A. Supplementary Materials

A.1. Overview

In this supplementary document, we first present additional experimental results, including run-time analysis. Second, we provide the implementation details of the proposed framework. Third, we compare our method (*i.e*. explicit representations) and the current approaches based on implicit representations. Finally, we discuss the limitations of the proposed scheme and the future research directions. More qualitative comparisons are available at https://hhsinping.github.io/3d_scene_stylization.

A.2. Additional Experimental Results

A.2.1 LLFF and Shiny datasets

To demonstrate the generalization ability of the proposed method, we use the model trained on the Tanks and Temples dataset [14] to produce the stylization results on two additional datasets: LLFF [26] and Shiny [55].

Figure 1. **Additional results.** We show additional results on LLFF and Shiny datasets using the model trained on the Tanks an Temples dataset.

A.2.2 LST→NeRF++

As shown in Figure 5 in the paper, applying image stylization schemes before the SVS [43] framework produces blurry results. In this experiment, we show that replacing the SVS [43] with NeRF++ [61] approaches suffers from the similar issue. In Figure 2, we present the results LST [17] → NeRF++ [61]. Since the input images are not consistent due to the per-image stylization by the LST approach, the NeRF++ model tends to *blend* such inconsistency, which leads to blurry results.

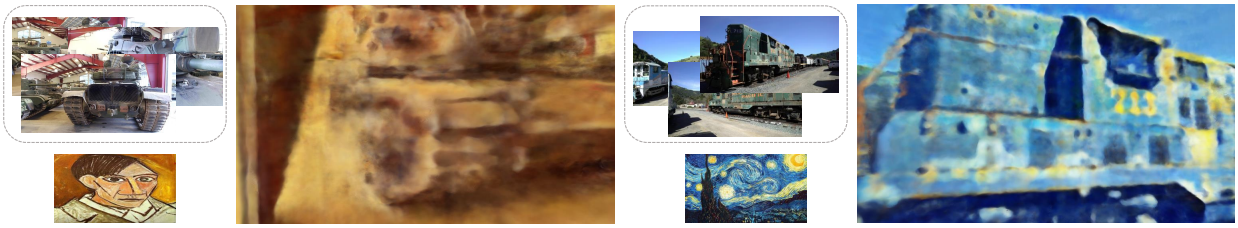
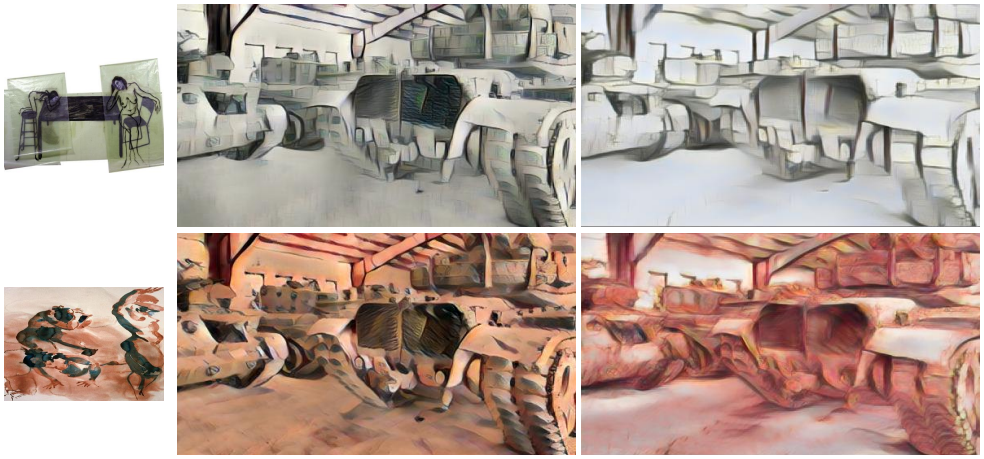


Figure 2. **LST→NeRF++.** The NeRF++ approach produces blurry results if the input images are not consistent due to the per-image stylization by the LST approach.

A.2.3 Ablation Study on Stylization Level

We use the pre-trained VGG-19 model [45] to extract the feature of the input images for the point cloud construction. By extracting the features from different layers of the VGG-19 network, our point cloud representation encodes different levels of the style information. Figure 3 demonstrates that our framework is capable of transferring the different style levels. Specifically, building the point cloud representation using the deeper (*e.g*. relu4_1) features produces more distortion, while using the shallower (*e.g*. relu3_1) features generates more photo-realistic (*i.e*. preserve more content information) effects.



relu3_1 (ours) relu4_1

Figure 3. **Ablation study on stylization level.** We show that our framework is able to transfer styles of different levels. Extracting the image features from the deeper layers (relu4_1) of the pre-trained VGG-19 network produces more distortion, while using features from the shallower layers (relu3_1) generates more photo-realistic stylization effects.

