NeRFrac: Neural Radiance Fields through Refractive Surface Supplemental Material

Yifan Zhan* Shohei Nobuhara† Ko Nishino† Yinqiang Zheng*
*The University of Tokyo, Japan †Kyoto University, Japan
zhan-yifan@g.ecc.u-tokyo.ac.jp, {nob, kon}@i.kyoto-u.ac.jp, yqzheng@ai.u-tokyo.ac.jp

1. Transformation between NDC and Camera Coordinate

We start our calculation following the derivation in Appendix of NeRF [6]:

$$\mathbf{o}' = \begin{pmatrix} -\frac{f_{cam}}{W/2}\frac{o_x}{o_z} \\ -\frac{f_{cam}}{H/2}\frac{o_y}{o_z} \\ 1 + \frac{2n}{o_z} \end{pmatrix}, \tag{1}$$

$$\mathbf{d}' = \begin{pmatrix} -\frac{f_{cam}}{W/2}\left(\frac{d_x}{d_z} - \frac{o_x}{o_z}\right) \\ -\frac{f_{cam}}{H/2}\left(\frac{d_y}{d_z} - \frac{o_y}{o_z}\right) \\ -2n\frac{1}{o_z} \end{pmatrix}, \tag{2}$$

where W and H are the width and height of the image and f_{cam} is the focal length of the pinhole camera. In our experiments, we set the near clipping plane to $n = -1$. o and d are the ray origin and the ray direction in the camera coordinate frame, and o' and d' are the ray origin and the ray direction in NDC respectively. To derive d, we first define intermediate variables

$$\begin{cases} t_1 = \dfrac{o_x}{o_z} = -\dfrac{W{o'}_x}{2f_{cam}}, \\ t_2 = \dfrac{o_y}{o_z} = -\dfrac{H{o'}_y}{2f_{cam}}, \\ t_3 = \dfrac{2n}{o_z} = {o'}_z - 1\,. \end{cases} \tag{3}$$

Given these variables, we can calculate the ratios d_x/d_z and d_y/d_z from d' by combining Eqs. (2) and (3)

$$\begin{cases} \dfrac{d_x}{d_z} = -\dfrac{W{d'}_x}{2f} + t_1, \\ \dfrac{d_y}{d_z} = -\dfrac{H{d'}_y}{2f} + t_2, \end{cases} \tag{4}$$

and we can obtain d based on the unit vector constraint.

2. Local Normal Calculation with QR Decomposition

Recall that in Section 4.3 we minimize $J(w)$ and obtain the optimal closed-form solution of w using the pseudo-inverse

$$w^* = (P_{xy}{}^{\mathrm{T}} P_{xy})^{-1} P_{xy}{}^{\mathrm{T}} P_z \,. \tag{5}$$

This is in the form of

$$w^* = (A^{\mathrm{T}} A)^{-1} A^{\mathrm{T}} B \,, \tag{6}$$

where A is an $m \times 2$ matrix and B is an $m \times 1$ vector. Although Eq. (6) is very simple, its direct computation can result in a large numerical error due to the complexity of the matrix and the accumulation of numerical errors in floating point computation. The impact of this can be measured by the condition number. The condition number of the matrix measures the ratio of the maximum relative stretching to the maximum relative shrinking that matrix applies to any non-zero vectors, and is defined as [5]

$$\mathrm{condi}(A) = \|A\| \left\|A^{-1}\right\| \,. \tag{7}$$

In Eq. (6) we try to calculate the inverse matrix of $A^{\mathrm{T}} A$, whose condition number is

$$\mathrm{condi}(A^{\mathrm{T}} A) = \mathrm{condi}(A)^2 \,. \tag{8}$$

As the condition number of matrix A increases, the condition number of the inverse matrix we compute increases quadratically. This is detrimental to the accuracy of our calculations. We reduce the condition number of the inverse matrix to be calculated by applying QR decomposition. For a real matrix A, it can be decomposed as the product of an orthogonal matrix Q and an upper triangular matrix R. By decomposing the matrix A, we can simplify Eq. (6) as

