StegaNeRF: Embedding Invisible Information within Neural Radiance Fields (Supplementary Material)

Chenxin Li[1]*, Brandon Y. Feng[2]*, Zhiwen Fan[3]*, Panwang Pan[4], Zhangyang Wang[3]
[1]Chinese University of Hong Kong, [2]University of Maryland,
[3]University of Texas at Austin, [4]ByteDance

The video included in the supplementary material is recommended to understand the problem background, motivation, and main results. The code and trained models will be released public.

The following contents are provided in the supplement:

- More experimental details (Sec. 4.1 in main paper).
- More implementation details on 2D steganography methods (Fig. 3, Tab. 1, and Sec. 4.2 in main paper).
- Details of network architecture of StegaNeRF.
- More results, visualization, and explanation about our experiments. (Sec. 4 in the main paper).

A. Detailed Experimental Setup

In both the first and second stage of StegaNeRF training, we use mean squared errors to compute the reconstruction loss for NeRF rendering and hidden recovery. Similar to the common practices in NeRF editing and stylizing [2, 8], we fix the weights regarding geometry components in the NeRF model in the second optimization stage for the steganography objectives.

We train the models including NeRF, the decoder and the classifier using the Adam optimizer with the default parameters. The learning rate for NeRF follows the same strategy in Plenoxel [7] and original NeRF [4]. To train the decoder F_ψ, we adopt the exponential decay learning rate scheduler, with initial learning rate as 10^{-3} and final learning rate as 10^{-4}. We leverage the same exponential decay strategy to adjust the learning rate for classifier F_ψ^c, with the initial value as 10^{-4} and the final one as 10^{-5}. On a machine with NVIDIA A100 GPU, training 100 iterations takes around 1 minute.

We obtain the rendering and recovery performance on each scene by the average results over all the testing views. For the LPIPS error metrics we use the AlexNet implementation at https://github.com/richzhang/PerceptualSimilarity.

*Equal contribution

B. Implementation of 2D Steganography for Comparison

Due to lack of the official code of the 2D steganography methods, we adopt the available PyTorch implementations. The code of LSB [3] can be found at https://github.com/RobinDavid/LSB-Steganography. We directly apply the encoding and decoding functions on the training views of each scene. The code of DeepStega [1] is provided at https://github.com/arnoweng/PyTorch-Deep-Image-Steganography. We apply the provided pretrained encoder and decoder in our experiments. Note that the encoder in DeepStega is set to embed the hidden images into the equal-size carriers. Therefore, we upsample the target hidden images to be the same size as the training views before applying DeepStega. We report the SSIM scores for recovered hidden images after they are downsampled back to the original size.

C. Network Architecture

C.1. Decoder

For the decoder F_ψ, we adopt a U-Net [5] architecture with the VGG-16 [6] backbone to recover the hidden images. For the downsampling brach of this U-Net, the channel numbers are 64, 128, 256, 512, and the bottleneck layer transforms the feature maps from 512 channels to 256 channels. The upsampling branch of the U-Net is a mirrored version of the downsampling brach.

To detect the hidden watermarks beyond image modality, we replace the upsampling brach of the U-Net with a max-pooling layer and two linear layers. These additional layers produce an output with the same shape with the target multi-modality information.

C.2. Classifier

We modify the VGG-16 backbone as the classifier F_ψ^c in StegaNeRF. We additionally introduce a max-pooling layer, two linear layers, and a softmax activation. We set 2 output channels for the single-scene embedding and the multi-

modal scenarios, and 4 output channels when embedding in multiple scenes and multiple identities.

C.3. Quantitative Details of Ground-Truth

Fig. A2 and Fig. A3 demonstrate the ground-truth renderings and hidden images in the experiments shown in the main paper (Fig. 4-7).

C.4. Explanation on "N/A" Results

Here we clarify some experimental results of "N/A" in the main paper. The 50% Acc. of "Standard NeRF" in Tab. 1 and Tab. 3 mean that Standard NeRF can not generate discriminative renderings, so the classification is not better than a binary guess. Meanwhile, the recovered hidden images from such renderings are always meaningless, so the SSIM score computed based on the ground truth images is "N/A". Furthermore, in Tab. 1, LSB and DeepStega have "N/A" Acc. because they do not contain a classifier branch as our method, so their classification results are not available.

