12

[53] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 2

[54] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *NeurIPS*, volume 32, 2019. 3

[55] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *arXiv preprint arXiv:2210.15947*, 2022. 2

[56] Xin Suo, Yuheng Jiang, Pei Lin, Yingliang Zhang, Minye Wu, Kaiwen Guo, and Lan Xu. Neuralhumanfvv: Real-time neural volumetric human performance rendering using rgb cameras. In *CVPR*, 2021. 2

[57] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *CVPR*, 2022. 2, 4

[58] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. In *CGF*, 2020. 2

[59] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, W Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. In *CGF*, 2022. 2

[60] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *ECCV*, 2020. 3

[61] 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 *ICCV*, 2021. 2

[62] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *CVPR*, 2020. 2

[63] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *CVPR*, 2022. 2, 4

[64] Liao Wang, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, Yingliang Zhang, Minye Wu, Jingyi Yu, and Lan Xu. Fourier plenoctrees for dynamic radiance field rendering in real-time. In *CVPR*, 2022. 1, 2, 3

[65] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. 2

[66] Shaofei Wang, Marko Mihajlovic, Qianli Ma, Andreas Geiger, and Siyu Tang. Metaavatar: Learning animatable clothed human models from few depth images. In *NeurIPS*, volume 34, 2021. 3

[67] Ziyan Wang, Timur Bagautdinov, Stephen Lombardi, Tomas Simon, Jason Saragih, Jessica Hodgins, and Michael Zollhofer. Learning compositional radiance fields of dynamic human heads. In *CVPR*, 2021. 2, 5, 7, 8, 12, 13

[68] Ziyan Wang, Giljoo Nam, Tuur Stuyck, Stephen Lombardi, Michael Zollhöfer, Jessica Hodgins, and Christoph Lassner. Hvh: Learning a hybrid neural volumetric representation for dynamic hair performance capture. In *CVPR*, 2022. 3

[69] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *CVPR*, 2021. 2

[70] Minye Wu, Yuehao Wang, Qiang Hu, and Jingyi Yu. Multi-view neural human rendering. In *CVPR*, 2020. 5

[71] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, 2021. 2

[72] 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. In *CGF*, 2022. 2

[73] Wenpeng Xing and Jie Chen. Temporal-mpi: Enabling multi-plane images for dynamic scene modelling via temporal basis learning. In *ECCV*, 2022. 2

[74] Hongyi Xu, Thiemo Alldieck, and Cristian Sminchisescu. H-nerf: Neural radiance fields for rendering and temporal reconstruction of humans in motion. In *NeurIPS*, 2021. 2

[75] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *CVPR*, 2022. 2

[76] Yinghao Xu, Sida Peng, Ceyuan Yang, Yujun Shen, and Bolei Zhou. 3d-aware image synthesis via learning structural and textural representations. In *CVPR*, 2022. 2

[77] 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 *CVPR*, 2020. 1, 2

[78] 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. 2, 3

[79] Tao Yu, Zerong Zheng, Kaiwen Guo, Jianhui Zhao, Qionghai Dai, Hao Li, Gerard Pons-Moll, and Yebin Liu. Doublefusion: Real-time capture of human performances with inner body shapes from a single depth sensor. In *CVPR*, 2018. 2

[80] Jiakai Zhang, Xinhang Liu, Xinyi Ye, Fuqiang Zhao, Yanshun Zhang, Minye Wu, Yingliang Zhang, Lan Xu, and Jingyi Yu. Editable free-viewpoint video using a layered neural representation. *ACM TOG*, 2021. 2

[81] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 7

[82] Enliang Zheng, Enrique Dunn, Vladimir Jojic, and Jan-Michael Frahm. Patchmatch based joint view selection and depthmap estimation. In *CVPR*, 2014. 2

Supplementary Material for Representing Volumetric Videos as Dynamic MLP Maps

1. More Implementation details

Empty space skipping. Resolution of the occupancy volume per video frame is set to $24 \times 24 \times 48$. During evaluation, we divide each volume cell into a subgrid of $5 \times 5 \times 5$ and calculate the density of each subgrid point following [52]. A volume cell is considered as occupied if there exists a subgrid point inside with density above threshold τ_1 ($\tau_1 = 5$ in all our experiments).

Rendering pipeline. During training, we uniformly sample 64 query locations along each camera ray within the scene bound. We do not use hierarchical sampling in [39]. For each query point, we first project it onto three axis-aligned orthogonal MLP maps which is predicted by the 2D CNN decoder to get the corresponding MLP parameters. Then, we embed the query point to high-dimensional feature vector using the multi-level hash tables and feed it into the MLP network to predict the color and density. To efficiently evaluate multiple MLP networks, we develop a custom PyTorch layer [46] based on the library MAGMA [1], following KiloNeRF [52]. We obtain the final result by combining the prediction from each MLP map via elementwise summation. Pixel color is rendered using the differentiable volume rendering following [39], which is defined as:

$$\begin{aligned} \tilde{C}(\mathbf{r}) &= \sum_{i=1}^{M} \omega_i \mathbf{c}_i, \\ \omega_i &= T_i(1 - \exp(-\sigma_i \delta_i)), \\ T_i &= \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j) \end{aligned} \tag{3}$$

where c_i, σ_i denotes color and density predicted by MLP maps, T_i is the accumulated transmittance, ω_i is the composition weight, and δ_i denotes the ray step size.