$$\begin{aligned} w^* &= (A^{\mathrm{T}} A)^{-1} A^{\mathrm{T}} B \,, \\ A^{\mathrm{T}} A w^* &= A^{\mathrm{T}} B \,, \\ R^{\mathrm{T}} Q^{\mathrm{T}} Q R w^* &= R^{\mathrm{T}} Q^{\mathrm{T}} B \,, \\ R^{\mathrm{T}} R w^* &= R^{\mathrm{T}} Q^{\mathrm{T}} B \qquad (Q^{\mathrm{T}} Q = I) \,, \\ R w^* &= Q^{\mathrm{T}} B \,, \\ w^* &= R^{-1} Q^{\mathrm{T}} B \,. \end{aligned} \tag{9}$$

Now we only need to calculate the inverse matrix of R, whose condition number should be

$$\mathrm{condi}(R) = \mathrm{condi}(A) \,. \tag{10}$$

By applying the QR decomposition, we successfully reduce the condition number from $\mathrm{condi}(A)^2$ to $\mathrm{condi}(A)$. This improves not only the accuracy of the calculation but also the speed, which allows us to train NeRFrac faster.

3. Additional Training Details

NeRF introduces a hierarchical volume sampling strategy. We follow this process and control the N_sample=64 and N_importance=64 during the comparison experiment for each baseline. We train our NeRFrac on a single GeForce RTX 3090, which takes around 200k iterations (3 hours) to converge (NeRF takes 2 hours with the same configurations).

4. Our Reproduction of LB-NeRF [4]

We showed comparisons with LB-NeRF [4] in the main text. As it is a new work with no code available to the public, we reproduced this work in PyTorch. Fig. 1 shows our reproduced pipeline. We train our reproduced LB-NeRF for 200k iterations with N_sample=64 and N_importance=64.

The visual coherence of the images generated by LB-NeRF is relatively good, as there are fewer blurs and artifacts compared with the original NeRF. This is likely because the network tries to learn a smooth offset for each 3D point. However, quantitative comparison in our main text tells that the rendered images of LB-NeRF do not perform as well as our model in terms of accuracy. In our opinion, LB-NeRF lacks real physical guide, which makes the network try to simply smooth the images and cannot maintain the consistency among multiple views.

We further demonstrate this by visualizing the sampled points after deformation in LB-NeRF (Fig. 3). Rays are emitted from plane $z = -1$ to $z = 1$, and the red box shows how rays bend, especially near fine sampling area. Because LB-NeRF lacks real physical constraints, rays are bent in a smooth way, but in a manner physically implausible.

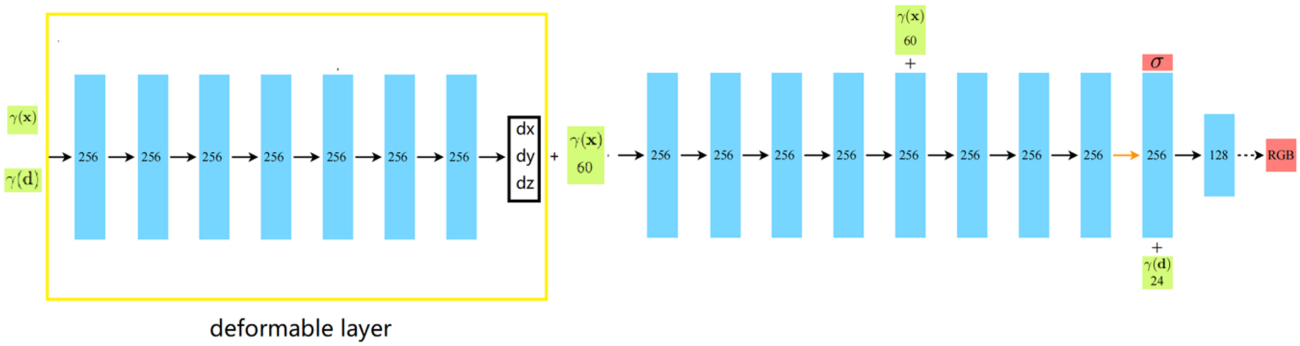

Figure 1: Our implementation of LB-NeRF.

