WaterSplatting: Fast Underwater 3D Scene Reconstruction Using Gaussian Splatting

Huapeng Li
University of Zurich
huapeng.li@uzh.ch

Wenxuan Song
ETH Zurich
wesong@student.ethz.ch

Tianao Xu
ETH Zurich
tianaxu@student.ethz.ch

Alexandre Elsig
ETH Zurich
elsiga@student.ethz.ch

Jonas Kulhanek
CTU in Prague, ETH Zurich
jonas.kulhanek@cvut.cz

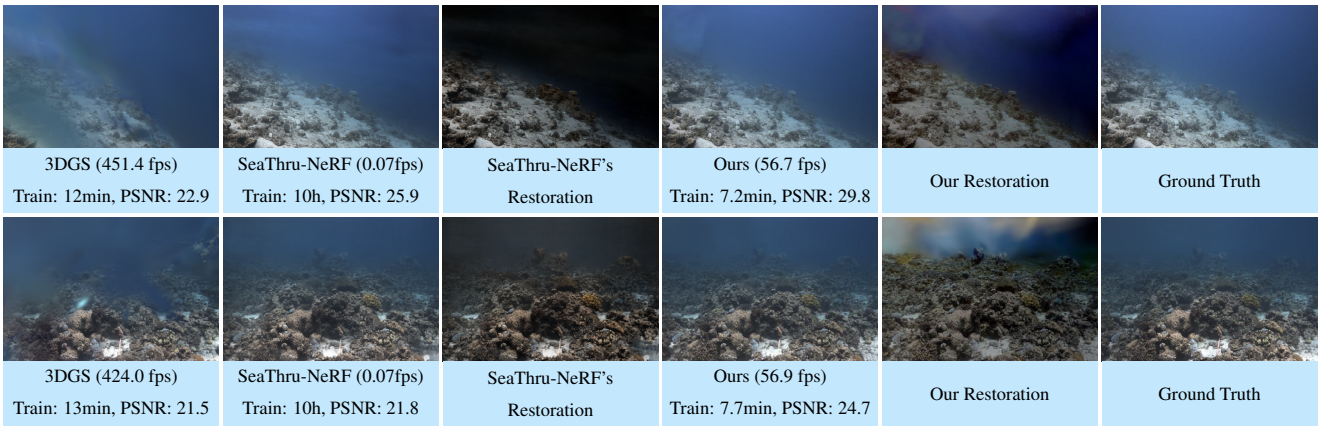

Figure 1. Our approach surpasses the performance of state-of-the-art NeRF-based underwater reconstruction methods [18] while offering real-time rendering speed [15].

Abstract

The underwater 3D scene reconstruction is a challenging, yet interesting problem with applications ranging from naval robots to VR experiences. The problem was successfully tackled by fully volumetric NeRF-based methods which can model both the geometry and the medium (water). Unfortunately, these methods are slow to train and do not offer real-time rendering. More recently, 3D Gaussian Splatting (3DGS) method offered a fast alternative to NeRFs. However, because it is an explicit method that renders only the geometry, it cannot render the medium and is therefore unsuited for underwater reconstruction. Therefore, we propose a novel approach that fuses volumetric rendering with 3DGS to handle underwater data effectively. Our method employs 3DGS for explicit geometry representation and a separate volumetric field (queried once per pixel) for capturing the scattering medium. This dual representation further allows the restoration of the scenes by removing the scattering medium. Our method outperforms state-of-the-art NeRF-based methods in rendering quality on the underwater SeaThru-NeRF dataset. Furthermore, it does so while offering real-time rendering performance, addressing the efficiency limitations of existing methods.
Web: https://water-splatting.github.io

1. Introduction

Neural Radiance Fields (NeRFs) [24] have recently gained significant popularity due to their ability to offer photorealistic 3D scene reconstruction quality. This has opened up new avenues in the field of 3D rendering and reconstruction. However, the landscape of 3D rendering techniques is rapidly evolving. More recently, point splatting methods have experienced a resurgence in the form of 3D Gaussian Splatting (3DGS) [15], which matches NeRFs in terms of rendering quality and offers real-time rendering speed, better editability, and control.

