Supplementary Material for 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

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 [6]. 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 [4]. 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 [5] based on the library MAGMA [1], following KiloNeRF [6]. 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 [4], which is defined as:

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

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. (1). The rendering pipeline is implemented based on [2].

Loss function. The loss function is defined as:

$$L = L_c + \lambda_{KL} L_{KL}. \tag{2}$$

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

$$L_c = \sum_{\mathbf{r} \in \mathcal{R}} ||\tilde{C}(\mathbf{r}) - C(\mathbf{r})||_2^2, \tag{3}$$

where $\mathcal{R}$ means the set of camera rays. The Kullback-Leibler divergence loss L_{KL} follows in the formation in [3]. 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 $M(\mathbf{r})$:

$$L_m = \sum_{\mathbf{r} \in \mathcal{R}} ||\tilde{M}(\mathbf{r}) - M(\mathbf{r})||_2^2. \tag{4}$$

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

2. Results on the data of Neural Volumes

Neural Volumes [3] 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

*Equal contribution. †Corresponding author.

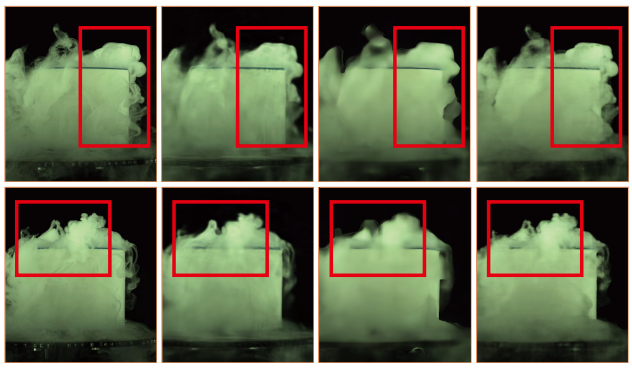

Figure 1. **Qualitative results on the data of Neural Volumes.** We render more photorealistic images than baseline methods.

	PSNR ↑	SSIM ↑	LPIPS ↓
NV [3]	32.01	0.951	0.218
C-NeRF [7]	31.84	**0.953**	0.227
Ours	**32.25**	0.948	**0.198**

Table 1. **Comparison on the data of Neural Volumes.** Our method outperforms Neural Volumes and C-NeRF quantitatively.

16417. We test our model on camera 400015 and train on the remaining cameras except 400055 and 400070.

The results in Table 1 demonstrate that our method outperforms Neural Volumes and C-NeRF [7]. Figure 1 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 [6]. 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 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.

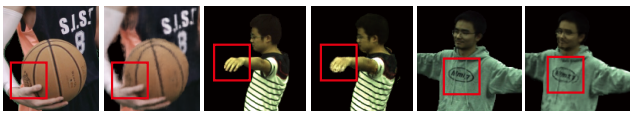

Figure 2. **Failure cases.** Our method has difficulty in rendering very detailed content, such as the text and fingers.

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.

Failure cases. Our method struggles to render very detailed content, such as the text and fingers. Figure 2 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 2 and 3 present the per-scene comparison. The results demonstrate that our proposed approach significantly outperforms baseline methods.

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. 1

[2] Ruilong Li, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: A general nerf acceleration toolbox. *arXiv preprint arXiv:2210.04847*, 2022. 1

[3] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural vol-

	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 2. **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 3. **Quantitative comparisons on the ZJU-MoCap dataset.**

umes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751*, 2019. 1, 2

[4] 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

[5] 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. 1

[6] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *ICCV*, 2021. 1, 2

[7] 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