A.2.4 Ablation Study on Point Cloud Aggregation

To gather the style information of the constructed point cloud $\{f_p^c\}_{p=1}^{P}$, we sample a subset of P' points $\{f_p^c\}_{p=1}^{P'}$ and then use a radius parameter r to find k nearby points to form a point group. Each point group is aggregated to a vector by MLP layers and the max pooling operator to form the aggregated point cloud $\{f_p^{c'}\}_{p=1}^{P'}$. We conduct the following ablation studies to analyze the hyper-parameters r and k.

Radius r. Figure 4 shows the results of using different sets of radius parameters r for our point cloud aggregation modules. We empirically choose to use r=$\{0.05,0.1,0.2\}$ for better visual quality.

r={0.025, 0.05, 0.1} r={0.05, 0.1, 0.2} (ours) r={0.1, 0.2, 0.4}

Figure 4. **Ablation study on hyper-parameter r in the point cloud aggregation.** We compare the visual results of setting $r = \{0.025, 0.05, 0.1\}$, $r = \{0.05, 0.1, 0.2\}$, $r = \{0.1, 0.2, 0.4\}$. We empirically determine to use $r = \{0.05, 0.1, 0.2\}$ for better visual quality.

Number of sampled points k. We conduct an ablation study to decide the parameter k. Figure 5 shows the results of setting $k = 32/64/128$. We found that increasing the value of k produces results with higher contrast. We set k=64 since the results better match the style of the reference image.

Quantitative analysis of applying point cloud aggregation modules. In Table 1, we provide the quantitative analysis to understand the impact of applying the point cloud aggregation modules on the consistency issue. The results validate that using point aggregation modules improves both short-range and long-range consistency.

Figure 5. **Ablation study on hyper-parameter k in the point cloud aggregation.** We compare the visual results of using $k = 32/\mathbf{64}/128$, and empirically choose to use $k = 64$ for better visual quality.

Table 1. **Ablation study on point cloud aggregation.** We compute the short-range and long-range warping errors of the results generated by models with and without the point aggregation modules. We validate that applying the point aggregation modules achieves better consistency across various novel views.

(a) Short-range consistency

Method	Truck	Playground	Train	M60	Average
w/ aggregation	0.182	0.150	0.166	0.164	**0.165**
w/o aggregation	0.187	0.159	0.167	0.164	0.168

(b) Long-range consistency

Method	Truck	Playground	Train	M60	Average
w/ aggregation	0.590	0.332	0.409	0.434	**0.428**
w/o aggregation	0.595	0.374	0.417	0.409	0.434

A.2.5 PSNet for 3D Scene Stylization

The PSNet [3] model aims to transfer the style of the point cloud. However, it is not applicable to our problem for two reasons. First, PSNet requires per scene optimization on the "RGB" point cloud. It fails to handle large-scale scenes in the real-world with more than 60M points, such as those in the Tanks and Temples dataset [14]. To make the PSNet framework applicable to our problem, we first use uniform sampling to reduce the number of RGB points in the point cloud to 1M, then run PSNet framework to stylize the point cloud. We conduct the optimization process for the M60 and Truck scenes with 5000 iterations, which takes around 30 minutes for one specific combination of a scene and a reference image with desired style. Compared to the runtime of the proposed method shown in Table 2, the PSNet approach is time-consuming, thus limited for real-world applications. After the construction of the RGB point cloud, we project the points to the 2D image plane to synthesize images at novel views. As shown in Figure 6, we observe that PSNet does not generate desired stylization effect that matches the input reference image. In addition, the PSNet produces projection artifacts that require post-processing schemes (*e.g.* in-painting, smoothing) to refine the novel view synthesis results.

Figure 6. **3D scene stylization results of PSNet.** The PSNet [3] generates projection artifacts and fails to produce desired stylization effect that matches the input reference image.

A.2.6 Runtime Analysis

In Table 2, we show the training and inference time of the proposed method. All the processes are conducted on a desktop machine equipped with a Nvidia Titan Xp GPU. We note that after the point cloud transformation (3rd row) is completed, we can synthesize novel view images in near-real-time (*i.e*. 17 fps).