The reconstruction of scattering scenes, such as foggy and underwater environments, is an interesting research area with applications ranging from naval robots to VR ex-

periences. Reconstructing geometry inside a water volume is challenging due to the presence of the scattering medium with properties different from air. In a typical scene, the primary requirement is to represent the surface. Both NeRFs and Gaussian splatting methods are optimized to focus on representing the surfaces only, thereby gaining better efficiency. In the case of NeRFs, since they are fully volumetric, they should theoretically be able to represent the medium fully volumetrically. However, this is no longer the case as the proposal sampler used to speed up NeRFs prevents them from learning volumes well.

To address this issue, a NeRF approach, SeaThru-NeRF [18], was proposed, which uses two fields: one for the geometry and one for the volume in between. However, it is slow in both rendering and training. Therefore, we propose a novel approach to represent the geometry explicitly using 3DGS but to represent the volume in between using a volumetric representation. The renderer we propose not only surpasses the rendering quality of fully volumetric representations, as demonstrated by [18] but also achieves rendering and training speeds comparable to 3DGS.

To validate our method, we evaluate it on the established benchmark underwater dataset - SeaThru-NeRF [18]. The results of our evaluation demonstrate the effectiveness of our proposed method in achieving high-quality, efficient underwater reconstruction. In summary, we make the following contributions:

1. **Splatting with Medium:** We introduce a novel approach that combines the strengths of Gaussian Splatting (GS) and volume rendering. Our method employs GS for explicit geometry representation and a separate volumetric field for capturing the scattering medium. This dual representation allows for the synthesis of novel views in scattering media and the restoration of clear scenes without a medium.

2. **Loss Function Alignment:** We propose a novel loss function designed to align 3DGS with human perception of High Dynamic Range (HDR) and low-light scenes.

3. **Efficient Synthesis and Restoration:** We demonstrate that our method outperforms other models on synthesizing novel view on real-world underwater data and restoring clean scenes on synthesized back-scattering scenes with much shorter training and rendering time.

2. Related Work

2.1. NeRF

The field of 3D scene reconstruction has gained significant attention with the advent of NeRF methods [23, 24, 34]. NeRFs represent the 3D scene as a radiance field—comprising differential volume density and view-dependent color—rendered using a volume rendering integral from a list of samples sampled along the ray [29]. Originally, NeRFs utilized Multilayer Perceptrons (MLPs) for representing the radiance field [3, 4, 24], but they were slow to train and render. To accelerate the training and rendering processes, alternative methods have been proposed using discrete grids [11, 43], hash grids [5, 26, 36], tensorial decomposition [9, 31], point clouds [40], or tetrahedral mesh [17]. NeRFs have been enhanced in various ways, including improved anti-aliasing [3, 5], handling of large 3D scenes [35], and complex camera trajectories [4, 38]. Moreover, NeRFs have been extended to a wide range of applications such as semantic segmentation [6, 16], few-view novel view synthesis [7, 8, 20, 39, 44], and generative 3D modeling [22, 27]. Despite these advancements, the slow rendering speed of NeRFs remains a critical limitation, hindering their widespread adoption on end-user devices.

2.2. 3D Gaussian Splatting

Recently, Gaussian Splatting (3DGS) [15] has seen a resurgence as a powerful method for real-time 3D rendering, matching the quality of Neural Radiance Fields (NeRFs) [24] but with significantly faster speeds even suitable for end-user devices [15]. This technique enhances control and editability since scenes are stored as editable sets of Gaussians, allowing for modifications, merging, and other manipulations. Additionally, the original 3DGS method has been refined to improve anti-aliasing [45] and adapt density control more effectively [42]. Owing to these advancements, 3DGS has been widely adopted in various applications such as large-scale reconstructions [21], 3D generation [10], simultaneous localization and mapping (SLAM) [14, 19], and open-set segmentation [28]. Despite its state-of-the-art rendering quality and impressive handling of complex scenes, 3DGS's explicit representation nature limits its use in scenarios requiring the depiction of semi-transparent volumes, such as underwater reconstructions where light scattering and absorption are significant challenges [15].