Scene	PSNR↑/SSIM↑/LPIPS↓/Average↓	
	Qian's	NeRFrac
gloves	26.02 / 0.831 / 0.167 / 0.056	**27.12 / 0.833 / 0.163 / 0.051**
toy	26.26 / 0.709 / 0.168 / 0.060	**28.55 / 0.823 / 0.154 / 0.045**

Table 1: Quantitative comparison with Qian's method.

5. Comparison with Qian's Method [8]

The comparison here is kindly conducted by the authors of [8] on their data. We show two scenes, respectively, "gloves" and "toy". In their setting, the underwater scenes were captured by 9 synchronized cameras and tested by another evaluation camera. First, cameras are calibrated with COLMAP [9]. Then, our method is applied on their data with identical camera poses. See Fig. 4 and Tab. 1 for test results.

6. Comparison with Eikonal Fields (EiF) [2]

Although both focus on novel view synthesis of refracted scenes, there are two main differences between our setting and the setting of EiF [2].

First, EiF reconstructs an opaque scene using a non-Eikonal emission-absorption model with straight rays, assuming that the opaque scene conforms to the straight ray sampling assumption in NeRF [6]. In our data, however, the refractive surface covers the whole scene, making the non-Eikonal emission-absorption model in EiF no longer accurate. Second, EiF models the remaining refractive part using an Eikonal formulation to learn the index of refraction (IOR) field. However, this procedure can be very hard on our data, where the refractive media and the scenes are completely mixed together.

In Fig. 5, we detail our rendered result with EiF. Since the setting are very different, EiF cannot converge on our data when learning the IOR field. For this, we omit further quantitative comparison.

7. Our Dataset and More Comparisons

Our synthetic data simulate different waveforms and underwater scenes. The code will be accessible for everyone to fit their own needs. Our real dataset contains 10 multi-view underwater image sequences, each including the full range of water surfaces from flat to fluctuating. We use our synchronous camera array to take multi-frame images continuously at 5fps. Therefore, these real data can be used for both multi-view tasks and video tasks. We show more qualitative and quantitative comparisons of different scenes below (part of our dataset). The "hybrid sine" data has a hybrid refractive surface which is $A + B$ (see Equation 10 in our main text). See Fig. 6 and Tab. 2 for more details. In Fig. 2, we also show our water surface reconstruction result on data "hybrid sine" (using RMSE as metrics).

8. Extra Design for Underwater Video Task

As discussed in Section 5, our method can be easily adapted to deal with continuous frames. To prove this, we further design a time encoded Refractive Field, which allows us to superresolve low-speed videos in the temporal domain. What's

	PSNR↑	SSIM↑	LPIPS↓	Average↓
Synthetic data (hybrid sine)				
NeRF[6]	20.78	0.704	0.390	0.121
Mip-NeRF[1]	20.98	0.681	0.459	0.127
IBRNet[10]	21.30	0.715	0.344	0.111
Tensorf[3]	20.71	0.622	0.450	0.133
Plenoxels[11]	16.95	0.313	0.633	0.220
LB-NeRF[4]	20.00	0.680	0.372	0.128
NeRFrac(ours/Ψ_D)	31.20	0.918	0.192	0.035
NeRFrac(ours/Ψ_R)	**33.45**	**0.940**	**0.159**	**0.026**
Real data (fish)				
NeRF[6]	27.66	0.881	0.219	0.051
Mip-NeRF[1]	28.21	0.867	0.238	0.051
IBRNet[10]	**29.37**	0.916	0.143	0.036
Tensorf[3]	27.91	0.873	0.150	0.044
Plenoxels[11]	20.89	0.576	0.431	0.132
LB-NeRF[4]	28.16	0.889	0.190	0.046
NeRFrac(ours/Ψ_D)	26.95	0.874	0.194	0.052
NeRFrac(ours/Ψ_R)	29.32	**0.918**	**0.118**	**0.034**
Synthetic data (second sine)				
NeRFrac(ours/Ψ_D)	34.49	0.943	0.151	0.023
NeRFrac(ours/Ψ_R)	**34.98**	**0.948**	**0.142**	**0.022**
Synthetic data (primary sine)				
NeRFrac(ours/Ψ_D)	32.54	0.930	0.170	0.029
NeRFrac(ours/Ψ_R)	**34.38**	**0.945**	**0.148**	**0.023**
Real data (plant)				
NeRFrac(ours/Ψ_D)	25.58	0.836	0.235	0.064
NeRFrac(ours/Ψ_R)	**28.29**	**0.883**	**0.153**	**0.043**
Real data (tree)				
NeRFrac(ours/Ψ_D)	27.15	0.740	0.437	0.075
NeRFrac(ours/Ψ_R)	**31.20**	**0.871**	**0.222**	**0.039**

