A TensoRF Representation Details.

We illustrate the feature grid of our tensorial radiance field and the tensor factors in TensoRF with both CP and VM decompositions in Fig. 5.

Number of components. The total number of tensor components (#Comp) is $(R_\sigma + R_c)$ for TensoRF-CP and $3(R_\sigma + R_c)$ for TensoRF-VM (because VM has three types of components). Therefore, the R we use for TensoRF-CP is three times as large as the R used for TensoRF-VM to achieve the same number of components shown in Tab. 2. We also find that using $R_\sigma < R_c$ is usually better than $R_\sigma = R_c$ when R_σ is large enough (> 8). In particular, for TensoRF-VM, we use $R_\sigma = R_c = 8$ for #Comp $= 48$; $R_\sigma = 8, R_c = 24$ for #Comp $= 96$; $R_\sigma = 16, R_c = 48$ for #Comp $= 192$; $R_\sigma = 32, R_c = 96$ for #Comp $= 384$. Note that, as discussed in Eqn. 3,4, here we apply the same number of components for $\mathcal{A}^X$, $\mathcal{A}^Y$, $\mathcal{A}^Z$ with $R_1 = R_2 = R_2 = R$ for both density and appearance (where R is R_σ and R_c respectively), assuming the three spatial dimensions are equally complex.

Forward-facing settings. We use the above settings with $R_1 = R_2 = R_2$, for all 360° object datasets in Tab. 1. On the other hand, Forward-facing scenes apparently appear differently in the three dimensions; especially, in NDC space, the X and Y spatial modes (corresponding to the image plane) contains more appearance information that is visible to rendering viewpoints. We therefore use more components for the $X - Y$ plane, corresponding to $\mathcal{A}^Z = \mathbf{v}^Z \circ \mathbf{M}^{X,Y}$. In this case, these $\mathcal{A}^Z$ components can also be seen as special compressed versions of neural MPIs. In particular, the detailed numbers of components we use for generating the results in Tab. 4 are: for #Comp=48 $R_{\sigma,1} = R_{\sigma,2} = 4$,$R_{\sigma,2} = 16$,$R_{c,1} = R_{c,2} = 4$,$R_{c,2} = 16$; for #Comp=96, $R_{\sigma,1} = R_{\sigma,2} = 4$,$R_{\sigma,2} = 16$,$R_{c,1} = R_{c,2} = 16$,$R_{c,48} = 16$.

Number of parameters. We briefly discuss the number of parameters in our model. With the same #Comp, when $I = J = K$ and $R_\sigma + R_c = R$, the total number of parameters used for TensoRF-CP is $3KR + PR_c$; for Tensor-VM, the number is $K^2R + KR + PR_c$ (here considering $R_\sigma/3$, $R_c/3$ are used to make the #Comp the same as TensoRF-CP). For example, for a $300 \times 300 \times 300$ feature grid with $P = 27$ channels (plus one density channel), the total number of parameters in a dense grid is 756 M; the number of parameters used for TensoRF-CP (when $R = 192$) is 360 K; the number of parameters used for TensoRF-VM (when $R = 192$) is 17 M. Our CP and VM model can achieve 0.048% and 2.25% compression rates respectively.

B More Implementation Details.

Loss functions. As described in sec. 4.4, we apply a L2 rendering loss and additional regularization terms to optimize our tensor factors for radiance field reconstruction. In general, this loss function is expressed by