During inference, we set the sampling step size to be $\frac{1}{256}^{th}$ the diagonal distance of the scene bound. Sampling step size denotes distance between adjacent sampling locations on a ray. For each ray, we sample points only in occupied voxel with the pre-computed occupancy volume. To further speed up the rendering process, we first obtain the density values and compute the accumulated transmittance and composition weight. Sampling points are filtered for the color evaluations with threshold τ_2 on composition weight ($\tau_2 = 1e^{-3}$ in all our experiments), which means we ignore a query point if its weight is below τ_2. Colors and densities are predicted in the same way as in the training stage, and the final predicted color is computed via volume rendering as Eq. (3). The rendering pipeline is implemented based on [27].

Loss function. The loss function is defined as:

$$L = L_c + \lambda_{KL} L_{KL}. \tag{4}$$

Here L_c is the error between rendered pixel color $\tilde{C}(\mathbf{r})$ and observed pixel color $C(\mathbf{r})$:

$$L_c = \sum_{\mathbf{r} \in \mathcal{R}} ||\tilde{C}(\mathbf{r}) - C(\mathbf{r})||_2^2, \tag{5}$$

where $\mathcal{R}$ means the set of camera rays. The Kullback-Leibler divergence loss L_{KL} follows in the formation in [35]. We set $\lambda_{KL} = 1e^{-6}$ in all our experiments. If the target scenes contain only foreground objects, we also add the mask loss to help the training, which measures the error between ground truth foreground mask $\tilde{M}(\mathbf{r})$ and rendered image opacities $M(\mathbf{r})$:

$$L_m = \sum_{\mathbf{r} \in \mathcal{R}} ||\tilde{M}(\mathbf{r}) - M(\mathbf{r})||_2^2. \tag{6}$$

The weight of the mask loss is set as 0.1 in experiments.

2. Results on the data of Neural Volumes

Neural Volumes [35] has released a video sequence in their paper, which records the floating dry ice. We found that there is a moving human in the background of some camera views, which is different from the data presented in the paper of Neural Volumes. Since Neural Volumes does not describe the training frames and camera views, we selected a 100-frame video clip, ranging from frame 16120 to 16417. We test our model on camera 400015 and train on the remaining cameras except 400055 and 400070.

The results in Table 5 demonstrate that our method outperforms Neural Volumes and C-NeRF [67]. Figure 8 presents the qualitative comparisons.

3. Discussion

Dynamic MLP maps vs. per-frame KiloNeRF. An alternative way to represent real-time volumetric video is storing a sequence of per-frame KiloNeRF [52]. This scheme makes the storage and training time increase linearly with the number of video frames. For example, given a 300-frame video, we need to store 300 KiloNeRF models. Consider that a KiloNeRF model requires 20 training hours and

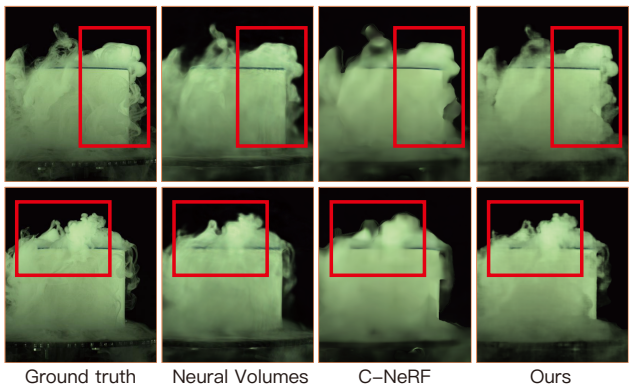

Figure 8. **Qualitative results on the data of Neural Volumes.** We render more photorealistic images than baseline methods.

	PSNR ↑	SSIM ↑	LPIPS ↓
NV [35]	32.01	0.951	0.218
C-NeRF [67]	31.84	**0.953**	0.227
Ours	**32.25**	0.948	**0.198**

Table 5. **Comparison on the data of Neural Volumes.** Our method outperforms Neural Volumes and C-NeRF quantitatively.

30 MB. 300 KiloNeRF models would take 6000 hours for training and consume 9 GB in storage. In contrast, our method requires 16 hours for training and takes up about 240 MB in storage.

Motivation of using two parameter sets. We represent the volumetric video with two set of parameters: (1) hash tables and (2) a CNN that generates the MLP maps. The motivation of using a hybrid of hash tables and MLP maps to achieve both high quality and efficiency. 1) Why MLP maps: Only using hash tables needs a relatively large MLP to achieve high quality, which will be slow due to the costly network evaluation, as suggested by the rendering speed of DyNeRF and C-NeRF. Using MLP maps increases the rendering speed with small MLPs. 2) Why hash tables: compared to MLP maps alone, hash tables improve the rendering quality, as shown in the section of ablation studies. We will make the motivation clearer in the revised paper.