C.5. More Visualization from The Entire 3D Space

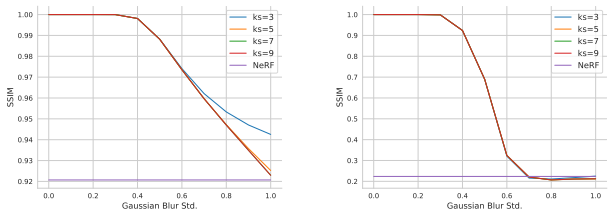
In Fig. A4-A12, we provide more visualization of the StegaNeRF renderings as well as the residual error against initial renderings. We can observe that the regions containing hidden information (salient residual errors) are persistent from the continuous multiple viewpoints in the 3D space.

C.6. Quantitative Details on NeRF-W

Fig. A13 shows each cluster of views in the *Brandenburg Gate* scene from NeRF-W. The views within each cluster possess similar patterns and perspectives, while the inter-cluster views significantly differ from each other.

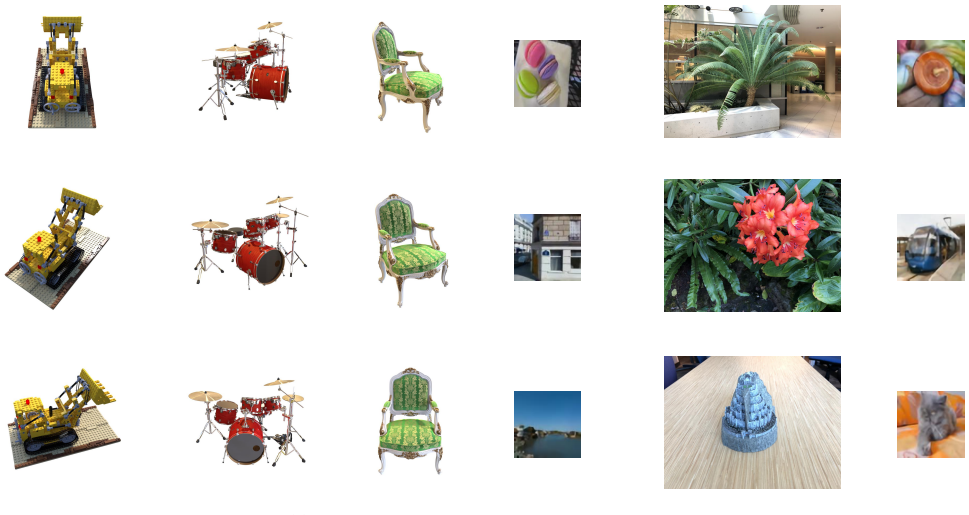
C.7. In-Depth Study on The Failures of 2D steganography

We perform experiments to understand how the hidden information generated by 2D steganography methods gets diminished in the NeRF renderings. Observing the failure of 2D steganography methods after NeRF, we hypothesize that it is mainly due to the smoothing effect brought by NeRF, which makes subtle high-frequency details in the training images hard to be preserved in the rendered output. To verify this, after embedding hidden information in images with 2D steganography methods, we mimic the smoothing effect of NeRF and apply Gaussian blur kernels with various blur strength (different Gaussian standard deviations) and kernel sizes on those images with hidden information. As shown in Fig. A1, the effect of NeRF training and rendering is indeed similar to the effect of applying Gaussian blur with around 0.7-1.0 standard deviation on images with hidden information. Such results reflect the difficulty of directly applying existing steganography methods in NeRF training, and the affirm the necessity of our proposed optimization framework.