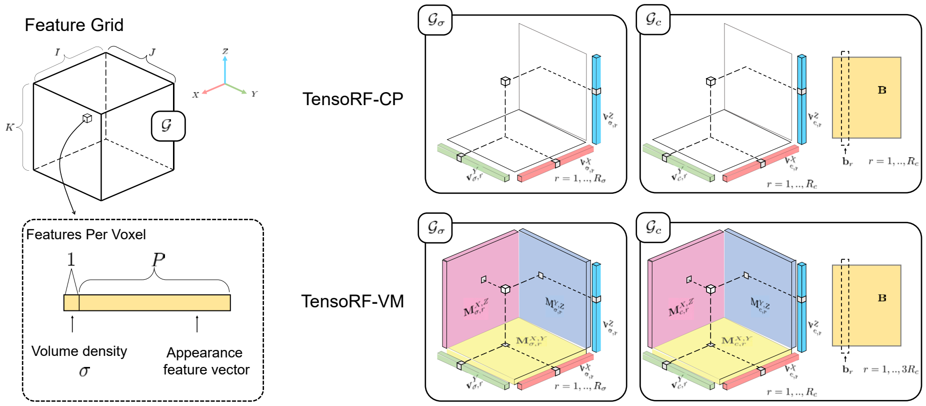

Fig. 5: Feature grids and factorized tensors in TensoRF. We leverage a regular voxel grid $\mathcal{G}$, covering a 3D scene, to model a radiance field of the scene. Each voxel of $\mathcal{G}$ contains multi-channel features, where one channel represents the volume density (σ) and the remaining P channels lead to an appearance feature vector (f_c) for computing view-dependent colors. We split the density and appearance features into two feature grids $\mathcal{G}_\sigma$ and $\mathcal{G}_c$, consider them as 3D and 4D tensors, and factorize them into compact factors with outer products. TensoRF with CP decomposition factorize the tensors into only vectors and TensoRF with VM decomposition factorize the tensor into vector and matrix factors (Eqn. 7, 8). Note that, each voxel of the original grid is only related to one value from each XYZ-mode vector/matrix factor in both decompositions, which are marked in the figure.

$$\mathcal{L} = \|C - \tilde{C}\|_2^2 + \omega \cdot \mathcal{L}_{reg} \tag{17}$$

Where $\tilde{C}$ is the groundtruth color, ω is weight of the regularization term.

To encourage the sparsity in the parameters of our tensor factors, we apply the standard L1 regularization, which we find is effective in improving the quality in extrapolating views and removing floaters/outerliers in final renderings. Note that, unlike previous methods [20,55] that penalize predicted per-point density with a Cauchy loss or entropy loss, our L1 regularizer is much simpler and directly applied on the parameters of tensor factors. We find that it is sufficient to apply the L1 sparsity loss only on the density parameters, expressed by

$$\mathcal{L}_{L1} = \frac{1}{N}\sum_{r=1}^{R_\sigma}(\|\mathbf{M}_{\sigma,r}\| + \|\mathbf{v}_{\sigma,r}\|), \tag{18}$$

where $\|\mathbf{M}_{\sigma,r}\|$ and $\|\mathbf{v}_{\sigma,r}\|$) are simply the sum of absolute values of all elements, and $N = R_\sigma \cdot (I \cdot J + I \cdot K + J \cdot K + I + J + K)$ is the total number of parameters. We use this L1 sparsity loss with a $\omega = 0.0004$ for the Synthetic NeRF and

Synthetic NSVF datasets. An ablation study on this L1 loss on the Synthetic NeRF dataset is shown in Tab. .

For real datasets that have very few input images (like LLFF[36]) or imperfect capture conditions (like Tanks and Temples [26] that has varying exposure and inconsistent masks), we find a TV loss is more efficient than the L1 sparsity loss, expressed by

$$\mathcal{L}_{TV} = \frac{1}{N}\sum(\sqrt{\triangle^2 \mathcal{A}^m_{\sigma,r}} + 0.1 \cdot \sqrt{\triangle^2 \mathcal{A}^m_{C,r}}), \tag{19}$$

Here $\triangle^2$ is the squared difference between the neighboring values in the matrix/vector factors; we apply a smaller weight (weighted by 0.1 additionally) on appearance parameters in the TV loss. We use $\omega = 1$ when using this TV loss.

Binary occupancy volume. To facilitate reconstruction, we compute a binary occupancy mask grid at steps 2000 and 4000 using the volume density prediction from the intermediate TensoRF model to avoid computation in empty space. For datasets that do not provide bounding boxes, we start from a conservatively large box and leverage the occupancy mask computed at step 2000 to re-compute a more compact bounding box, with which we shrink and resample our tensor factors, leading to more precise modeling. For forward-facing scenes in the LLFF dataset [36], we apply NDC transformation that bounds the scene in a perspective frustum.

More details. As described in Sec. 5, we use a small two-layer MLP with 128 channels in hidden layers as our neural decoding function. In particular, the input to this MLP contains the viewing direction and the features recovered by our tensor factors (no xyz positions are used). Similar to NeRF and NSVF [37,31], we also apply frequency encodings (with Sin and Cos functions) on both the viewing direction and features. Unlike NeRF that uses ten different frequencies, we use only two.