2.3. Computer Vision in Scattering Media

There are many challanges in underwater computer vision. The complex lighting conditions including scattering and attenuation of light leading to distorted images and the failure of traditional algorithms trained on clear-air scenes [12]. WaterNeRF [33] estimates the medium parameters separately from the rendering and works with images that have been histogram-equalized. In our method we directly modify the rendering equation. ScatterNeRF [30] uses a volumetric representation of the scene, by extending the NeRF rendering equation to include scattering properties of the medium for adverse weather environments, with a separation of the backscatter component of the image. Our method is more specialized for underwater and foggy scenes. SeaThru [2] introduces a method for removing wa-

ter from underwater images. This method addresses color distortion in underwater images by revising the image formation model from [1], accurately estimating backscatter, and correcting colors along the depth axis. SeaThru-NeRF [18] implements the image formation model [1] that separates direct and backscatter components, into the NeRF rendering equations, which is highly specialized for underwater scenes. We implemented a similar model but on Gaussian Splatting, which yields higher performance and enables real-time rendering.

3. Method

We start by briefly reviewing 3DGS and the rendering model in scattering media in Sec. 3.1. Then, we illustrate our proposed rendering model combining 3DGS with medium encoding in Sec. 3.2. At last, we explain our proposed loss function to align 3DGS with human perception of HDR scenes in Sec. 3.3.

3.1. Preliminaries

3D Gaussian Splatting models a scene with explicit learnable primitives $\mathcal{G}_0$, $\mathcal{G}_1$, ..., $\mathcal{G}_N$. Each Gaussian $\mathcal{G}_i$ is defined by a central position μ_i and covariance matrix Σ_i [15]:

$$G_i(p) = e^{-\frac{1}{2}(p-\mu_i)^T(\Sigma_i)^{-1}(p-\mu_i)}. \quad (1)$$

3DGS primitive also has two additional parameterized properties: opacity o_i and spherical harmonics coefficients SH_i to represent directional appearance component (anisotropic color). In order to render pixel-wise color, primitives are transformed into camera space via a viewing transformation W and the Jacobian of the affine approximation of the projective transformation J on Σ_i, then we get the projected 2D means $\hat{\mu}_i$ and 2D covariance matrices $\hat{\Sigma}_i$ as:

$$\hat{\Sigma}_i = (JW\Sigma_i W^T J^T)_{1:2,1:2}\,, \quad \hat{\mu}_i = (W\mu_i)_{1:2}\,, \quad (2)$$

and the depth of $\mathcal{G}_i$ on z-coordinate:

$$s_i = (W\mu_i)_3. \quad (3)$$

The Gaussian kernel $\hat{G}_i$ of 2D Gaussian is represented as:

$$\hat{G}_i(p) = e^{-\frac{1}{2}(p-\hat{\mu}_i)^T(\hat{\Sigma}_i)^{-1}(p-\hat{\mu}_i)}, \quad (4)$$

where p is the coordinate of the pixel. For rasterization, each Gaussian is truncated at 3 sigma, considering only those intersecting with the patch comprising 16×16 pixels within this range, based on the property that about 99.7% of the probability lies within 3 sigma of the mean. Thus, the pixel color is computed by alpha blending of the sorted intersected Gaussians $\mathcal{G}_i$ whose α_i are higher than a threshold:

$$C = \sum_{i=1}^{N} c_i \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j)\,, \quad \alpha_i = \sigma(o_i)\cdot \hat{G}_i(p)\,, \quad (5)$$

where c_i is the color given the view direction, $\sigma(\cdot)$ is the Sigmoid function and N is the number of Gaussians involved in alpha blending. During optimization, 3DGS periodically densify Gaussians with high average gradient on 2D coordinates $\hat{\mu}_i$ across frames via splitting large ones and duplicating small ones. In the meantime, 3DGS prunes primitives with low opacity for acceleration and periodically set α_i close to zero for all Gaussians to moderate the increase of floaters close to the input cameras.