Table 2: More quantitative comparison for our synthetic and real data.

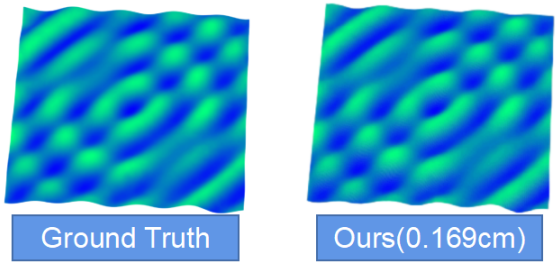

Figure 2: Water surface reconstruction result on "hybrid sine" data.

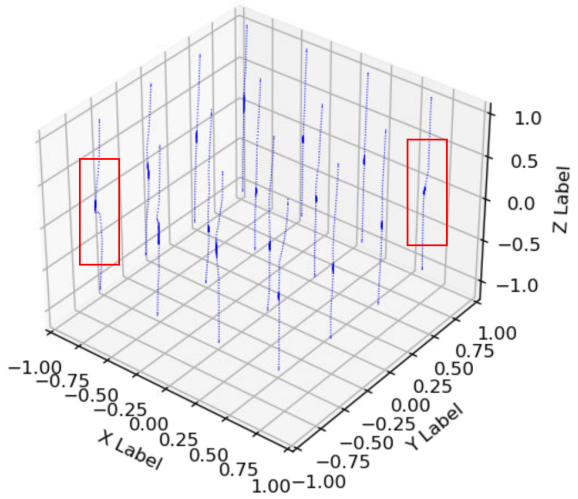

Figure 3: Sampled points after deformation in LB-NeRF. The red box we draw shows how rays bend, especially near fine sampling area.

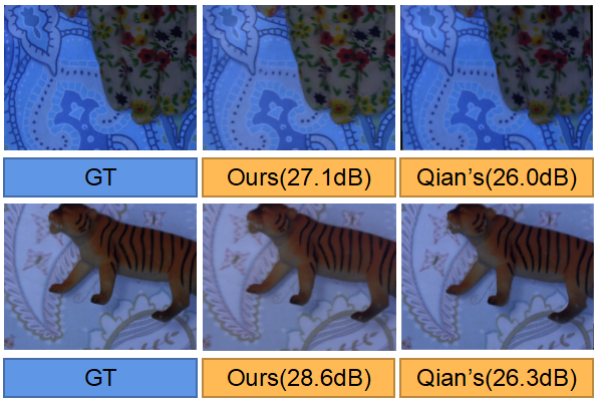

Figure 4: Qualitative comparison with Qian's method on their data "gloves" and "toy". We show the PSNR of the rendered images within parenthesis (higher is better).

Figure 5: Our data tested with EiF [2]. Artifacts and blurs in the left image show that the non-Eikonal step in EiF fails on our data. In the middle is the rendered image after learning IOR field. And the right is the rendered image after learning the radiance field for the object inside the transparent surface.

Figure 6: More qualitative comparisons for our synthetic and real data. From top to bottom: "second sine", "primary sine", "hybrid sine", "plant", "tree", and "fish". Please zoom in to see more details.

more, to model dynamic underwater scene, we follow the time encoded Refractive Field with a Motion Deform Field, with the same implementation in [7].