During optimization, we also apply an exponential learning rate decay to make the optimization more stable when the reconstruction is being finished. Specifically, we decay our initial learning rates at every training step, until decayed by a factor of 0.1 in the end of the optimization.

C More Evaluation.

We perform an ablation study to evaluate our L1 regularization. Tab. 5 shows how our framework performs by removing the L1 regularization on the Synthetic-NeRF dataset, our models exceed NeRF fidelity (31.01 in average) even without regularization. We observe the performance gap between w/ and w/o L1 regularization is mostly caused by the floaters in the empty space. We also provide more results on our model with different numbers of training steps in Tab. 6, which is basically a detailed version of Tab. 3 with more settings. These results showcase that our models consistently improve when training with more iterations.

	PSNR	SSIM	$LPIPS_{VGG}$	$LPIPS_{Alex}$
CP-384	31.56/31.23	0.949/0.947	0.076/0.078	0.041/0.043
VM-48	32.39/31.71	0.957/0.953	0.057/0.062	0.032/0.036
VM-192-SH	32.00/31.14	0.955/0.949	0.058/0.068	0.058/0.044
VM-192	33.14/32.43	0.963/0.960	0.047/0.052	0.027/0.030

Table 5: We compare the averaged scores against w/o L1 regularization on the Synthetic-NeRF dataset.

	5k	8k	10k	12k	15k	20k	30k	40k	60k	100k
CP-192	28.38	29.13	29.93	30.38	30.80	31.18	31.56	31.75	32.03	32.18
VM-48	29.28	30.39	31.11	31.47	31.80	32.08	32.39	32.55	32.68	32.84
VM-96	29.65	30.72	31.52	31.93	32.26	32.56	32.86	33.00	33.17	33.29
VM-192	29.86	30.93	31.74	32.17	32.52	32.85	33.14	33.27	33.44	33.54
VM-384	29.95	30.88	31.75	32.20	32.62	32.94	33.21	33.35	33.52	33.64

Table 6: PSNRs on the Synthetic NeRF datasets with different numbers of training steps. This is more detailed version than Tab. 3.

D Discussion

In fact, the reconstruction problem with dense feature grid representation is relatively over-parameterized/under-determined; e.g., a 300^3 grid with 27 channels has >700M parameters, while one hundred 800×800 images provide only 64M pixels for supervision. Therefore, many design choices – including pruning empty voxels, coarse-to-fine reconstruction, and adding additional losses, which have been similarly used in TensoRF and concurrent works (DVGO, Plenoxels) – are all essentially trying to reduce/constrain the parameter space and avoid over-fitting. In general, low-rank regularization is crucial in addressing many reconstruction problems, like matrix completion [6], compressive sensing [15], denoising [21]; tensor decomposition has also been widely used in tensor completion [30,16], which is similar to our task. Tensor decomposition naturally provides low-rank constraints and reduces parameters; this similarly benefits the radiance field reconstruction as demonstrated by our work.

Moreover, TensoRF represents a 5D radiance field function that expresses both scene geometry and appearance; hence, we believe our 4D tensor is generally low-rank, because a 3D scene typically contains a lot of similar geometry structures and material properties across different locations. Note that, in various appearance acquisition tasks, similar low-rank constraints have been successfully applied for reconstructing other functions, including the 4D light transport function in relighting [60] and the 6D SVBRDF function in material reconstruction [72,39] (where a common idea is to model a sparse set of basis BRDFs; this is similar to our modeling of vector components in the feature dimension in the matrix $\mathbf{B}$). We combine low-rank constraints and neural networks from a novel perspective, in tensor-based radiance field reconstruction. TensoRF essentially models the scene with global basis components, discovering the scene geometry and appearance commonalities across the spatial and feature dimensions.

E Limitations and Future Work.

Our approach achieves high-quality radiance field reconstruction for 360° objects and forward-facing scenes; however, our method currently only supports bounded scenes with a single bounding box and cannot handle unbounded scenes with both foreground and background content. Combining our method with techniques like NeRF++ [70] to separately model a foreground field inside a regular box and a background field inside another box defined in a spherical coordinate space can potentially extend our method to address unbounded scenes. Despite the success in per-scene optimization shown in this paper, an interesting future direction is to discover and learn general basis factors across scenes on a large-scale dataset, leveraging data priors to further improve the quality or enable other applications like GANs (as done in GRAF [51], GIRRAF [40] and EG3D [8]).