For **scene rendering in scattering media** we use the revised underwater image formation model from [1] where the final image I is separated into a direct and backscatter component

$$I = \underbrace{O \cdot e^{-\beta^D(\mathbf{v}_D)\cdot z}}_{\text{Direct Image component}} + \underbrace{B^\infty \cdot (1 - e^{-\beta^B(\mathbf{v}_B)\cdot z})}_{\text{Backscatter Image component}}, \quad (6)$$

where O is the clear scene captured at depth z in no medium, B^∞ is the backscatter color of the water at infinite distance. The colors get multiplied with attenuations, where the β^D and β^B are attenuation coefficients for the direct and backscatter components of the image which represent the effect the medium has on the color. The vector $\mathbf{v}^D$ represents the dependencies for the direct component, which includes factors such as the depth z, reflectance, ambient light, water scattering properties, and the attenuation coefficient of the water. The vector $\mathbf{v}^B$ represents the dependencies for the backscatter component, which includes ambient light, water scattering properties, the backscatter coefficient, and the attenuation coefficient of the water.

3.2. Splatting with Medium

We illustrate the pipeline of our method in Fig. 2. The input to our model is a set of images with scattering medium and corresponding camera poses. We initialize a set of 3D Gaussians via SfM [15] and optimize them with medium properties encoded by a neural network. Under the occlusion of both primitives and medium, our model acquires the transmittance along the ray and is capable of synthesizing medium component and object component in the novel view. Below we derive the whole model in detail.

Considering the expected color of a pixel integrated along the camera ray $r(s) = o + d(s)$ from the camera to infinitely far $C(r) = \int_0^\infty T(s)\sigma(s)c(s)ds$ [13] because of unbounded rendering of 3DGS [15], we release the constraints on light traveling in clear air to through a scattering medium [18] by adding the medium term:

$$C(r) = \int_0^\infty T(s)(\sigma^{\text{obj}}(s)c^{\text{obj}}(s) + \sigma^{\text{med}}(s)c^{\text{med}}(s))ds \quad (7)$$

$$T(s) = exp(-\int_0^\infty (\sigma^{\text{obj}}(s) + \sigma^{\text{med}}(s))ds), \quad (8)$$

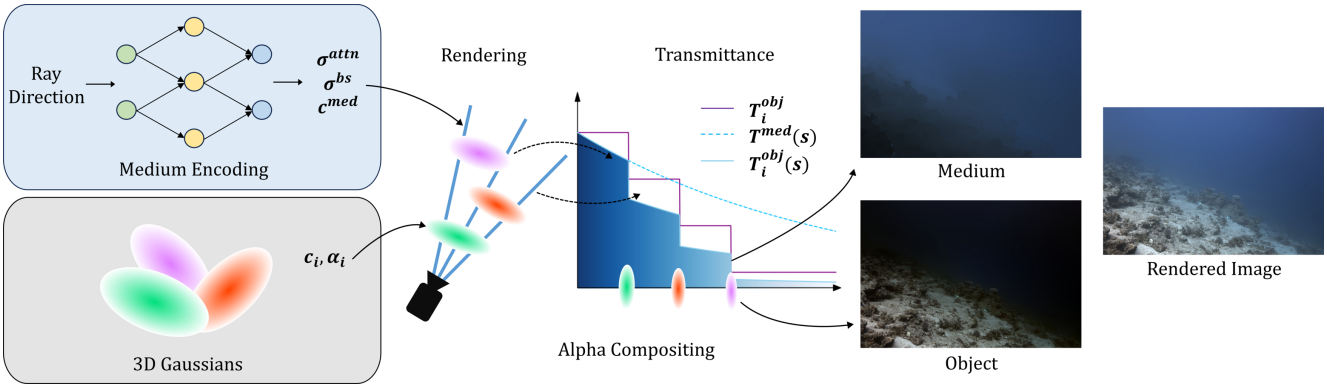

Figure 2. **Splatting with Medium**: We start rendering by casting a ray per pixel and collect the patch-intersected Gaussians along the ray and their color given ray direction. Then, we walk through the list of sorted Gaussians per pixel, query their opacity and depth, based on which we acquire the transmittance of both Gaussians and medium, rendering the Gaussians and the segments between adjacent two to obtain the **Medium** component and the **Object** component.

where $\sigma^{\text{obj}}/\sigma^{\text{med}}$ and $c^{\text{obj}}/c^{\text{med}}$ are density and color of objects and medium respectively.