Table 2. **Run-time analysis.** We present the training and inference time of each stage in the proposed method.

Training time: decoder (seconds / per iteration)	0.31
Training time: point cloud transformation module (seconds / per iteration)	1.78
Inference time: constructing point cloud (seconds / per input image)	0.21
Inference time: stylizing point cloud (seconds / per scene)	0.74
Inference time: rendering novel view (seconds / per view)	0.06

A.3. Implementation Details

Network architecture. In Figure 7, we present the detailed architecture of each component in Figure 4 in the paper. We present the decoder architecture in Figure 8.

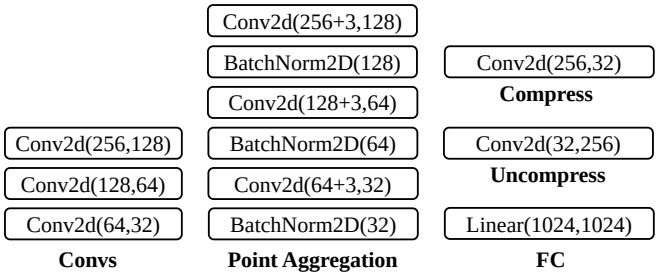


Figure 7. **Network architecture.** We present the network architecture of our point cloud transformation module illustrated in Figure 4 in the paper.

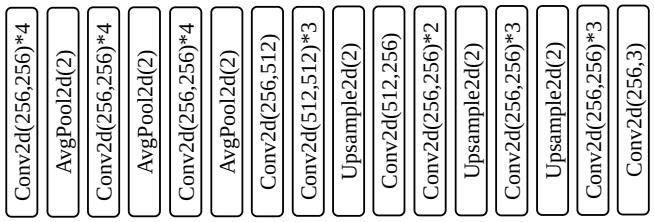


Figure 8. **Decoder.** We present the network architecture of our decoder module.

Point cloud transformation. To reduce the computation cost in our point cloud transformation step, we employ the *compress* and *uncompress* operations in practice, described as follows. The key is to reduce the feature dimension to accelerate the computation of the transformation matrix $\mathbf{T}$. Specifically, we reduce (*i.e*. compress) the feature dimension ($256 \rightarrow 32$) in the constructed point cloud $\{f_p^c\}_{p=1}^P$ through a MLP layer. We then transform the constructed point cloud $\{f_p^c\}_{p=1}^P$ using the transformation matrix $\mathbf{T}$ of size 32×32. Finally, we use a MLP layer to recover (*i.e*. uncompress) the feature dimension ($32 \rightarrow 256$) to produce the transformed point cloud $\{f_p^d\}_{p=1}^P$ for the following novel view synthesis stage. The process can be formulated as

$$f_p^d = \text{uncompress}(\mathbf{T}(\text{compress}(f_p^c - \bar{f}^c))) + \bar{f}^s \quad \forall p \in [1, \cdots, P], \tag{1}$$

where $\bar{f}^c$ is the mean of the features in the point cloud $\{f_p^c\}_{p=1}^P$, and $\bar{f}^s$ is the mean of the style feature map $\mathbf{F}^s$.

Point cloud aggregation. The number of points P and feature dimension c in each point cloud aggregation module is $\{P, c\} : \{\approx 2M, 256\} \rightarrow \{4096, 128\} \rightarrow \{2048, 64\} \rightarrow \{1024, 32\}$.

Novel view synthesis. Given a novel view v with the camera pose $\{\mathbf{R}_v, t_v\}$ and intrinsic $\mathbf{K}_v$, we first project the features in our point cloud to the 2D image plane. Specifically, we use the Pytorch3D [39] point cloud renderer for the projection of

features. We set the size of the z-buffer as 128 and the points are splatted to a region with radius of 2 pixels. We then use a decoder presented in Figure 8 to synthesize the final image from the projected 2D feature map.

Training. We implement our system in PyTorch, and use the Adam optimizer [13] with $\beta_1 = 0.9$, $\beta_2 = 0.9999$ for all network training. We first train the decoder module for 50K iterations with a batch size of 1 and learning rate of 0.0001. Following the WCT approach [18], the $\ell 1$ reconstruction loss illustrated in Line 417 in the paper is the combination of the pixel reconstruction loss and feature loss. Particularly, the feature loss is computed using the features of a pre-trained VGG-19 network, including {conv1_2, conv2_2, conv3_2, conv4_2, conv5_2}. We then train the transformation module for 50K iterations with a batch size of 1 and learning rate of 0.0001. The content loss described in Eq. (2) in the paper is computed by the features of layer relu4_1, while the style loss is computed by {relu1_1, relu2_1, relu3_1, relu4_1}. The weight λ for the style loss is set to 0.02. To improve the training efficiency, we uniformly down-sample the constructed point cloud to 600K features for each scene, and use all the features in the point cloud during the testing time.