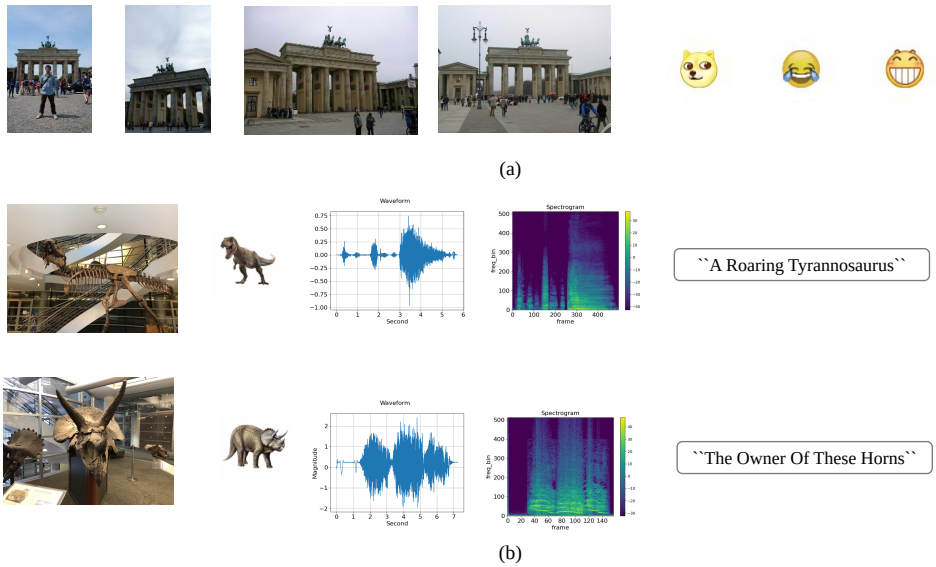
(a) Test Views (b) Recovered Information

Figure A1. **Analysis of the smoothing effect on 2D steganography methods.** After embedding hidden information into images with 2D steganography methods, we train a NeRF on those images with embedded information and render at the test views. We also embed hidden information with 2D methods into the ground truth images at test views, and we directly apply the Gaussian blurs on those images. We then compare the SSIM of (a) test views and (b) recovered hidden information, from NeRF renderings and from the blurred embedded images. We observe that the impact of NeRF training and rendering is comparable to directly blurring the embedded images with a kernel standard deviation around 0.7-1.0.



(a) (b)

Figure A2. Ground-truth images of renderings and hidden watermarks for (a) Fig. 4 and (b) Fig. 5 in in our main paper.



(a) (b)

Figure A3. Ground-truth (images and other multi-modal signals) of renderings and multi-modal signals hidden watermarks for (a) Fig. 6 and (b) Fig. 7 in our main paper.

Figure A4. All the testing views rendered by StegaNeRF in *lego* scene.

Figure A5. The residual error (magnified by 5 times) of StegaNeRF renderings against initial ones on all the testing views in *lego* scene.

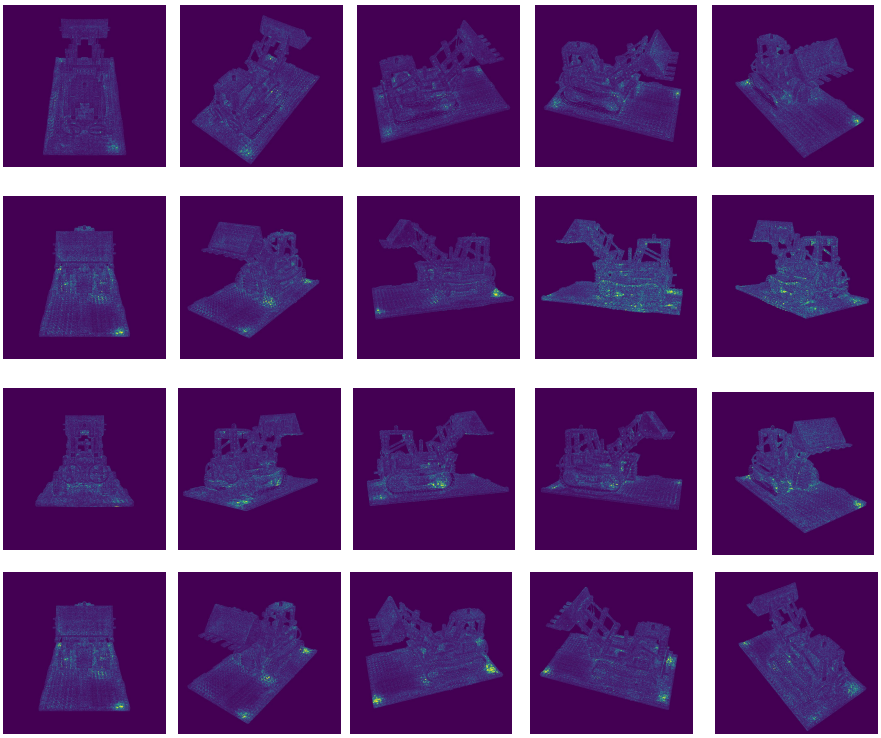


Figure A6. The residual error (magnified by 25 times) of StegaNeRF renderings against initial ones on all the testing views in *lego* scene.