Following [18], we take σ^{med} and c^{med} to be constant per ray and separate per color channel. In order to apply discretized representation in 3DGS, the transmittance $T_i(s)$ in front of the i-th Gaussian $\mathcal{G}_i$ (and behind (i-1)-th Gaussian $\mathcal{G}_{i-1}$) with depth $s \in [s_{i-1}, s_i]$ can be decomposed as

$$T_i(s) = T_i^{\text{obj}} T^{\text{med}}(s), \quad T_i^{\text{obj}} = \prod_{j=1}^{i-1}(1-\alpha_j) \tag{9}$$

where T_i^{obj} is the accumulated transmittance contributed by previous primitives' occlusion [15] and

$$T^{\text{med}}(s) = exp(-\int_0^s \sigma^{\text{med}}(s)ds) = exp(-\sigma^{\text{med}} s) \tag{10}$$

is the accumulated transmittance under the effect of medium from the camera to depth s. Then, the color is composed with discretized Gaussians and integrable medium

$$C(r) = \sum_{i=1}^{N} C_i^{\text{obj}}(r) + \sum_{i=1}^{N} C_i^{\text{med}}(r). \tag{11}$$

The contributed color of the $\mathcal{G}_i$ to final output is

$$\begin{aligned} C_i^{\text{obj}}(r) &= T_i^{\text{obj}} T^{\text{med}}(s_i)\alpha_i c_i \\ &= T_i^{\text{obj}} \alpha_i c_i exp(-\sigma^{\text{med}} s_i), \end{aligned} \tag{12}$$

where α_i is the opacity given the relative position between the pixel p and μ_i in Eq. (5) and $c_i = c_i^{\text{obj}}$ is the color given the ray direction. The contributed color of the medium between the (i-1)-th and $\mathcal{G}_i$ is

$$\begin{aligned} C_i^{\text{med}}(r) &= \int_{s_{i-1}}^{s_i} T_i^{\text{obj}} T^{\text{med}}(s)\sigma^{\text{med}} c^{\text{med}} ds \\ &= T_i^{\text{obj}} c^{\text{med}} [exp(-\sigma^{\text{med}} s_{i-1}) - exp(-\sigma^{\text{med}} s_i)]. \end{aligned} \tag{13}$$

To precisely estimate the properties of medium, we also include the background medium term from the last Gaussian $\mathcal{G}_N$ to infinitely far

$$\begin{aligned} C_\infty^{\text{med}}(r) &= \int_{s_N}^{\infty} T_i^{\text{obj}} T^{\text{med}}(s)\sigma^{\text{med}} c^{\text{med}} ds \\ &= T_i^{\text{obj}} c^{\text{med}} exp(-\sigma^{\text{med}} s_N) \end{aligned} \tag{14}$$

into the accumulated color.

As discussed in [1], the effective σ^{med} experienced a camera with wide-band color channels by differs in $C_\cdot^{\text{obj}}(r)$ and $C_\cdot^{\text{med}}(r)$, following [18], we use two sets of parameters, object attenuation σ^{attn} and medium back-scatter σ^{bs} for $C_i^{\text{obj}}(r)$ and $C_i^{\text{med}}(r)$ respectively. By setting $s_0 = 0$, our final equations of rendered color are:

$$C(r) = \sum_{i=1}^{N} C_i^{\text{obj}}(r) + \sum_{i=1}^{N} C_i^{\text{med}}(r) + C_\infty^{\text{med}}(r), \tag{15}$$

$$C_i^{\text{obj}}(r) = T_i^{\text{obj}} \alpha_i c_i exp(-\sigma^{attn} s_i), \tag{16}$$

$$C_i^{\text{med}}(r) = T_i^{\text{obj}} c^{\text{med}} [exp(-\sigma^{\text{bs}} s_{i-1})) - exp(-\sigma^{\text{bs}} s_i)], \tag{17}$$

$$C_\infty^{\text{med}}(r) = T_N^{\text{obj}} c^{\text{med}} exp(-\sigma^{\text{bs}} s_N). \tag{18}$$