F Acknowledgements

We would like to thank Yannick Hold-Geoffroy for his useful tips in video animation, Qiangeng Xu for providing some baseline results, and, Katja Schwarz and Michael Niemeyer for providing helpful video materials. This project was supported by NSF grant IIS-1764078 and gift money from Kingstar. We would also like to thank all anonymous reviewers for their encouraging comments.

G Per-scene Breakdown.

Tab. 8-11 provide a per-scene break down for quantity metrics in Synthesis-nerf [34], Synthe-nsvf [31], Tanks&Templates [26] and forward-facing [36] dataset.

Methods	Avg.	*Chair*	*Drums*	*Ficus*	*Hotdog*	*Lego*	*Materials*	*Mic*	*Ship*
PSNR↑									
SRN [54]	22.26	26.96	17.18	20.73	26.81	20.85	18.09	26.85	20.60
NeRF [37]	31.01	33.00	25.01	30.13	36.18	32.54	29.62	32.91	28.65
NSVF [31]	31.75	33.19	25.18	31.23	37.14	32.29	32.68	34.27	27.93
SNeRG [20]	30.38	33.24	24.57	29.32	34.33	33.82	27.21	32.60	27.97
PlenOctrees [68]	31.71	34.66	25.31	30.79	36.79	32.95	29.76	33.97	29.42
Plenoxels [50]	31.71	33.98	25.35	31.83	36.43	34.10	29.14	33.26	29.62
DVGO [55]	31.95	34.09	25.44	32.78	36.74	34.64	29.57	33.20	29.13
Ours-CP-384	31.56	33.60	25.17	30.72	36.24	34.05	30.10	33.77	28.84
Ours-VM-192-SH	32.00	34.68	25.37	32.30	36.30	35.42	29.30	33.21	29.46
Ours-VM-48	32.39	34.68	25.58	33.37	36.81	35.51	29.45	33.59	30.12
Ours-VM-192-15k	32.52	34.95	25.63	33.46	36.85	35.78	29.78	33.69	30.04
Ours-VM-192-30k	**33.14**	**35.76**	**26.01**	**33.99**	**37.41**	**36.46**	**30.12**	**34.61**	**30.77**

Table 7: PSNR results on each scene from the **Synthetic-NeRF** [37] dataset.

