Supplementary Material for Neural Residual Radiance Fields for Streamably Free-Viewpoint Videos

Liao Wang[1,3] Qiang Hu[1] Qihan He[1,4] Ziyu Wang[1] Jingyi Yu[1]
Tinne Tuytelaars[2] Lan Xu[1†] Minye Wu[2†]
[1] ShanghaiTech University [2] KU Leuven [3] NeuDim [4] DGene

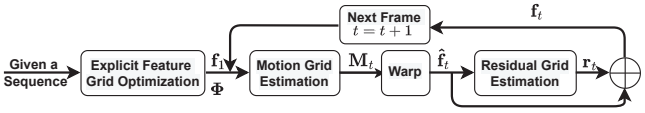

Figure 1. Overview for the training of ReRF.

A. Training details for Neural Residual Field.

Here we provide a detailed illustration of the training scheme to generate ReRF from RGB video inputs, as shown in Fig. 1. Specifically, we construct an explicit feature grid for the first frame. Then, sequentially given the former feature grid $\mathbf{f}_{t-1}$ and the RGB images at the current timestamp, we optimize our motion grid $\mathbf{M}_t$ and residual grid $\mathbf{r}_t$ for the current frame to generate our compact neural representation.

A.1. Feature Grid Optimization at the First Frame.

Given a long-duration multi-view sequence, we first learn an explicit feature grid and a global MLP Φ from the first frame. Similar to DVGO [19], we use an explicit density grid $\mathbf{V}_\sigma$ and a color feature grid $\mathbf{V}_c$ to represent the first frame. To render a view, we will cast rays through the pixels and sample points along rays. For the sampled point $\mathbf{p}$, we will query the scene property (density and color feature) efficiently through trilinear interpolation from the grids:

$$\begin{aligned}\text{Tri-Interp}\left(\mathbf{p}=[x,y,z],\mathbf{V}_\sigma\right):\left(\mathbb{R}^3,\mathbb{R}^{N_x\times N_y\times N_z}\right)\rightarrow\mathbb{R},\\ \text{Tri-Interp}\left(\mathbf{p}=[x,y,z],\mathbf{V}_c\right):\left(\mathbb{R}^3,\mathbb{R}^{C\times N_x\times N_y\times N_z}\right)\rightarrow\mathbb{R}^C,\end{aligned} \tag{1}$$

where C is the number of color feature dimension, N_x, N_y and N_z are the voxel resolutions of $\mathbf{V}_\sigma$ and $\mathbf{V}_c$. We choose $C = 12$ in our experiments. Following DVGO, we adopt the softplus and post-activation to obtain the density property of the sample points and apply the global MLP Φ to the color feature for the view-dependent rendering. This shallow MLP contains two hidden layers and each layer is 128 channels. During the training, we progressive upscale our density grid $\mathbf{V}_\sigma$ and color feature grid $\mathbf{V}_c$. The initial number of voxels is $125 \times 125 \times 125$. After reaching the training step 1000, 2000 and 4000, the final resolution will be upscaled to $150 \times 150 \times 150$, $200 \times 200 \times 200$ and $250 \times 250 \times 250$, respestively.

During training this explicit feature grid, we employ the photometric MSE loss and apply total variation loss on $\mathbf{V}_\sigma$:

$$\begin{aligned}\mathcal{L}_{render_{explicit}} &= \sum_{\mathbf{l}\in\mathbb{L}}\|\mathbf{c}(\mathbf{l})-\hat{\mathbf{c}}(\mathbf{l})\|,\\ \mathcal{L}_{TV_{explicit}} &= \frac{1}{|\mathbf{V}_\sigma|}\sum_{\mathbf{v}\in\mathbf{V}_\sigma}\sqrt{\Delta_x^2\mathbf{v}+\Delta_y^2\mathbf{v}+\Delta_z^2\mathbf{v}},\\ \mathcal{L}_{explicit} &= \mathcal{L}_{render_{explicit}}+\lambda_{\text{TV}}\mathcal{L}_{TV_{explicit}},\end{aligned} \tag{2}$$

where $\lambda_{\text{TV}} = 0.000016$; $\mathbb{L}$ is the set of training pixel rays; $\mathbf{c}(\mathbf{l})$ and $\hat{\mathbf{c}}(\mathbf{l})$ are the ground truth color and predicted color of a ray $\mathbf{l}$ respectively. $\Delta^2_{x,y,z}\mathbf{v}$ denotes the squared difference between the density value in the voxel. The total variation loss is only activated during the training iteration 1000 to 12000. We utilize the Adam optimizer for training 5000 iterations in the coarse stage and 16000 iterations in the fine stage with a batch size of 10192 rays. The learning rate for $\mathbf{V}_\sigma$, $\mathbf{V}_c$ and global MLP is 0.1, 0.11 and 0.002, respectively.