3.3. Loss Function Alignment

In vanilla 3DGS, the loss function is combined with an $\mathcal{L}_1$ Loss and a D-SSIM Loss, which suits primitives without shared parameters. Inspired by [25], we propose a regularized loss function $\mathcal{L}_{\text{Reg}}$ to boost the weight of the dark regions in optimization to align with how humans perceive dynamic range. To be more specific, we apply a pixel-wise weight $W = \{w_{i,j}\}$ on both rendered estimate $\hat{y}$ and target

image y, where $w_{i,j} = (sg(\hat{y}_{i,j}) + \epsilon)^{-1}$ with pixel coordinate (i, j) and $sg(\cdot)$ denotes stopping gradient of its argument, which backpropagates zero derivative.

Thus, we have our regularized $\mathcal{L}_1$ Loss

$$\mathcal{L}_{\text{Reg-}\mathcal{L}_1} = |W \odot (\hat{y} - y)|, \tag{19}$$

which results in less blur and sharper edges, and our regularized D-SSIM Loss

$$\mathcal{L}_{\text{Reg-DSSIM}} = \mathcal{L}_{\text{DSSIM}}(W \odot y, W \odot \hat{y}), \tag{20}$$

which encourages high structural similarity between y and $\hat{y}$. Our final proposed loss function is

$$\mathcal{L}_{\text{Reg}} = (1 - \lambda)\mathcal{L}_{\text{Reg-}\mathcal{L}_1} + \lambda\mathcal{L}_{\text{Reg-DSSIM}}, \tag{21}$$

which encourages Gaussian optimization to better align the human eye's perception of the dynamic range.

4. Experiments

4.1. Experiments

Implementation Details: Our implementation is based on the reimplemented version of 3DGS released by NeRF-Studio. Following [42, 46], we accumulate the norms of the individual pixel gradients of μ_i for primitive densification. For the medium encoding, we use a spherical harmonic encoding [37] and a MLP with 2 linear layers with 128 hidden units and Sigmoid activation, followed by Sigmoid activation for c^{med} and Softplus activation for σ^{attn} and σ^{bs}. At each densification and pruning step of the 3DGS, the moving averages in the Adam optimizer of the medium encoding are reset.

SeaThru-NeRF Dataset: SeaThru-NeRF Dataset released by [18] contains real-world scenes acquired from four different scenes in sea: IUI3 Red Sea, Curaçao, Japanese Gardens Red Sea, and Panama. There are 29, 20, 20 and 18 images respectively, among which 25, 17, 17 and 15 images are used for training and the rest 4, 3, 3 and 3 are used for validation. The dataset encompasses a variety of water and imaging conditions. The images were captured in RAW format using a Nikon D850 SLR camera housed in a Nauticam underwater casing with a dome port, which helped prevent refractions that could disrupt the pinhole model. These images were then downsampled to an approximate resolution of 900 × 1400. Prior to processing, the input linear images underwent white balancing with a 0.5% clipping per channel to eliminate extreme noise pixels. Lastly, COLMAP [32] was employed to determine the camera poses.

Simulated Dataset: To further evaluate the performance of the proposed method, we take a standard NeRF dataset - the Garden scene from the Mip-NeRF 360 dataset [4] - and added fog to it to simulate the presence of medium. We used 3DGS to extract the depth maps. These maps were then utilized to create scenarios simulating both underwater and foggy conditions. In line with method Eq. (6), we utilized the following parameters to simulate easy foggy scenario: $\beta^D = [0.6, 0.6, 0.6]$, $\beta^B = [0.6, 0.6, 0.6]$, and $B^\infty = [0.5, 0.5, 0.5]$. The parameters for the hard foggy case are: $\beta^D = [0.8, 0.8, 0.8]$, $\beta^B = [0.6, 0.6, 0.6]$, and $B^\infty = [0.5, 0.5, 0.5]$.