Figure A7. All the testing views rendered by StegaNeRF in *drums* scene.

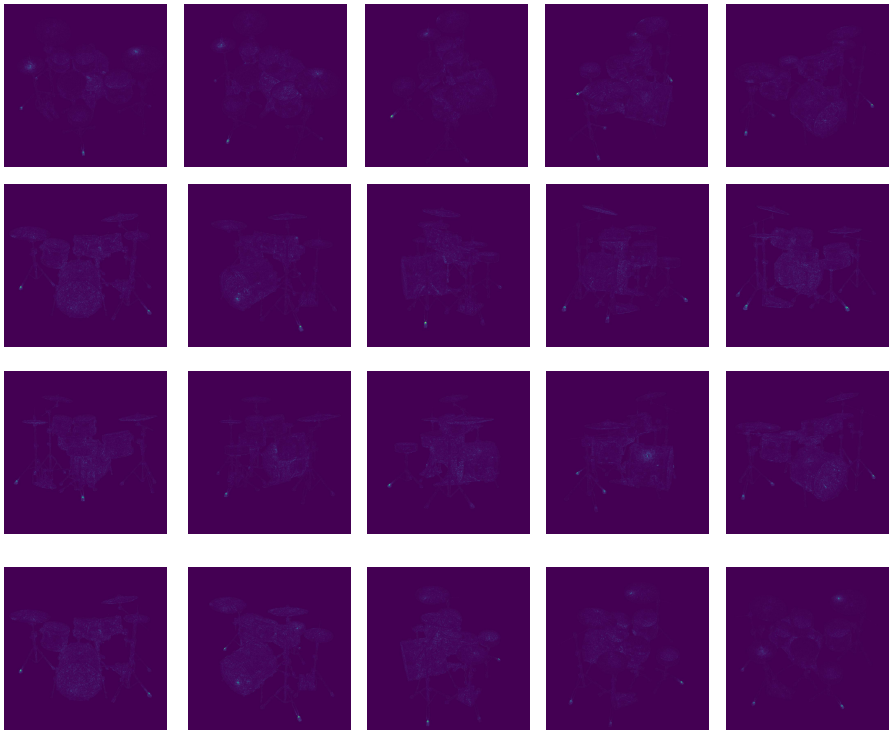


Figure A8. The residual error (magnified by 5 times) of StegaNeRF renderings against initial ones on all the testing views in *drums* scene.