A.2. Motion Grid Optimization.

Here we provide details to generate our compact low-resolution motion grid $\mathbf{M}_t$, which represents the position offset from the current frame to the previous so as to exploit feature similarities. We propose to generate $\mathbf{M}_t$ from a densely estimated motion field $\mathbf{D}_t$, which is a grid with a shape of $3 \times N_x \times N_y \times N_z$ and contains the warping information from the frame t to frame $t-1$.

† The corresponding authors are Minye Wu (minye.wu@kuleuven.be) and Lan Xu (xulan1@shanghaitech.edu.cn).

For our dense motion field estimation, we first sample point $\mathbf{p}_t$ along the pixel ray of frame t. Then the sampled point $\mathbf{p}_t$ will query the 3D motion $\Delta\mathbf{p}_{t\to t-1} = \mathbf{D}_t(\mathbf{p}_t)$ through trilinear interpolation:

$$\text{Tri-Interp}\left(\mathbf{p}_t = [x, y, z], \mathbf{D}_t\right) : \left(\mathbb{R}^3, \mathbb{R}^{3\times N_x\times N_y\times N_z}\right) \to \mathbb{R}^3. \tag{3}$$

After finding the corresponding point $\mathbf{p}_{t-1} = \mathbf{p}_t + \Delta\mathbf{p}_{t\to t-1}$, we can get the feature from the previous feature grid $\mathbf{f}_{t-1}$ for $\mathbf{p}_t$. Then, the global MLP Φ will decode color feature to RGB space. Finally, the pixel color can be calculated through volume rendering.

During this estimation, we also progressively upscale the deformation field $\mathbf{D}_t$ from $(125 \times 125 \times 125) \to (150 \times 150 \times 150) \to (200 \times 200 \times 200) \to (250 \times 250 \times 250)$ after reaching the training step 1000, 2000 and 4000, respectively. We adopt the following photometric MSE loss and total variation loss to estimate $\mathbf{D}_t$:

$$\mathcal{L}_{render_{deform}} = \sum_{\mathbf{l}\in\mathbb{L}} \|\mathbf{c}(\mathbf{l}) - \hat{\mathbf{c}}(\mathbf{l})\|,$$

$$\mathcal{L}_{TV_{deform}} = \frac{1}{|\mathbf{D}_t|}\sum_{\mathbf{v}\in\mathbf{D}_t}\sqrt{\Delta_x^2(\mathbf{v}, d) + \Delta_y^2(\mathbf{v}, d) + \Delta_z^2(\mathbf{v}, d)},$$

$$\mathcal{L}_{deform} = \mathcal{L}_{render_{deform}} + \lambda_{\text{TV}}\mathcal{L}_{TV_{deform}}, \tag{4}$$

where the total variation loss enforces the smoothness of the dense motion field, and λ_{TV} is set to be 1. We use the Adam optimizer for training 3000 iterations in the coarse stage and 16000 iterations in the fine stage, with a batch size of 10192 rays and a learning rate of 10^{-4}.

Then, we generate the smooth and compact motion grid $\mathbf{M}_t$ from $\mathbf{D}_t$ through a motion pooling strategy as described in the main manuscript.

A.3. Residual Grid Optimization.

Here we provide implementation details to generate the sparse residual grid $\mathbf{r}_t$ of the current frame, which is used to compensate for warping errors and newly observed regions. Specifically, we first warp the previous frame feature $\mathbf{f}_{t-1}$ using the compact motion field $\mathbf{M}_t$ to generate a base feature grid $\hat{\mathbf{f}}_t$. Then, during the optimization, we shoot rays from the image pixels and sample points $\mathbf{p}_t$ along it. The base feature grid and residual grid are both queried through trilinear interpolation to obtain $\hat{\mathbf{f}}_t(\mathbf{p}_t)$ and $\mathbf{r}_t(\mathbf{p}_t)$:

$$\begin{aligned}&\text{Tri-Interp}\left(\mathbf{p}_t = [x, y, z], \hat{\mathbf{f}}_t\right) : \left(\mathbb{R}^3, \mathbb{R}^{C\times N_x\times N_y\times N_z}\right) \to \mathbb{R}^C,\\ &\text{Tri-Interp}\left(\mathbf{p}_t = [x, y, z], \mathbf{r}_t\right) : \left(\mathbb{R}^3, \mathbb{R}^{C\times N_x\times N_y\times N_z}\right) \to \mathbb{R}^C.\end{aligned} \tag{5}$$