Our total inputs are $N_{cam} \times N_t$ images and their corresponding camera parameters, where N_{cam} is the number of training cameras ($N_{cam} = 8$) and N_t is the number of frames. Based on Refractive Field Ψ_R, we further query t (timestamp of each frame) to output time-related refractive surface depth. Next we apply Snell's Law to sample the refracted ray. Our Motion Deform Field queries 3D spatial location of these sampled points and t to estimate an offset for each point, which will be used by Radiance Field for radiance and volume density calculation.

We show our pipeline in Fig. 7. Notice that our time encoded Refractive Field takes into account both multi-views and temporal continuity, and thus effectively overcomes color aberrations and jitter which can easily happen in single frame training, allowing continuous interpolation on both time and space. Please watch our supplemental video for more visualization on this extension.

9. Design of Our Depth Map Network Ψ_D

Recall that in Section 6, we experimentally replace Ψ_R with a depth map network Ψ_D which requires redundant sampling for accurate intersection calculation between rays and the refractive surface. We design Ψ_D to illustrate the difficulty in

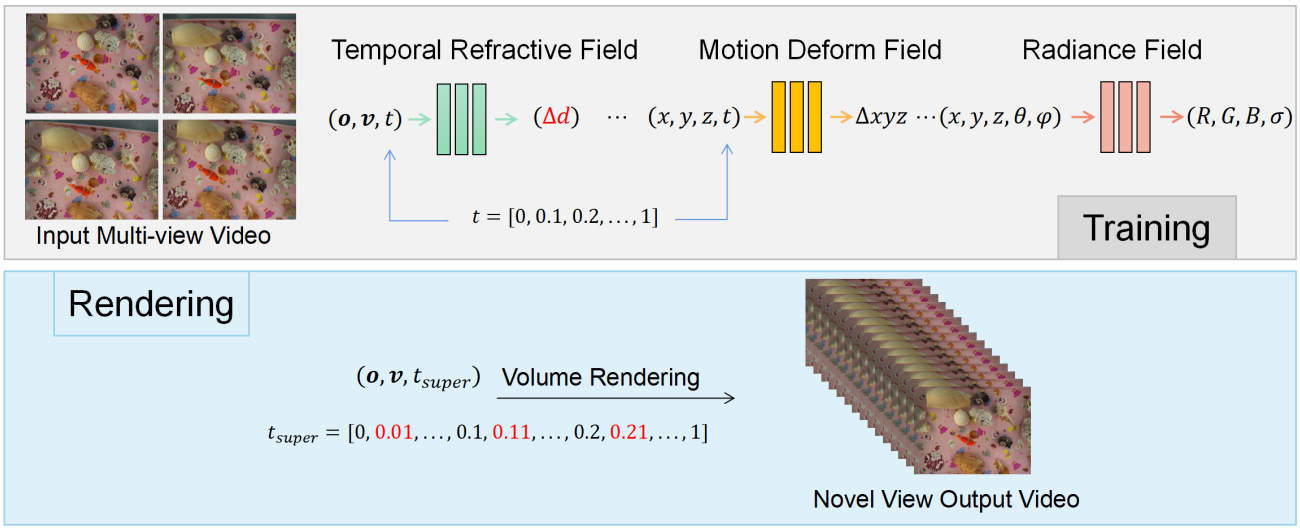

Figure 7: Pipeline for underwater video rendering.

gradient back propagation with such redundant sampling. Here we show the implementation of our Ψ_D.

In Section 4.1, we first transform all rays from the camera coordinate system to the NDC frame. Our Ψ_D only queries ray origins o and outputs estimated depth d' (we denote this as d' to distinguish it from the d output by Ψ_R), which is the distance between origins and the refractive surface along the z axis of the NDC frame. Given a certain ray, We first find the projection of this ray on the NDC near plane $z = -1$, and sample the projected 2D line to get a series of o: $o_{sample} = \{(u_i, v_i)\}_{i=1,2...,N_{sample}}$, where N_{sample} is the number of sampled points. For each 2D point (u, v), we estimate the depth d' with our Ψ_D. Also we have the z value of the ray, denoted by z_r. To find the intersection between the ray and the refractive surface, we are actually calculating

$$(u_s, v_s) = \arg\min_{u,v} \ |d'(u, v) - z_r(u, v)|. \tag{11}$$

Now we have the intersection point $X_s = (u_s, v_s, \Psi_D(u_s, v_s))$. The next step is to follow the steps from Section 4.2 to update our network. As can be seen in Eq. (11), only one of o_{sample} will be used for calculating X_s, making it difficult to update Ψ_D. Fig. 8 shows the comparison between our Ψ_R and Ψ_D.