Baseline methods: All methods were trianed on the same set of white-balanced images. For rendering scenes with the medium, we compare several NeRF techniques: SeaThru-NeRF [18], the reimplementation of SeaThru-NeRF released on NeRF-Studio (SeaThru-NeRF-NS) [36], Zip-NeRF [5] and 3DGS [15]. For each baseline method, we use the PSNR, SSIM, and LPIPS [47] metrics to compare rendering quality. We present the alpha blending of depth as the depth map and the rendering without medium to demonstrate the ability to decouple the medium and the object for SeaThru-NeRF and our method. We also calculate the FPS and total training time using the same RTX 4080 GPU to illustrate the speed difference between baselines and our method. All reported results are averaged over three runs.

4.2. Results

First, we evaluated the performance of our method using the standard benchmark dataset, the SeaThru-NeRF Dataset. Table 1 compares PSNR, SSIM, LPIPS, average FPS and average training time with the baseline methods across the validation sets of four scenes. Our method demonstrates its superiority in the majority of cases and efficiency on both rendering and training.

Fig. 3 shows that the mainstream 3DGS and NeRF approaches are short at reconstruction on back-scattering media. 3DGS prunes the Gaussians with low opacity, leaving dense and muddy cloud-like primitives to fit the medium, which causes artifacts in the novel views. Zip-NeRF struggles to model the geometrical surface, leading to an unreal scene reconstructed with little media left.

Fig. 4 demonstrates that in both the 'IUI3 Red Sea', the 'Japanese Gardens Red Sea' and 'Panama' scenes, our method delivers superior quality and more effectively separates medium and object than the SeaThru-NeRF, especially in deeper and more complex scenes (as highlighted in the red square). Additionally, our depth map reveals much finer details compared to SeaThru-NeRF, which struggles to produce a reasonable depth map at greater distances, as indicated by the red color in the upper right corner of the depth map. We also achieve higher PSNR values in both scenes. The same advantages are observed in simulated scenes, where our method renders better details (indicated by the red square) than the SeaThru-NeRF in both easy and hard foggy scenes as depicted in Fig. 5. Our rendering without medium and depth maps significantly outperform those from the SeaThru-NeRF, especially in scenes that are far-

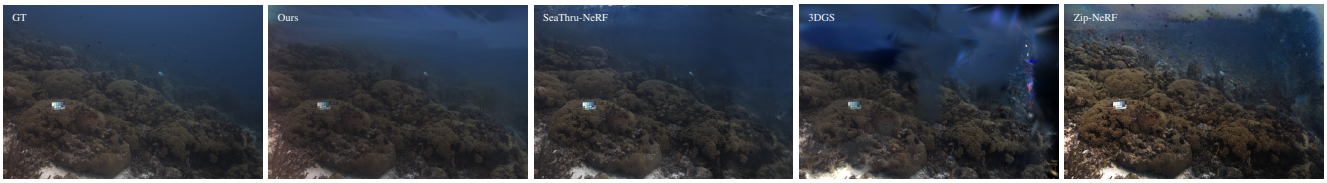

Figure 3. Underwater scene rendering in the 'Curasao' scene. From left to right: white-balanced ground-truth image, our result, SeaThru-NeRF's result, 3DGS' result, and Zip-NeRF's result. Both traditional 3DGS and NeRF with a proposal sampler cannot handle semi-transparent medium well.