Note that $C = 13$, since we union the density and color feature in our feature grid $\mathbf{f}$ representation. We obtain the final scene property of the current frame through the summation: $\mathbf{f}_t(\mathbf{p}_t) = \hat{\mathbf{f}}_t(\mathbf{p}_t) + \mathbf{r}_t(\mathbf{p}_t)$. Finally, the global MLP Φ will decode it into radiance fields with volume rendering to calculate pixel color.

We employ the same progressive training scheme for the residual grid, starting from $(125 \times 125 \times 125) \to (150\times150\times150) \to (200\times200\times200) \to (250\times250\times250)$ after reaching the training step 1000, 2000 and 4000, respectively. Besides the photo-metric MSE loss and total variation loss on density residual, we utilize an additional L1 loss to encourage the residual sparsity:

$$\mathcal{L}_{render_{residual}} = \sum_{\mathbf{l}\in\mathbb{L}} \|\mathbf{c}(\mathbf{l}) - \hat{\mathbf{c}}(\mathbf{l})\|,$$

$$\mathcal{L}_{TV_{residual}} = \frac{1}{|\mathbf{f_t^d}|}\sum_{\mathbf{v}\in\mathbf{f_t^d}}\sqrt{\Delta_x^2\mathbf{v} + \Delta_y^2\mathbf{v} + \Delta_z^2\mathbf{v}},$$

$$\mathcal{L}_{1residual} = \frac{1}{|\mathbf{r}_t|}\sum_{\mathbf{v}\in\mathbf{r}_t}(|\mathbf{v}_x| + |\mathbf{v}_y| + |\mathbf{v}_z|),$$

$$\mathcal{L}_{residual} = \mathcal{L}_{render_{residual}} + \lambda_{\text{TV}}\mathcal{L}_{TV_{residual}} + \lambda_{\text{residual}}\cdot\mathcal{L}_{\text{1residual}}, \tag{6}$$

where $\mathbf{f_t^d}$ represents the density of feature grid $\mathbf{f_t}$; $\lambda_{\text{TV}} = 0.000016$ and $\lambda_{\text{residual}} = 0.01$.

Similar to our first frame explicit grid optimization, the total variation loss of density residual is only activated during the training iteration 1000 to 12000. We adopt the Adam optimizer for training 5000 iterations in the coarse stage and 16000 iterations in the fine stage with a batch size of 10192 rays. The learning rate for the density of residual grid $\mathbf{r_t}$, and the color feature of residual grid $\mathbf{r_t}$ is 0.1, 0.11, respectively. Note that the base feature grid $\hat{\mathbf{f}}_t$ is fixed and we initialize the residual grid with zero value during our residual grid optimization.

B. ReRF Codec and Streamble Application

In the past decades, a number of image and video compression standards like JPEG [22], JPEG2000 [21], H.264/AVC [23] and H.265/HEVC [18] have been proposed and widely used in many practical applications. Most of these video compression methods follow a hybrid coding structure, where motion compensation and residual coding are adopted to reduce spatial and temporal redundancy. Recent work has also attempted to employ neural networks for video compression and has shown considerable performance [3, 4, 8, 9, 13, 14]. Inspired by these compression methods, we propose a ReRF-based codec and a companion FVV player for online streaming of long-duration dynamic scenes, while still guaranteeing an immersive exploration experience over existing networks. Fig. 2 demonstrates the overall pipeline of our framework.