Methods	Avg.	*Chair*	*Drums*	*Ficus*	*Hotdog*	*Lego*	*Materials*	*Mic*	*Ship*
SSIM$\uparrow$									
SRN [54]	0.846	0.910	0.766	0.849	0.923	0.809	0.808	0.947	0.757
NeRF [37]	0.947	0.967	0.925	0.964	0.974	0.961	0.949	0.980	0.856
NSVF [31]	0.953	0.968	0.931	0.973	0.980	0.960	0.973	0.987	0.854
SNeRG [20]	0.950	0.975	0.929	0.967	0.971	0.973	0.938	0.982	0.865
PlenOctrees [68]	0.958	0.981	0.933	0.970	**0.982**	0.971	**0.955**	0.987	0.884
Plenoxels [50]	0.958	0.977	0.933	0.976	0.980	0.976	0.949	0.985	0.890
DVGO [55]	0.957	0.977	0.930	0.978	0.980	0.976	0.951	0.983	0.879
Ours-CP-384	0.949	0.973	0.921	0.965	0.975	0.971	0.950	0.983	0.857
Ours-VM-192-SH	0.955	0.979	0.928	0.976	0.977	0.978	0.941	0.983	0.875
Ours-VM-48	0.957	0.980	0.929	0.979	0.979	0.979	0.942	0.984	0.883
Ours-VM-192-15k	0.959	0.982	0.933	0.981	0.980	0.981	0.949	0.985	0.886
Ours-VM-192-30k	**0.963**	**0.985**	**0.937**	**0.982**	**0.982**	**0.983**	0.952	**0.988**	**0.895**
LPIPS$_{VGG}\downarrow$									
SRN [54]	0.170	0.106	0.267	0.149	0.100	0.200	0.174	0.063	0.299
NeRF [37]	0.081	0.046	0.091	0.044	0.121	0.050	0.063	0.028	0.206
PlenOctrees [68]	0.053	0.022	0.076	0.038	**0.032**	0.034	0.059	0.017	0.144
Plenoxels [50]	0.049	0.031	**0.067**	0.026	0.037	0.028	**0.057**	**0.015**	**0.134**
DVGO [55]	0.053	0.027	0.077	0.024	0.034	0.028	0.058	0.017	0.161
Ours-CP-384	0.076	0.044	0.114	0.058	0.052	0.038	0.068	0.035	0.196
Ours-VM-192-SH	0.058	0.031	0.082	0.028	0.048	0.024	0.069	0.022	0.160
Ours-VM-48	0.057	0.030	0.087	0.028	0.039	0.024	0.072	0.021	0.155
Ours-VM-192-15k	0.053	0.026	0.078	0.025	0.038	0.021	0.063	0.020	0.153
Ours-VM-192-30k	**0.047**	**0.022**	0.073	**0.022**	**0.032**	**0.018**	0.058	**0.015**	0.138
LPIPS$_{Alex}\downarrow$									
NSVF [31]	0.047	0.043	0.069	0.017	0.025	0.029	0.021	0.010	0.162
DVGO [55]	0.035	0.016	0.061	0.015	0.017	0.014	**0.026**	0.014	0.118
Ours-CP-384	0.041	0.022	0.069	0.024	0.024	0.014	0.031	0.018	0.130
Ours-VM-192-SH	0.058	0.031	0.082	0.028	0.048	0.024	0.069	0.022	0.160
Ours-VM-48	0.032	0.014	0.059	0.015	0.017	0.009	0.036	0.012	0.098
Ours-VM-192-15k	0.032	0.013	0.056	0.014	0.017	0.009	0.029	0.013	0.101
Ours-VM-192-30k	**0.027**	**0.010**	**0.051**	**0.012**	**0.013**	**0.007**	**0.026**	**0.009**	**0.085**

Table 8: Quantitative results on each scene from the **Synthetic-NeRF** [37] dataset.

Fig. 6: Our rendering results on **Synthetic-NeRF** dataset. From top to bottom: Ship, Hotdog, Lego, Mic, Chair, Drums, Materials, Ficus.