Why use MLP maps instead of feature maps. We use MLP maps to improve rendering speed. Table 1 in the section of ablation studies indicates that using feature maps instead of MLP maps needs a large MLP to achieve high quality, which will be slow. Feature maps can work together with MLP maps, but it does not perform better than hash tables, as demonstrated by the results of ablation studies.

Contribution to rendering speed. When the ESS is not used, the rows 6 and 9 in Table 1 of ablation studies shows that MLP maps make the rendering 9x faster. When the ESS is used, the results of ablation studies indicate that MLP maps make the rendering 4x faster. ESS is a super effective technique. We believe that achieving another 4x speedup on top of it by a novel representation design is non-straightforward, beneficial to the community, and critical for real-time applications.

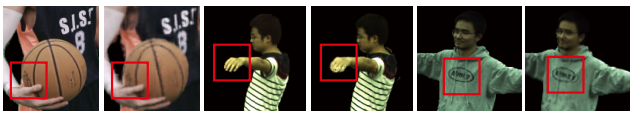

Figure 9. **Failure cases.** Our method has difficulty in rendering very detailed content, such as the text and fingers.

Failure cases. Our method struggles to render very detailed content, such as the text and fingers. Figure 9 presents some qualitative results.

4. Societal impact

The volumetric videos could be misused for recording and spreading some moments of real people without permission, which poses a threat to the privacy. We strongly oppose such usage of our technique.

5. Detailed results

Tables 6 and 7 present the per-scene comparison. The results demonstrate that our proposed approach significantly outperforms baseline methods.

	sport1	sport2	sport3	basketball
Metric	PSNR↑			
Neural Volumes	31.76	31.48	31.04	29.17
C-NeRF	31.81	32.12	31.99	**29.35**
D-NeRF	30.12	30.18	29.66	27.02
DyNeRF	31.76	32.43	31.33	27.97
Ours	**32.92**	**33.19**	**33.59**	29.11
Metric	SSIM↑			
Neural Volumes	0.951	0.933	0.940	0.938
C-NeRF	0.954	0.950	0.950	0.942
D-NeRF	0.934	0.917	0.914	0.914
DyNeRF	0.954	0.945	0.944	0.929
Ours	**0.959**	**0.954**	**0.956**	**0.943**
Metric	LPIPS ↓			
Neural Volumes	0.106	0.143	0.131	0.141
C-NeRF	0.085	0.107	0.097	0.118
D-NeRF	0.111	0.169	0.155	0.163
DyNeRF	0.095	0.119	0.114	0.142
Ours	**0.067**	**0.084**	**0.076**	**0.094**

Table 6. **Quantitative comparisons on the NHR dataset.**

	313	315	377	386	387	390	392	393	394
Metric	PSNR↑								
Neural Volumes	30.23	26.92	26.90	30.15	25.84	28.06	28.76	27.95	28.28
C-NeRF	31.84	29.03	28.92	30.75	27.52	29.81	30.82	29.62	29.81
D-NeRF	27.48	26.68	26.20	28.78	25.53	28.10	27.37	26.14	27.47
DyNeRF	31.50	**30.29**	28.92	30.88	**27.90**	**30.14**	30.09	29.28	29.88
Ours	**32.15**	29.94	**29.40**	**31.05**	27.89	30.10	**31.06**	**29.78**	**30.15**
Metric	SSIM↑								
Neural Volumes	0.949	0.947	0.921	0.949	0.910	0.924	0.939	0.936	0.932
C-NeRF	0.973	0.968	0.958	0.958	0.952	0.957	0.959	0.955	0.952
D-NeRF	0.927	0.939	0.920	0.937	0.918	0.933	0.911	0.897	0.915
DyNeRF	0.970	0.976	0.960	**0.960**	**0.953**	**0.959**	0.953	0.952	0.952
Ours	**0.976**	**0.977**	**0.963**	**0.960**	**0.953**	**0.959**	**0.962**	**0.958**	**0.957**
Metric	LPIPS ↓								
Neural Volumes	0.103	0.114	0.147	0.122	0.169	0.149	0.127	0.128	0.124
C-NeRF	0.057	0.076	0.079	**0.072**	**0.080**	0.071	0.077	0.086	0.090
D-NeRF	0.122	0.106	0.150	0.132	0.140	0.119	0.166	0.162	0.150
DyNeRF	0.070	0.061	0.083	0.082	0.094	0.083	0.113	0.102	0.099
Ours	**0.049**	**0.047**	**0.069**	**0.072**	0.081	**0.068**	**0.074**	**0.080**	**0.072**

Table 7. **Quantitative comparisons on the ZJU-MoCap dataset.**