VideoRF: Rendering Dynamic Radiance Fields as 2D Feature Video Streams

Supplementary Material

8. Training Details for VideoRF

8.1. Coarse Stage Pre-training and Baking

Given a long-duration multi-view sequence, we initially adopt the approach from DVGO [55] to generate an explicit density volume grid $\mathbf{V}_\sigma$ and color feature grids $\mathbf{V}_c$ representation for each frame t. Following ReRF [65], we employ a global MLP Φ_c during this coarse stage training. This MLP comprises a single hidden layer with 129 channels, and we set the color feature dimension at $h = 12$. Throughout the training, we incrementally upscale the volume grid, from $(125 \times 125 \times 125) \rightarrow (150 \times 150 \times 150) \rightarrow (200 \times 200 \times 200) \rightarrow (250 \times 250 \times 250)$, after reaching the training step 2000, 4000 and 6000, respectively. For loss calculation, we utilize both the photometric MSE loss and the total variation loss on $\mathbf{V}_\sigma$, expressed as:

$$\mathcal{L}_{\text{rgb}_{\text{coarse}}} = \sum_{\mathbf{r}\in\mathcal{R}} \|\mathbf{c}(\mathbf{r}) - \hat{\mathbf{c}}(\mathbf{r})\|^2, \tag{9}$$

$$\mathcal{L}_{\text{TV}_{\text{coarse}}} = \frac{1}{|\mathbf{V}_\sigma|} \sum_{\mathbf{v}\in\mathbf{V}_\sigma} \sqrt{\Delta_x^2\mathbf{v} + \Delta_y^2\mathbf{v} + \Delta_z^2\mathbf{v}}, \tag{10}$$

$$\mathcal{L}_{\text{coarse}} = \mathcal{L}_{\text{rgb}_{\text{coarse}}} + \lambda_{\text{TV}}\mathcal{L}_{\text{TV}_{\text{coarse}}}, \tag{11}$$

where $\lambda_{\text{TV}} = 0.000016$. Here, $\mathcal{R}$ represents the set of training pixel rays, with $\mathbf{c}(\mathbf{r})$ and $\hat{\mathbf{c}}(\mathbf{r})$ denoting the actual and predicted colors of a ray $\mathbf{r}$, respectively. $\Delta^2_{x,y,z}\mathbf{v}$ signifies the squared difference in the voxel's density value. Notably, the total variation loss is activated only during the training iterations from 1000 to 12000. For optimization, we utilize the Adam optimizer for training 16000 iterations with a batch size of 10192 rays. The learning rates for $\mathbf{V}_\sigma$, $\mathbf{V}_c$ and global MLP are 0.1, 0.11 and 0.002, respectively.

Once we obtain the density grid $\mathbf{V}^t_\sigma$ for each frame t in the coarse training phase, we generate a per-frame occupancy grid $\mathbf{O}^t$ by retaining voxels with a density above the threshold $\gamma = 0.003$. During our adaptive grouping stage, we set the pixel limit to $\theta = 512 \times 512$.

8.2. Fine-grained Sequential Training

After creating the mapping tables, we proceed to fine-grained sequential training within each group. At this stage, we also introduce a global tiny MLP Φ_f designed for efficient rendering on mobile platforms. This minimal MLP Φ_f consists of only one hidden layer with 16 channels, and we maintain the color feature dimension h at 12. Similar to the coarse stage, we progressively upscale the volume grid during training, moving from $(125 \times 125 \times 125)$ to $(150 \times 150 \times 150)$, then to $(200 \times 200 \times 200)$, and finally to $(250 \times 250 \times 250)$, corresponding to the training steps at 2000, 4000, and 6000, respectively. For loss calculations, we employ both the photometric MSE loss and the total variation loss on the density volume $\mathbf{V}_\sigma$, as well as spatial consistency loss and temporal consistency loss on the feature image $\mathbf{I}$:

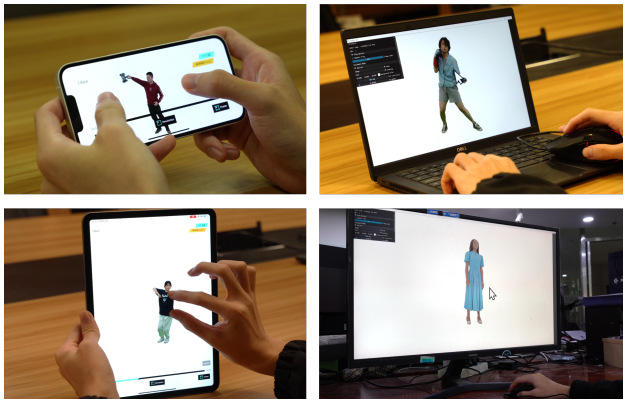

Figure 9. Our VideoRF facilitates dynamic radiance field rendering on ubiquitous devices, from desktops to mobile phones.