Methods	Avg.	*Wineholder*	*Steamtrain*	*Toad*	*Robot*	*Bike*	*Palace*	*Spaceship*	*Lifestyle*
PSNR↑									
SRN [54]	24.33	20.74	25.49	25.36	22.27	23.76	24.45	27.99	24.58
NeRF [37]	30.81	28.23	30.84	29.42	28.69	31.77	31.76	34.66	31.08
NSVF [31]	35.13	32.04	35.13	33.25	35.24	37.75	34.05	**39.00**	**34.60**
DVGO [55]	35.08	30.26	36.56	33.10	36.36	38.33	34.49	37.71	33.79
Ours-CP-384	34.48	29.92	36.07	31.37	35.92	36.74	36.26	37.01	32.54
Ours-VM-192-SH	35.30	29.72	37.33	34.03	37.59	38.61	36.09	35.82	33.21
Ours-VM-48	35.34	30.46	37.06	33.13	36.92	37.98	36.32	37.19	33.68
Ours-VM-192-15k	35.59	30.31	37.20	33.63	37.29	38.33	36.57	37.77	33.62
Ours-VM-192-30k	**36.52**	**31.32**	**37.87**	**34.85**	**38.26**	**39.23**	**37.56**	38.60	34.51
SSIM↑									
SRN [54]	0.882	0.850	0.923	0.822	0.904	0.926	0.792	0.945	0.892
NeRF [37]	0.952	0.920	0.966	0.920	0.960	0.970	0.950	0.980	0.946
NSVF [31]	0.979	**0.965**	0.986	0.968	0.988	0.991	0.969	**0.991**	**0.971**
DVGO [55]	0.975	0.949	0.989	0.966	0.992	0.991	0.962	0.988	0.965
Ours-CP-384	0.971	0.947	0.986	0.950	0.990	0.987	0.971	0.984	0.951
Ours-VM-192-SH	0.977	0.953	0.988	0.974	0.993	0.991	0.972	0.982	0.964
Ours-VM-48	0.976	0.952	0.988	0.968	0.992	0.990	0.973	0.985	0.962
Ours-VM-192-15k	0.978	0.953	0.989	0.972	0.993	0.991	0.975	0.987	0.964
Ours-VM-192-30k	**0.982**	0.961	**0.991**	**0.978**	**0.994**	**0.993**	**0.979**	0.989	0.968
LPIPS$_{VGG}$ ↓									
DVGO [55]	0.033	0.055	0.019	0.047	0.013	0.011	0.043	**0.019**	0.054
Ours-CP-384	0.045	0.082	0.031	0.067	0.016	0.023	0.031	0.028	0.084
Ours-VM-192-SH	0.031	0.057	0.024	0.035	0.011	0.013	0.030	0.026	0.051
Ours-VM-48	0.034	0.061	0.023	0.047	0.013	0.014	0.029	0.025	0.059
Ours-VM-192-15k	0.031	0.060	0.020	0.040	0.011	0.012	0.028	0.022	0.055
Ours-VM-192-30k	**0.026**	**0.051**	**0.017**	**0.031**	**0.010**	**0.010**	**0.022**	0.020	**0.048**
LPIPS$_{Alex}$ ↓									
SRN [54]	0.141	0.224	0.082	0.204	0.120	0.075	0.240	0.061	0.120
NeRF [37]	0.043	0.096	0.031	0.069	0.038	0.019	0.031	0.016	0.047
NSVF [31]	0.015	**0.020**	0.010	0.032	0.007	0.004	0.018	**0.006**	**0.020**
DVGO [55]	0.019	0.038	0.010	0.030	0.005	0.004	0.027	0.009	0.027
Ours-CP-384	0.021	0.040	0.010	0.039	0.006	0.007	0.014	0.015	0.042
Ours-VM-192-SH	0.015	0.030	0.008	0.021	**0.003**	**0.003**	0.016	0.016	0.025
Ours-VM-48	0.016	0.031	0.008	0.025	0.004	0.004	0.015	0.013	0.026
Ours-VM-192-15k	0.015	0.033	0.008	0.022	0.004	0.004	0.015	0.011	0.026
Ours-VM-192-30k	**0.012**	0.024	**0.006**	0.016	**0.003**	**0.003**	**0.011**	0.009	0.021

Table 9: Quantitative results on each scene from the **Synthetic-NSVF** [31] dataset.

Fig. 7: Our rendering results on **NSVF** [31] dataset. From top to bottom: Spaceship, Robot, Toad, Lifestyle, Palace, Wineholder, Steamtrain.