A.4. Explicit vs. Implicit Representations

While implicit representation-based approaches [1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15, 16, 19, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 40, 41, 42, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63] produce high-quality (non-stylized) novel view synthesis results, we choose to leverage explicit representations due to the practical considerations that support real-world VR/AR applications: *efficiency* and *scalability*. Specifically, the NeRF++ method [61] is designed for complex unbounded 3D scenes. Nevertheless, it takes 24 hours to reconstruct a *particular* scene, and 30 seconds to render a 546×980 image. Moreover, the NeRF++-based framework produces blurry stylization results due to the inconsistency issue, as shown in Figure 2. Although there are recent efforts [11, 20, 21, 23, 28, 40, 41, 59] to accelerate the rendering process, these schemes are limited to single 3D objects or bounded 3D scenes. In contrast, the proposed method is efficient, and renders the stylized novel views in near-real-time, as presented in Table 2. Furthermore, the proposed method is more scalable than the NeRF++-based approaches since it handles arbitrary unbounded scenes and styles with a single trained model.

A.5. Limitations and Future Direction

We discuss the limitation of our method, which we plan to explore in the future work as follows. First, as shown in Figure 9, our 3D scene stylization approach is not aware of the objects in the scene. As a result, we cannot transfer the style of the particular part of the style image to the specific object/region of the 3D scene. Second, the proposed approach cannot significantly modify the geometry of the scene during the stylization process since 1) our point cloud is built according to the 3D proxy of the original scene and 2) we only transform the features in our point cloud, but not adjust the location of each point. In the future, we plan to explore the solution that is 1) 3D object-aware and 2) capable of modulating the geometry of the 3D scene to match the desired style.

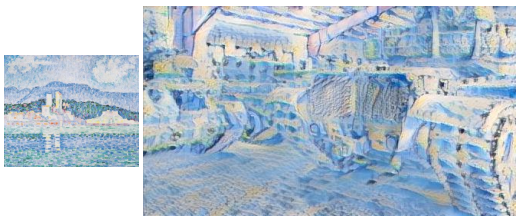


Figure 9. **Limitations.** Our model is not aware of individual objects in the scene during the stylization process, thus fail to transfer the style of a particular part of the reference image to the specific object/region in the scene.

References

[1] 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. 5

[2] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. Nerd: Neural reflectance decomposition from image collections. *arXiv preprint arXiv:2012.03918*, 2020. 5

[3] Xu Cao, Weimin Wang, Katashi Nagao, and Ryosuke Nakamura. Psnet: A style transfer network for point cloud stylization on geometry and color. In *WACV*, 2020. 3

[4] Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *CVPR*, 2021. 5

[5] 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. 5

[6] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. *arXiv preprint arXiv:2107.02791*, 2021. 5

[7] Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W. Taylor, and Joshua M. Susskind. Unconstrained scene generation with locally conditioned radiance fields. In *ICCV*, 2021. 5

[8] 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. 5

[9] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *CVPR*, 2021. 5

[10] Chen Gao, Yichang Shih, Wei-Sheng Lai, Chia-Kai Liang, and Jia-Bin Huang. Portrait neural radiance fields from a single image. *arXiv preprint arXiv:2012.05903*, 2020. 5

[11] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380*, 2021. 5

[12] Michelle Guo, Alireza Fathi, Jiajun Wu, and Thomas Funkhouser. Object-centric neural scene rendering. *arXiv preprint arXiv:2012.08503*, 2020. 5

[13] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[14] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM TOG (Proc. SIGGRAPH)*, 36(4), 2017. 1, 3

[15] Adam R. Kosiorek, Heiko Strathmann, Daniel Zoran, Pol Moreno, Rosalia Schneider, Soňa Mokrá, and Danilo J. Rezende. Nerf-vae: A geometry aware 3d scene generative model. In *ICML*, 2021. 5

[16] Tianye Li, Mira Slavcheva, M. Zollhöfer, S. Green, Christoph Lassner, Changil Kim, Tanner Schmidt, S. Lovegrove, M. Goesele, and Z. Lv. Neural 3d video synthesis. *arXiv preprint arXiv:2103.02597*, 2021. 5