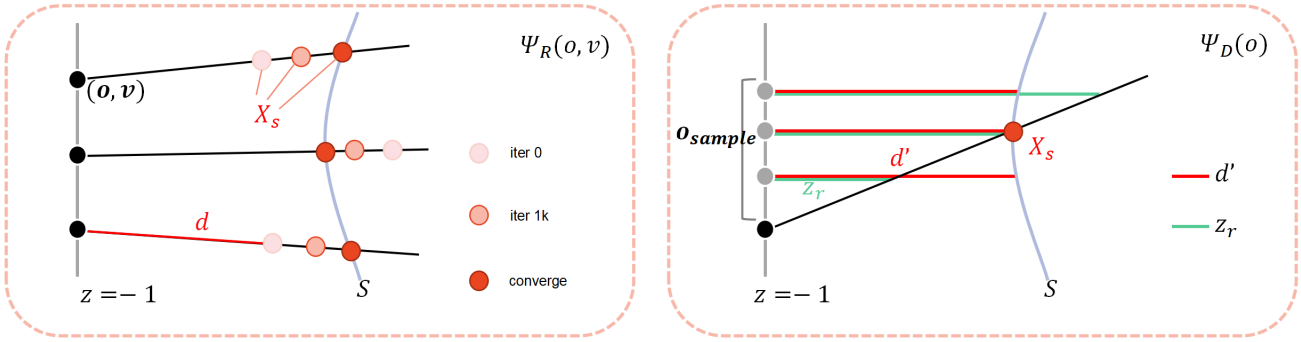

Figure 8: Comparison between Ψ_R and Ψ_D.

10. Our Supplemental Video

We have included a video in the supplementary material. The video consists of 6 parts: 1) we show novel view videos in spiral to compare our method with NeRF [6] and LB-NeRF [4]; 2) we demonstrate the effect of removing the refractive

surface compared with LB-NeRF [4]; 3) we achieve refractive surface modification by swapping the Refractive Field Ψ_R from two different underwater scenes; 4) we show our temporal super resolution result, described in Sec. 8. We use multi-view video in 5fps for training and finally interpolate it to video in 50 fps; 5) we first show the effect of removing the water surface from a dynamic underwater scene with swimming fish. Then, we add a temporal Refractive Field designed in Sec. 8 to render the dynamic scene under wavy water surface; 6) we further display our rendered dynamic underwater video in spiral. Readers are encouraged to watch the supplemental video for a more intuitive understanding of this effect.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 4

[2] Mojtaba Bemana, Karol Myszkowski, Jeppe Revall Frisvad, Hans-Peter Seidel, and Tobias Ritschel. Eikonal Fields for Refractive Novel-View Synthesis. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022. 3, 4

[3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 333–350. Springer, 2022. 4

[4] Taku Fujitomi, Ken Sakurada, Ryuhei Hamaguchi, Hidehiko Shishido, Masaki Onishi, and Yoshinari Kameda. LB-NERF: Light Bending Neural Radiance Fields for Transparent Medium. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 2142–2146. IEEE, 2022. 2, 4, 6, 7

[5] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013. 2

[6] 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, 3, 4, 6

[7] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 5

[8] Yiming Qian, Yinqiang Zheng, Minglun Gong, and Yee-Hong Yang. Simultaneous 3d Reconstruction for Water Surface and Underwater Scene. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 754–770, 2018. 3

[9] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion Revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 3

[10] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning Multi-view Image-based Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021. 4

[11] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance Fields without Neural Networks. *arXiv preprint arXiv:2112.05131*, 2021. 4