$$\mathcal{L}_{\text{rgb}_{\text{fine}}} = \sum_{\mathbf{r}\in\mathcal{R}} \|\mathbf{c}(\mathbf{r}) - \hat{\mathbf{c}}(\mathbf{r})\|^2, \tag{12}$$

$$\mathcal{L}_{\text{TV}_{\text{fine}}} = \frac{1}{|\mathbf{V}_\sigma|} \sum_{\mathbf{v}\in\mathbf{V}_\sigma} \sqrt{\Delta_x^2\mathbf{v} + \Delta_y^2\mathbf{v} + \Delta_z^2\mathbf{v}} \tag{13}$$

$$\mathcal{L}_{\text{spatial}} = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p}\in\mathcal{V}} \left(\Delta_u(\mathbf{p}) + \Delta_v(\mathbf{p})\right), \tag{14}$$

$$\mathcal{L}_{\text{temporal}} = \|\mathbf{I}^t - \mathbf{I}^{t-1}\|_1, \tag{15}$$

$$\mathcal{L}_{\text{fine}} = \mathcal{L}_{\text{rgb}_{\text{fine}}} + \lambda_{\text{TV}}\mathcal{L}_{\text{TV}_{\text{fine}}} + \lambda_{\text{s}}\mathcal{L}_{\text{spatial}} + \lambda_{\text{t}}\mathcal{L}_{\text{temporal}}, \tag{16}$$

where $\lambda_{\text{TV}} = 0.000016$, $\lambda_{\text{s}} = 0.0001$ and $\lambda_{\text{t}} = 0.0001$. The total variation loss is specifically activated during training iterations 1000 to 12000. We continue to use the Adam optimizer for 16000 iterations with a batch size of 10192 rays. The learning rates for $\mathbf{V}_\sigma$, $\mathbf{V}_c$, and the global MLP are set to 0.1, 0.11, and 0.002, respectively.

9. Implementation Details for VideoRF Player

During the codec process of our player, the 2D density map and each feature map channel are considered as single channel images. These images are normalized and quantized

Method	Metric	20	50	100	250	500	1000
		best	second-best				
HumanRF	↓ LPIPS	0.120	0.138	0.135	0.151	0.155	0.160
	↑ PSNR	31.02	30.26	30.25	28.98	29.50	29.19
	↑ SSIM	0.893	0.888	0.896	0.888	0.885	0.881
TiNeuVox	↓ LPIPS	0.352	0.298	0.406	0.430	0.436	0.452
	↑ PSNR	27.51	26.62	24.13	22.98	22.30	21.28
	↑ SSIM	0.782	0.791	0.760	0.752	0.751	0.747
NDVG	↓ LPIPS	0.240	0.281	0.354	0.435	0.453	0.481
	↑ PSNR	28.76	25.83	23.13	21.17	20.05	17.83
	↑ SSIM	0.841	0.812	0.763	0.731	0.724	0.692
HyperNeRF	↓ LPIPS	0.233	0.250	0.275	0.322	0.374	0.388
	↑ PSNR	25.75	26.53	25.96	24.85	23.29	23.04
	↑ SSIM	0.827	0.818	0.800	0.777	0.758	0.761
NeuralBody	↓ LPIPS	0.288	0.333	0.354	0.368	0.396	0.429
	↑ PSNR	27.51	25.88	27.18	25.30	24.81	25.68
	↑ SSIM	0.804	0.777	0.739	0.762	0.745	0.668
TAVA	↓ LPIPS	0.261	0.303	0.341	0.410	0.467	0.504
	↑ PSNR	28.47	26.93	25.83	24.28	23.13	22.21
	↑ SSIM	0.820	0.801	0.782	0.749	0.721	0.704
MeRF	↓ LPIPS	0.278	0.276	0.259	0.271	0.272	0.263
	↑ PSNR	28.24	28.19	27.24	27.22	27.31	27.68
	↑ SSIM	0.783	0.791	0.815	0.807	0.805	0.814
ReRF	↓ LPIPS	0.297	0.296	0.297	0.296	0.292	0.294
	↑ PSNR	28.69	28.51	28.55	28.33	28.12	27.73
	↑ SSIM	0.834	0.828	0.827	0.836	0.836	0.841
Ours	↓ LPIPS	0.276	0.285	0.283	0.278	0.274	0.275
	↑ PSNR	29.14	28.79	28.81	28.46	28.50	28.32
	↑ SSIM	0.840	0.835	0.830	0.838	0.840	0.844