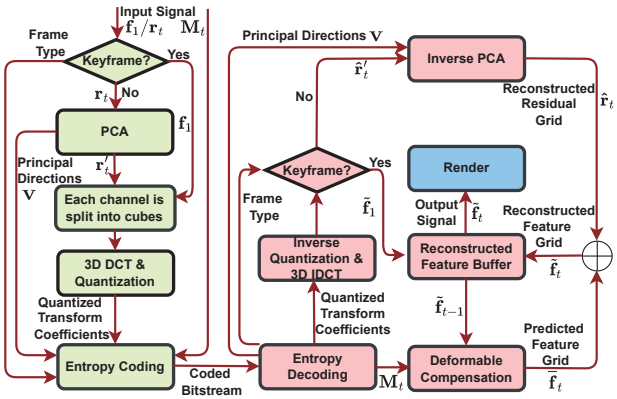

Figure 2. Overview of our proposed ReRF-based codec and player (the modeling elements of the encoder and decoder are shaded in light green and pink, respectively). The encoder compresses the input signal to produce a bitstream by using PCA, 3D-DCT, quantization, and entropy coding. The decoder receives the compressed bitstream, decodes each of the syntax elements, and reverses the coding process. Additionally, given the decoded motion field $\mathbf{M}_t$ and the previously reconstructed feature grid $\widetilde{\mathbf{f}}_{t-1}$, we can obtain the predicted feature grid $\bar{\mathbf{f}}_t$ by deformation.

B.1. Feature-level Residual Compression.

Both motion and residual grids are amenable to compression, especially for long-duration dynamic scenes. To make ReRF practical for users, we design a ReRF-based codec that follows the traditional keyframe-based strategy. We first divide the feature grid sequence into several continuous groups of feature grids (GOF), which is a collection of successive grids as shown in Fig 3. GOFs are comprised of two frame types, the I-feature grid (keyframe) and the P-feature grid. Each GOF begins with an I-feature grid which is coded independently of all other feature grids and contains most of the vital information for the following sequence of the P-feature grid. The p-feature grid contains a deformable compensated residual grid relative to the previous feature grid. Let $\{\mathbf{f}_1, \mathbf{r}_2, \cdots, \mathbf{r}_{t-1}, \mathbf{r}_t, \cdots\}$ denote a GOF, where $\mathbf{f}_1$ is the original feature grid and $\mathbf{r}_t$ is the residual grid at the current time step. Our goal is to generate high quality reconstructed feature grid $\widetilde{\mathbf{f}}_t$ at any given bitrate.

PCA. We first reshape $\mathbf{f}_1$ and $\mathbf{r}_t$ into $\mathbf{f}_1(m,n)$ and $\mathbf{r}_t(m,n)$, a $m \times n$ feature matrix, where m and n are the number of non-empty feature voxels and feature channels, respectively. Then, we perform linear Principal Component Analysis (PCA) [6] on $\mathbf{r}_t(m,n)$ to obtain the named-tuple $(\mathbf{U}, \mathbf{S}, \mathbf{V})$ which is the nearly optimal approximation of a singular value decomposition of $\mathbf{r}_t(m,n)$:

$$\mathbf{r}_t(m,n) = \mathbf{U} \cdot \mathrm{diag}(\mathbf{S}) \cdot \mathbf{V}^T, \tag{7}$$

where $\mathbf{V}$ is the $n \times q$ matrix, representing the principal directions, $\mathbf{S}$ is the q-vector, $\mathbf{U}$ is the $m \times q$ matrix. Finally,

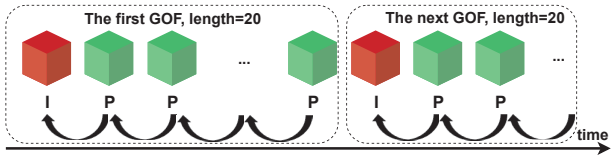

Figure 3. GOF structure.

we project the $\mathbf{r}_t$ to principal directions as follows:

$$\mathbf{r}'_t = \mathbf{r}_t \cdot \mathbf{V}. \tag{8}$$

3D DCT. Each channel of grid $\mathbf{f}_1$ and $\mathbf{r}'_t$ is divided into cubes of $8 \times 8 \times 8$ voxels and each cube is separately transformed by using a 3D DCT [1, 11]. Let residual voxels for each cube are denoted by $r(i,j,k)$, and the DCT coefficients $R(u,v,w)$ can be calculated as:

$$\begin{aligned} R(u,v,w) =& C_u C_v C_w \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1} r(i,j,k) \cos\left[\frac{(2i+1)\pi}{2N}u\right] \\ & \cos\left[\frac{(2j+1)\pi}{2N}v\right] \cos\left[\frac{(2k+1)\pi}{2N}w\right] \\ \text{where } C_u, C_v, C_w =& \begin{cases} \sqrt{\frac{1}{N}} & i,j,k = 0 \\ \sqrt{\frac{2}{N}} & \text{otherwise.} \end{cases} \end{aligned} \tag{9}$$

The transformed value at the coordinate origin $R(0,0,0)$ is the DC coefficient which is the most important value of the transformed coefficients. The amplitude of the DC coefficient is larger and contains more energy. While the rest coefficients are the AC coefficients, they contain little energy in the whole 3D-DCT, and most of their energy is concentrated on the major axis of the cube.