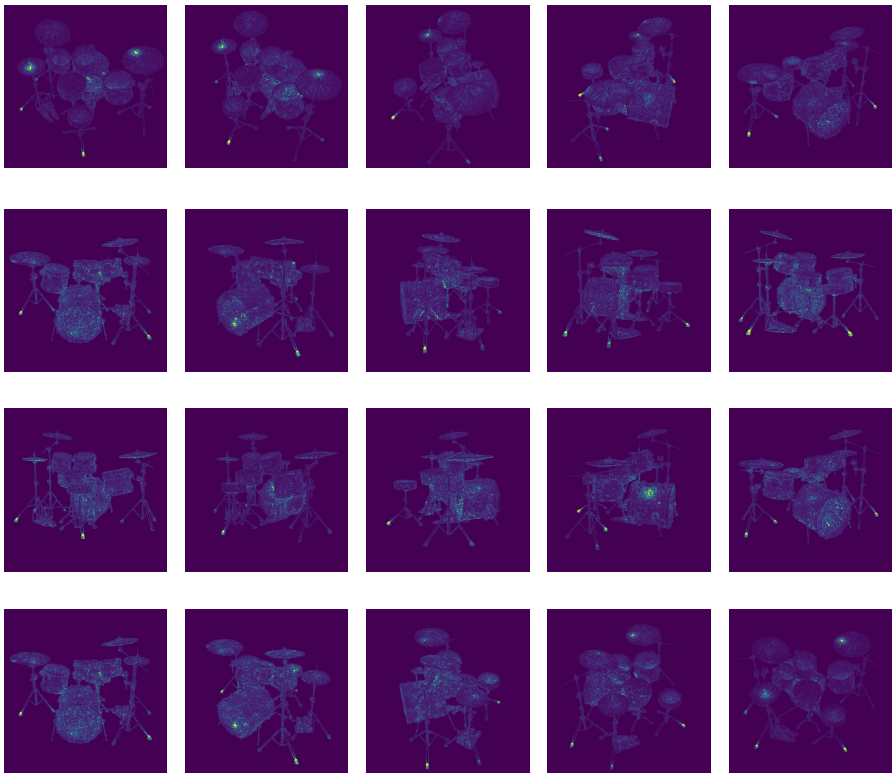


Figure A9. The residual error (magnified by 25 times) of StegaNeRF renderings against initial ones on all the testing views in *drums* scene.

Figure A10. All the testing views rendered by StegaNeRF in *chair* scene.

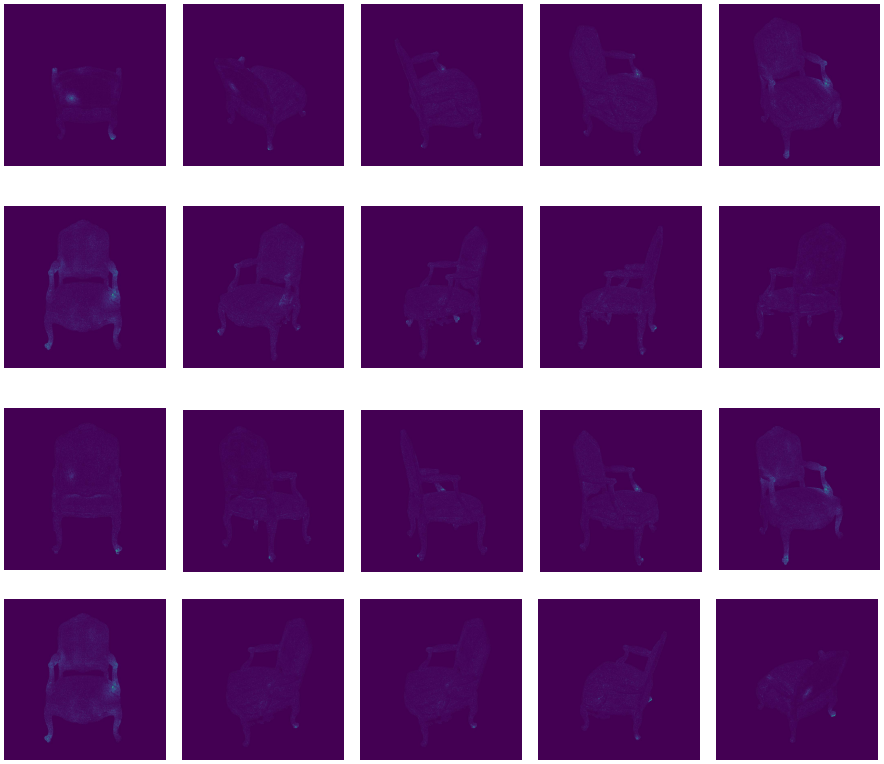


Figure A11. The residual error (magnified by 5 times) of StegaNeRF renderings against initial ones on all the testing views in *chair* scene.