Table 3. Quantitative comparison on long-duration sequence. We evaluate on the Actor 3, Sequence 1 of the Actors-HQ Dataset.

into 8-bit depth. The H.264 encoder converts these images into the yuvj444p color space for hardware compatibility. During decoding, the data is converted back from the yuvj444p color space to single-channel images with 8-bit depth. Meanwhile, we adopt a multi-resolution occupancy grid to bypass empty 3D spaces at various levels. This approach significantly reduces unnecessary network inferences during the ray marching process. The largest occupancy grid is derived from max-pooling the full-resolution binary mask. Each subsequent grid is designed to be half the resolution of its predecessor. For instance, considering our full-resolution binary mask is of size $288\times288\times288$, our multi-resolution occupancy grids follow suit with sizes of $144\times144\times144$, $72\times72\times72$, $36\times36\times36$, $18\times18\times18$, and $9\times9\times9$.

10. Additional Experiments

As illustrated in Fig. 9, our method can enable dynamic radiance field rendering on a wide range of devices, including desktops (an i7-12700F CPU and NVIDIA RTX3090 GPU), laptops (an i5-1135G7CPU and Integrated GPU), tablets (iPad Pro, an M2 chip) and mobile phones (iPhone 14 Pro, an A16 Bionic chip).

Long-duration dynamic scenes. Following the approach

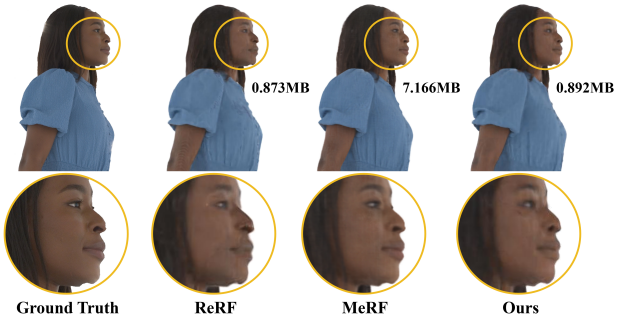

Figure 10. Qualitative comparison on the long-duration sequence against recent dynamic scene reconstruction methods and per frame static reconstruction methods.

Components	Size(KB)
Feature Images	661.62
3D to 2D Mapping Table	2.58
Occupancy Images	2.18
MLP Parameters	3.40
Total Size	669.78

Table 4. Storage of different components. The result is averaged over a sequence of Kpop scene from ReRF [65] dataset.

in HumanRF [18], we assess performance on a long-duration sequence (Actor3, sequence1, 1000 frames) from the Actors-HQ dataset. We compare our method with ReRF [65] and MeRF [50] through per-frame static reconstruction in Fig. 10. Our method keeps a small storage to enable streaming while maintaining a high rendering quality. We adopt the testing methods outlined in HumanRF. The performance metrics for HumanRF [18], TiNeuVox [11], NDVG [16], HyperNeRF [43], NeuralBody [45], TAVA [28] are directly sourced from the HumanRF publication. As shown in Fig. 10 and Tab. 3, our approach demonstrates its capability to sustain high photorealism which is only second to HumanRF throughout long-duration sequences. Note that, our VideoRF is the only method to enable rendering dynamic scenes on mobile platforms.

Storage of different components analysis. We present the storage requirements of each VideoRF component in Tab. 4. This encompasses the average file sizes for several key elements: the feature images for model detail, 3D-to-2D mapping table, occupancy images used to efficiently skip over empty spaces, and MLP parameters for the neural network. Note that our model's total average size is a mere 669.78KB. This compact representation facilitates rapid streaming across various devices.

11. Limitation and Future Work

As the first trial to enable a real-time dynamic radiance field approach capable of decoding and rendering on mobile de-

vices, our approach presents certain limitations. On average, training each frame takes approximately 20 minutes using a single NVIDIA RTX3090 GPU. This includes about 5 minutes for coarse training, less than 1 second for the baking stage, and around 14 minutes for the fine-grained sequential training. Fortunately, our coarse training stage can be parallel for each frame, and fine-grained sequential training can be parallel across groups. Consequently, with a setup of 8 RTX3090 GPUs, the training time can be reduced to approximately 3 minutes per frame. It should be noted that once a sequence has been trained and encoded, users as content consumers can directly use it, and the training cost is transparent to the users. Therefore, on a practical application level, our method can provide a smooth experience on multiple platforms for users. Faster training speed is indeed important for content creators, and this remains an area for our future work. Additionally, our method currently lacks support for temporal interpolation, signifying another direction for future exploration.