Quantization. Thereafter, the transform coefficients are quantized using a quantization matrix. The quantization matrix for a coefficient cube should have an entry for each coefficient. The values in the quantization matrix depend on if the corresponding coefficients are significant and also on the underlying quality factor being adopted. We perform scalar quantization on the 3D DCT coefficients. Each quantized transform coefficient is given by

$$\hat{R}(u,v,w) = round\left(\frac{R(u,v,w)}{S_q \times Q(u,v,w)}\right). \tag{10}$$

where S_q is a scaling factor and $Q(u,v,w)$ is a quantization matrix entry. In this work, we construct a quantization matrix based on psycho-visual experiments. The values of the quantization matrix are provided in quant.npy in the supplementary material.

Entropy Coding. The quantized transform coefficients are entropy coded and transmitted together with auxiliary

information such as motion field $\mathbf{M}_t$, frame type, etc. Entropy coding involves arranging the quantized DCT coefficients in a "3D zigzag" order [11], employing a run-length encoding (RLE) algorithm to group similar frequencies together, inserting length coding zeros, and then using Huffman coding [22, 24] for the remainder. The DC coefficients are coded separately from AC ones [22]. Specifically, the DC coefficients are coded using the Differential Pulse Code Modulation (DPCM) method [22]: except for the first DC coefficient, we encode the difference between the current DC coefficient and the previous DC coefficient. The AC coefficients are coded using the RLC method. To make it most likely hit a long run of zeros, a "3D zig-zag" scan is adopted. Finally, we use Huffman coding to further compress the DPCM-coded DC coefficients and the RLE-coded AC coefficients.

The experimental results show that our ReRF-based codec achieves three orders of magnitudes compression rate compared to per-frame-based neural representations [19]. Another advantage of our compression method is the ability to achieve variable bitrates via adjusting the scaling factor S_q during quantization, thus enabling dynamic adaptive streaming of ReRF according to the available bandwidth.

B.2. Network Streaming ReRF Player

We also implement a companion ReRF player for online streaming dynamic radiance fields of long sequences, with broad control functions. Our ReRF player supports downloading the coded bitstream from streaming media servers. When the bitstream is received, the I-feature grid $\widetilde{\mathbf{f}}_1$ is first reconstructed by performing inverse quantization and inverse transform on the quantized transform coefficients.

After the I-feature grid is reconstructed, the subsequently received P-feature grid will then be reconstructed. Specifically, the initial reconstructed residual grid $\hat{\mathbf{r}}'_t$ is generated by inverse quantization and inverse transform of the quantized transform coefficients. Then $\hat{\mathbf{r}}'_t$ is back-projected to the origin space by

$$\hat{\mathbf{r}}_t = \hat{\mathbf{r}}'_t \cdot \mathbf{V}^T. \tag{11}$$

Additionally, given the decoded motion field $\mathbf{M}_t$ and the previously reconstructed feature grid $\widetilde{\mathbf{f}}_{t-1}$, we can obtain the predicted feature grid $\overline{\mathbf{f}}_t$ by deformation. Let $\mathbf{p}$ denote the index of our explicit grids. Then, the predicted feature grid is formulated as:

$$\overline{\mathbf{f}}_t(\mathbf{p}) = \widetilde{\mathbf{f}}_{t-1}(\mathbf{p} + \mathbf{M}_t(\mathbf{p})). \tag{12}$$

The predicted feature grid $\overline{\mathbf{f}}_t$ as well as the initial reconstructed residual grid $\hat{\mathbf{r}}_t$ are added to produce the final reconstructed feature grid $\widetilde{\mathbf{f}}_t$, as follows:

$$\widetilde{\mathbf{f}}_t = \overline{\mathbf{f}}_t + \hat{\mathbf{r}}_t. \tag{13}$$

Figure 4. Our rendering results for forward facing scenes in neural 3D dataset.