Methods	Avg.	*Ignatius*	*Truck*	*Barn*	*Caterpillar*	*Family*
PSNR↑						
SRN [54]	24.10	26.70	22.62	22.44	21.14	27.57
NeRF [37]	25.78	25.43	25.36	24.05	23.75	30.29
NSVF [31]	28.48	27.91	26.92	27.16	**26.44**	33.58
PlenOctrees [68]	27.99	28.19	26.83	26.80	25.29	32.85
Plenoxels [50]	27.43	27.51	26.59	26.07	24.64	32.33
DVG [55]	28.41	28.16	**27.15**	27.01	26.00	33.75
Ours-CP-384	27.59	27.86	26.25	26.74	24.73	32.39
Ours-VM-192-SH	27.81	27.78	26.73	26.03	25.37	33.12
Ours-VM-48	28.06	28.22	26.81	26.70	25.43	33.12
Ours-VM-192-15k	28.07	28.27	26.57	26.93	25.35	33.22
Ours-VM-192-30k	**28.56**	**28.34**	27.14	**27.22**	26.19	**33.92**
SSIM↑						
SRN [54]	0.847	0.920	0.832	0.741	0.834	0.908
NeRF [37]	0.864	0.920	0.860	0.750	0.860	0.932
NSVF [31]	0.901	0.930	0.895	0.823	0.900	0.954
PlenOctrees [68]	0.917	**0.948**	**0.914**	0.856	0.907	0.962
Plenoxels [50]	0.906	0.943	0.901	0.829	0.902	0.956
DVGO [55]	0.911	0.944	0.906	0.838	0.906	0.962
Ours-CP-384	0.897	0.934	0.885	0.839	0.879	0.948
Ours-VM-192-SH	0.907	0.942	0.900	0.834	0.897	0.960
Ours-VM-48	0.909	0.943	0.902	0.845	0.899	0.957
Ours-VM-192-15k	0.913	0.944	0.905	0.855	0.902	0.960
Ours-VM-192-30k	**0.920**	**0.948**	**0.914**	**0.864**	**0.912**	**0.965**
LPIP$_{VGG}$ ↓						
PlenOctrees [68]	0.131	0.080	0.130	0.226	**0.148**	0.069
Plenoxels [50]	0.162	0.102	0.163	0.303	0.166	0.078
DVGO [55]	0.155	0.083	0.160	0.294	0.167	0.070
Ours-CP-384	0.181	0.106	0.202	0.283	0.227	0.088
Ours-VM-192-SH	0.156	0.089	0.161	0.286	0.175	0.069
Ours-VM-48	0.155	0.085	0.161	0.278	0.177	0.074
Ours-VM-192-15k	0.152	0.084	0.162	0.269	0.173	0.071
Ours-VM-192-30k	**0.140**	**0.078**	**0.145**	**0.252**	0.159	**0.064**
LPIPS$_{Alex}$ ↓						
SRN [54]	0.251	0.128	0.266	0.448	0.278	0.134
NeRF [37]	0.198	0.111	0.192	0.395	0.196	0.098
NSVF [31]	0.155	0.106	0.148	0.307	0.141	0.063
DVGO [55]	0.148	0.090	0.145	0.290	0.152	0.064
Ours-CP-384	0.144	0.089	0.154	0.237	0.176	0.063
Ours-VM-192-SH	0.164	0.098	0.168	0.309	0.175	0.072
Ours-VM-48	0.145	0.089	0.145	0.266	0.161	0.066
Ours-VM-192-15k	0.140	0.087	0.150	0.240	0.157	0.066
Ours-VM-192-30k	**0.125**	**0.081**	**0.129**	**0.217**	**0.139**	**0.057**

Table 10: Quantitative results on each scene from the **Tanks&Temples** [26] dataset.

Fig. 8: Our rendering results on **Tanks&Temples**[26] dataset. From top to bottom: Family, Ignatius, Truck, Caterpillar, Barn.

Methods	Avg.	*Room*	*Fern*	*Leaves*	*Fortress*	*Orchids*	*Flower*	*T-Rex*	*Horns*
PSNR↑									
NeRF [37]	26.50	**32.70**	25.17	20.92	31.16	**20.36**	27.40	26.80	27.45
Plenoxels [50]	26.29	30.22	25.46	**21.41**	31.09	20.24	27.83	26.48	27.58
Ours-VM-48	26.51	31.80	25.31	21.34	31.14	20.02	28.22	26.61	27.64
Our-VM-96	**26.73**	32.35	**25.27**	21.30	**31.36**	19.87	**28.60**	**26.97**	**28.14**
SSIM↑									
NeRF [37]	0.811	0.948	0.792	0.690	0.881	0.641	0.827	0.880	0.828
Plenoxels [50]	**0.839**	0.937	**0.832**	**0.760**	0.885	**0.687**	0.862	0.890	0.857
Ours-VM-48	0.832	0.946	0.816	0.746	0.889	0.655	0.859	0.890	0.859
Ours-VM-96	**0.839**	**0.952**	0.814	0.752	**0.897**	0.649	**0.871**	**0.900**	**0.877**
LPIPS$_{VGG}$ ↓									
NeRF [37]	0.250	0.178	0.280	0.316	0.171	0.321	0.219	0.249	0.268
Plenoxels [50]	0.210	0.192	**0.224**	**0.198**	0.180	**0.242**	0.179	0.238	0.231
Ours-VM-48	0.217	0.181	0.237	0.230	0.159	0.283	0.187	0.236	0.221
Ours-VM-96	**0.204**	**0.167**	0.237	0.217	**0.148**	0.278	**0.169**	**0.221**	**0.196**
LPIPS$_{Alex}$ ↓									
Ours-VM-48	0.135	0.093	0.161	0.167	0.084	0.204	0.121	0.108	0.146
Ours-VM-96	0.124	0.082	0.155	0.153	0.075	0.201	0.106	0.099	0.123

Table 11: Quantitative results on each scene from the **forward-facing** [31] dataset.