[17] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast arbitrary style transfer. In *CVPR*, 2019. 1

[18] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *NIPS*, 2017. 5

[19] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 5

[20] David B. Lindell, Julien N. P. Martel, and Gordon Wetzstein. Autoint: Automatic integration for fast neural volume rendering. In *CVPR*, 2021. 5

[21] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020. 5

[22] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. *arXiv preprint arXiv:2105.06466*, 2021. 5

[23] Stephen Lombardi, Tomas Simon, Gabriel Schwartz, Michael Zollhoefer, Yaser Sheikh, and Jason M. Saragih. Mixture of volumetric primitives for efficient neural rendering. *ACM TOG (Proc. SIGGRAPH)*, 40:1 – 13, 2021. 5

[24] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021. 5

[25] Quan Meng, Anpei Chen, Haimin Luo, Minye Wu, Hao Su, Lan Xu, Xuming He, and Jingyi Yu. GNeRF: GAN-based Neural Radiance Field without Posed Camera. In *ICCV*, 2021. 5

[26] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM TOG (Proc. SIGGRAPH)*, 38:1 – 14, 2019. 1

[27] 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. 5

[28] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 40(4), 2021. 5

[29] Michael Niemeyer and Andreas Geiger. Campari: Camera-aware decomposed generative neural radiance fields. *arXiv preprint arXiv:2103.17269*, 2021. 5

[30] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *CVPR*, 2021. 5

[31] Atsuhiro Noguchi, Xiao Sun, Stephen Lin, and Tatsuya Harada. Neural articulated radiance field. *arXiv preprint arXiv:2104.03110*, 2021. 5

[32] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *CVPR*, 2021. 5

[33] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. In *ICCV*, 2021. 5

[34] 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. 5

[35] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Animatable neural radiance fields for human body modeling. In *ICCV*, 2021. 5

[36] 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. 5

[37] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021. 5

[38] Amit Raj, Michael Zollhoefer, Tomas Simon, Jason Saragih, Shunsuke Saito, James Hays, and Stephen Lombardi. Pva: Pixel-aligned volumetric avatars. In *CVPR*, 2021. 5

[39] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020. 4

[40] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. Derf: Decomposed radiance fields. In *CVPR*, 2021. 5

[41] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *ICCV*, 2021. 5

[42] Konstantinos Rematas, Ricardo Martin-Brualla, and Vittorio Ferrari. Sharf: Shape-conditioned radiance fields from a single view. In *ICML*, 2021. 5

[43] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *CVPR*, 2021. 1

[44] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *NeurIPS*, 2020. 5

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[46] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, 2021. 5

[47] Shih-Yang Su, Frank Yu, Michael Zollhoefer, and Helge Rhodin. A-nerf: Surface-free human 3d pose refinement via neural rendering. *arXiv preprint arXiv:2102.06199*, 2021. 5

[48] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew Davison. iMAP: Implicit mapping and positioning in real-time. *arXiv preprint arXiv:2103.12352*, 2021. 5

[49] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *CVPR*, 2021. 5

[50] 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 deforming scene from monocular video. In *ICCV*, 2021. 5

[51] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d scene representation and rendering. *arXiv preprint arXiv:2010.04595*, 2020. 5

[52] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. 5

[53] Ziyan Wang, Timur Bagautdinov, Stephen Lombardi, Tomas Simon, Jason Saragih, Jessica Hodgins, and Michael Zollhöfer. Learning compositional radiance fields of dynamic human heads. In *CVPR*, 2021. 5

[54] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF−−: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021. 5

[55] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *CVPR*, 2021. 1, 5

[56] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, 2021. 5

[57] Christopher Xie, Keunhong Park, Ricardo Martin-Brualla, and Matthew Brown. Fig-nerf: Figure-ground neural radiance fields for 3d object category modelling. *arXiv preprint arXiv:2104.08418*, 2021. 5

[58] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. iNeRF: Inverting neural radiance fields for pose estimation. In *IROS*, 2021. 5

[59] 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. 5

[60] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 5

[61] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 1, 5

[62] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. NeRFactor: Neural Factorization of Shape and Reflectance Under an Unknown Illumination. *ACM TOG (Proc. SIGGRAPH Asia)*, 2021. 5

[63] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew Davison. In-place scene labelling and understanding with implicit scene representation. In *ICCV*, 2021. 5