Finally, the reconstructed feature grid $\widetilde{\mathbf{f}}_t$ is output to the renderer to generate photo-realistic FVV of dynamic scenes. As our ReRF player can efficiently reconstruct and render dynamic scenes, users are free to choose their views as if they were in the target scene.

Benefiting from the design of the GOF structure, our ReRF player allows fast seeking to a new position to play during playback. The reason is that the coded bitstream consists of successive GOFs. The first frame in a GOF is an I-feature grid (keyframe) which contains an independently coded feature. Encountering a new GOF in a compressed bitstream means that the decoder can decode a compressed feature grid without reconstructing any previous feature grid. With ReRF player, for the first time, users can pause, play, fast forward/backward, and seek dynamic radiance fields just like viewing a 2D video, bringing an unprecedented high-quality free-viewpoint viewing experience.

Note that the I-feature grid (keyframe) is different from the explicit feature grid for the first frame. We only use explicit feature grid to representation in the first frame training. All other sequential frame features are trained using residual grid $\mathbf{r}_t$ and can be generated to feature grid $\mathbf{f}_t$. GOF structure is used to enable fast seeking. It will choose key frame every GOF size. For I-feature grid, the full feature grid $\mathbf{f}_t$ is encoded (generated from residual grid) . For P-feature grid, the residual grid $\mathbf{r}_t$ is encoded.

C. Experiments

C.1. Dataset Details

Our captured dynamic datasets contain around 74 views at the resolution of 1920×1080 at 25fps. The cameras are the cylindrical distribution looking at the center. Most sequences are more than 1000 frames, The longest sequence contains 4000 frames. We use five real-world captured data and two synthetic data for experiments.

		Synthetic-NeRF		TanksTemples	
Method	Size↓	PSNR↑	SSIM↑	PSNR↑	SSIM↑
SRN [17]	-	22.26	0.846	24.10	0.847
NeRF [15]	5.0	31.01	0.947	25.78	0.864
NSVF [12]	-	31.75	0.953	28.48	0.901
SNeRG [7]	1771.5	30.38	0.950	-	-
POctrees [25]	1976.3	31.71	-	27.99	0.917
Plenoxels [5]	778.1	31.71	-	27.43	0.906
DVGO [19]	612.1	31.95	0.975	28.41	0.911
TRF-CP [2]	3.9	31.56	0.949	27.59	0.897
TRF-VM [2]	71.8	33.14	0.963	28.56	0.920
INGP [16]	63.3	33.18	-	-	-
CC-CP [20]	4.4	30.55	0.935	27.01	0.878
CC-HY [20]	88.0	32.37	0.955	28.08	0.913
Ours (high)	3.21	31.81	0.955	28.30	0.910
Ours (low)	0.98	30.14	0.941	27.16	0.893

Table 1. Comparison with recent methods on static scenes. We compare our method with previous and concurrent novel view synthesis methods on two datasets. All scores of the baseline methods are directly taken from their papers whenever available. Our method use minimal storage while maintaining a high PSNR.

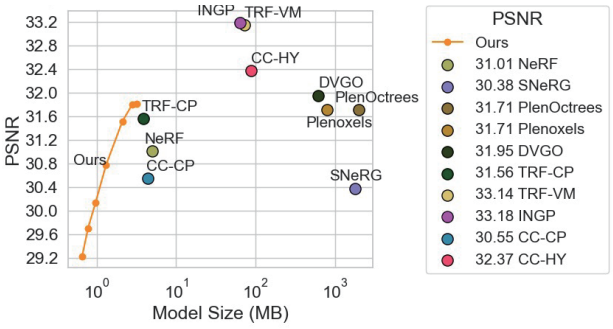

Figure 5. Quantitative results on Synthetic NeRF Dataset.

C.2. Additional Experimental Results

Static Scene Comparison. To demonstrate our I-feature grid (keyframe) compression performance, we also compare it with the existing static scene novel view synthesis approaches on the Synthetic NeRF dataset [15] and TanksTemples dataset [10] in Tab. 1. Compared to the original DVGO, orders of magnitudes smaller bitrates are achieved, without significantly sacrificing quality. We choose 2 different quantization factors to show our high-quality compression and low-quality compression results. Note that our high quality version has achieved the most compact modeling with essentially the same rendering quality as the original DVGO and also outperforms vanilla NeRF and many other methods. Our low-quality version also demonstrates that we can use a much more compact storage (<1MB) to reach a high PSNR(>30).