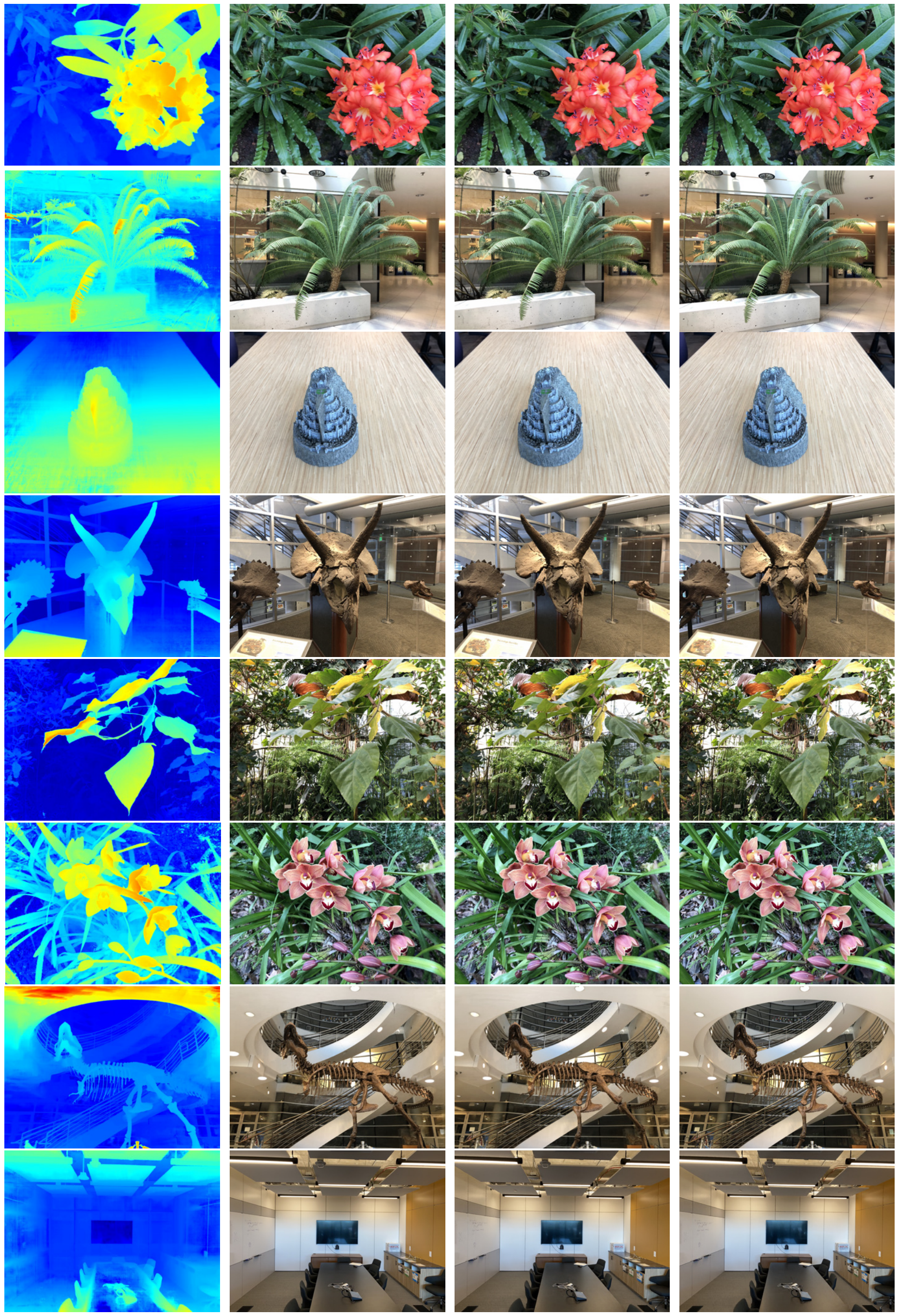

Fig. 9: Our rendering results on **forward-facing** [31] dataset. From top to bottom: Flower, Fern, Fortress, Horn, Leaves, Orchids, T-Rex, Room.