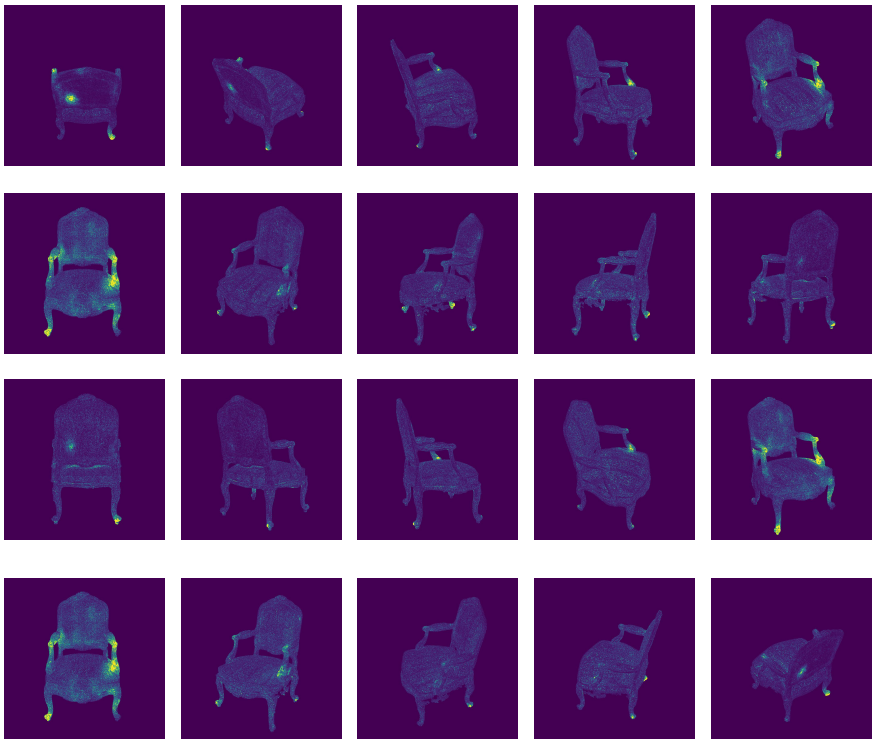


Figure A12. The residual error (magnified by 25 times) of StegaNeRF renderings against initial ones on all the testing views in *chair* scene.

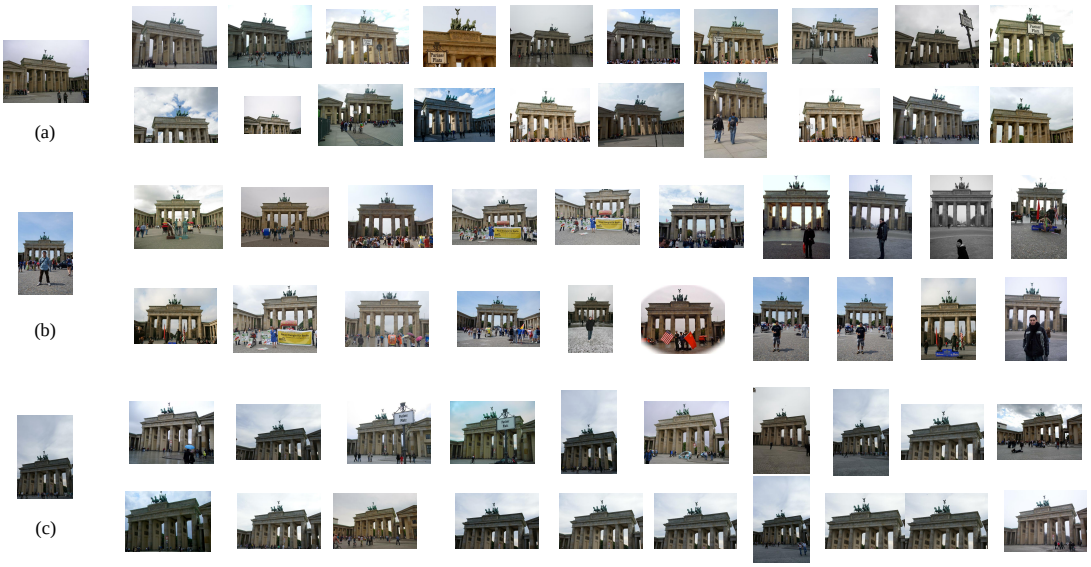


Figure A13. The selected views for each cluster (user) as (a), (b) and (c). In each block, the first column indicates the anchor image while the rest ones are the searched KNN views with the anchor one.

References

[1] Shumeet Baluja. Hiding Images in Plain Sight: Deep Steganography. *Advances in Neural Information Processing Systems*, 30, 2017. 1

[2] Zhiwen Fan, Yifan Jiang, Peihao Wang, Xinyu Gong, Dejia Xu, and Zhangyang Wang. Unified implicit neural stylization. In *European Conference on Computer Vision*, pages 636–654. Springer, 2022. 1

[3] Shailender Gupta, Ankur Goyal, and Bharat Bhushan. Information Hiding Using Least Significant Bit Steganography and Cryptography. *International Journal of Modern Education and Computer Science*, 4(6):27, 2012. 1

[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. *Communications of the ACM*, 65(1):99–106, 2021. 1

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1

[6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[7] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance Fields Without Neural Networks. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5491–5500, 2022. 1

[8] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *European Conference on Computer Vision*, pages 717–733. Springer, 2022. 1