Fig. 5 and 6 further show that our method is the most compact and maintains a high render quality on

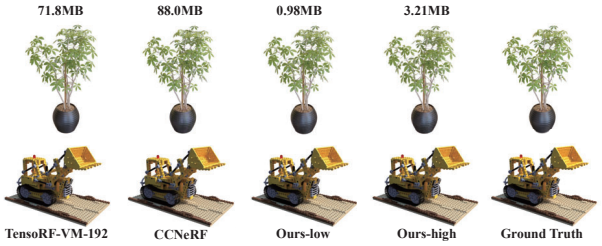

Figure 6. Qualitative comparisons with several recent works.

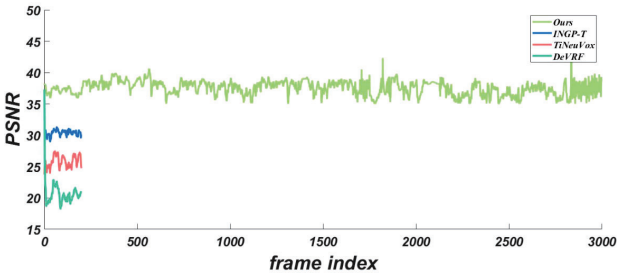

Figure 7. Performance on the long sequences (3000 frames).

static scenes in comparison with previous and concurrent methods. Seven different quantization scaling factors are adopted to achieve variable bitrates. Our methods also simultaneously achieve fast reconstruction and rendering.

References

[1] Raymond KW Chan and MC Lee. 3d-dct quantization as a compression technique for video sequences. In *Proceedings. International Conference on Virtual Systems and MultiMedia VSMM'97 (Cat. No. 97TB100182)*, pages 188–196. IEEE, 1997. 3

[2] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022. 5

[3] Zhenghao Chen, Guo Lu, Zhihao Hu, Shan Liu, Wei Jiang, and Dong Xu. Lsvc: A learning-based stereo video compression framework. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6063–6072, 2022. 2

[4] Runsen Feng, Yaojun Wu, Zongyu Guo, Zhizheng Zhang, and Zhibo Chen. Learned video compression with feature-level residuals. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 529–532, 2020. 2

[5] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 5

[6] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011. 3

[7] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radi-

ance fields for real-time view synthesis. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5855–5864, 2021. 5

[8] Zhihao Hu, Guo Lu, and Dong Xu. Fvc: A new framework towards deep video compression in feature space. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1502–1511, 2021. 2

[9] Zhihao Hu, Dong Xu, Guo Lu, Wei Jiang, Wei Wang, and Shan Liu. Fvc: An end-to-end framework towards deep video compression in feature space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–17, 2022. 2

[10] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017. 5

[11] M.C. Lee, Raymond K.W. Chan, and Donald A. Adjeroh. Quantization of 3d-dct coefficients and scan order for video compression. *Journal of Visual Communication and Image Representation*, 8(4):405–422, 1997. 3, 4

[12] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15651–15663. Curran Associates, Inc., 2020. 5

[13] Guo Lu, Xiaoyun Zhang, Wanli Ouyang, Li Chen, Zhiyong Gao, and Dong Xu. An end-to-end learning framework for video compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10):3292–3308, 2021. 2

[14] Guo Lu, Tianxiong Zhong, Jing Geng, Qiang Hu, and Dong Xu. Learning based multi-modality image and video compression. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6073–6082, 2022. 2

[15] 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 *European conference on computer vision*, pages 405–421. Springer, 2020. 5

[16] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. 5

[17] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *NeurIPS*, 2019. 5

[18] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, 2012. 2

[19] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5449–5459, 2022. 1, 4, 5

[20] Jiaxiang Tang, Xiaokang Chen, Jingbo Wang, and Gang Zeng. Compressible-composable nerf via rank-residual decomposition. In *Advances in Neural Information Processing Systems*, 2022. 5

[21] D.S. Taubman and M.W. Marcellin. Jpeg2000: standard for interactive imaging. *Proceedings of the IEEE*, 90(8):1336–1357, 2002. 2

[22] G.K. Wallace. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, 1992. 2, 4

[23] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003. 2

[24] En-hui Yang and Longji Wang. Joint optimization of run-length coding, huffman coding, and quantization table with complete baseline jpeg decoder compatibility. *IEEE Transactions on Image Processing*, 18(1):63–74, 2009. 4

[25] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5752–5761, 2021. 5