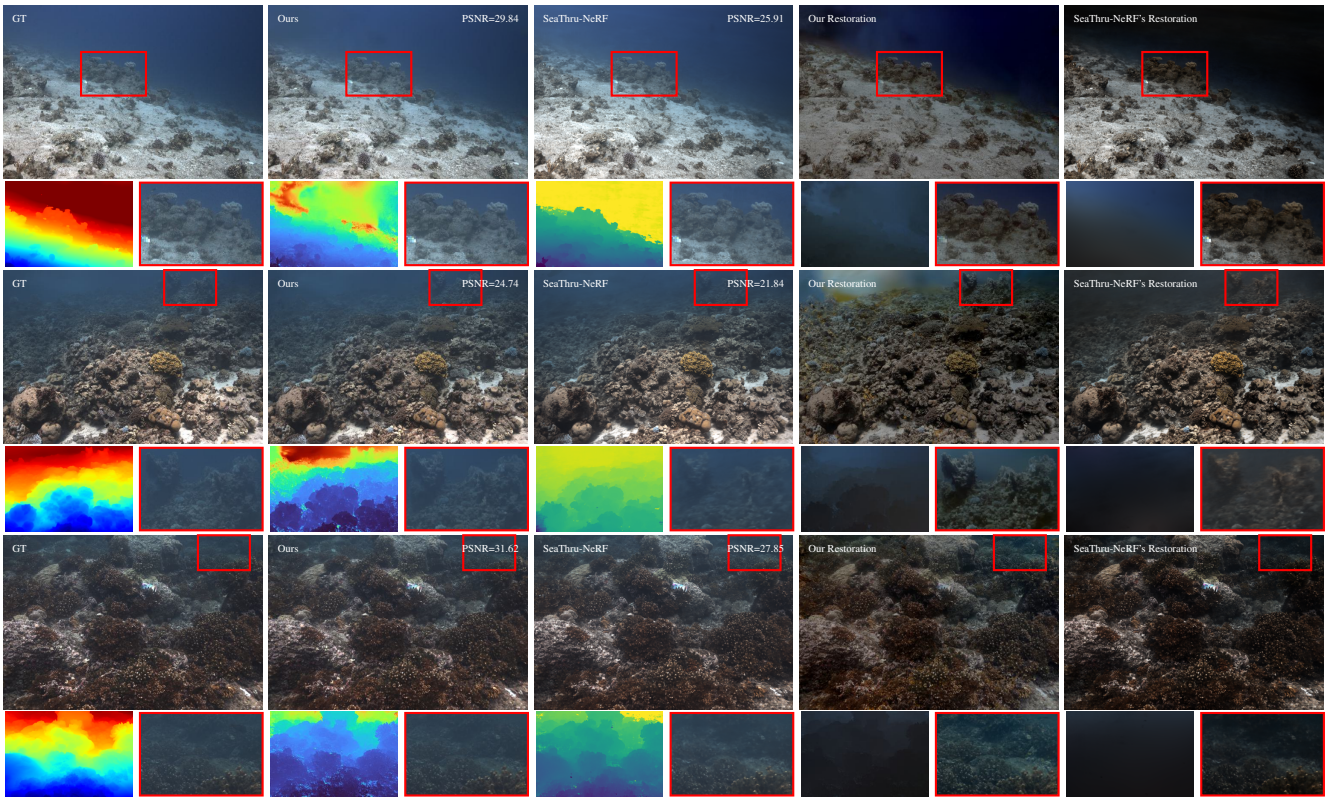

Figure 4. Underwater scene rendering in the 'IUI3 Red Sea' scene, 'Japanese Gardens Red Sea' scene and 'Panama' scene. We compare our method with SeaThru-NeRF by showing both the full image and the rendering without the medium. Furthermore, under each image, we show the depth maps (for GT the depth map from pre-trained model [41], and highlighted region from the image. For Restoration, we further show the rendered medium without rendering objects. Our method achieves better rendering quality and preserves finer distant geometric details while reducing the amount of floaters.

Table 1. **Quantitative evaluation on the SeaThru-NeRF dataset.** We show PSNR↑, SSIM↑, LPIPS↓, Avg. FPS↑, and Avg. Training Time↓

Dataset/Metric	**IUI3 Red Sea**			**Curaçao**			**J.G. Red Sea**			**Panama**			Avg.	Avg.
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	FPS	Time
SeaThru-NeRF	25.908	0.785	0.304	30.193	0.873	0.210	21.841	0.767	0.249	27.846	0.834	0.224	0.07	10h
SeaThru-NeRF-NS	26.755	0.826	0.168	30.959	0.915	0.133	23.282	0.876	0.111	31.276	0.937	0.071	0.9	2h
ZipNeRF	16.937	0.474	0.412	19.956	0.442	0.421	19.022	0.349	0.483	19.012	0.349	0.482	0.9	6h
3D Gauss.	22.980	0.843	0.2458	28.313	0.873	0.221	21.493	0.854	0.216	29.200	0.893	0.152	412.1	17.4min
Ours	29.840	0.889	0.203	32.203	0.948	0.116	24.741	0.892	0.116	31.616	0.942	0.080	41.8	9.4min

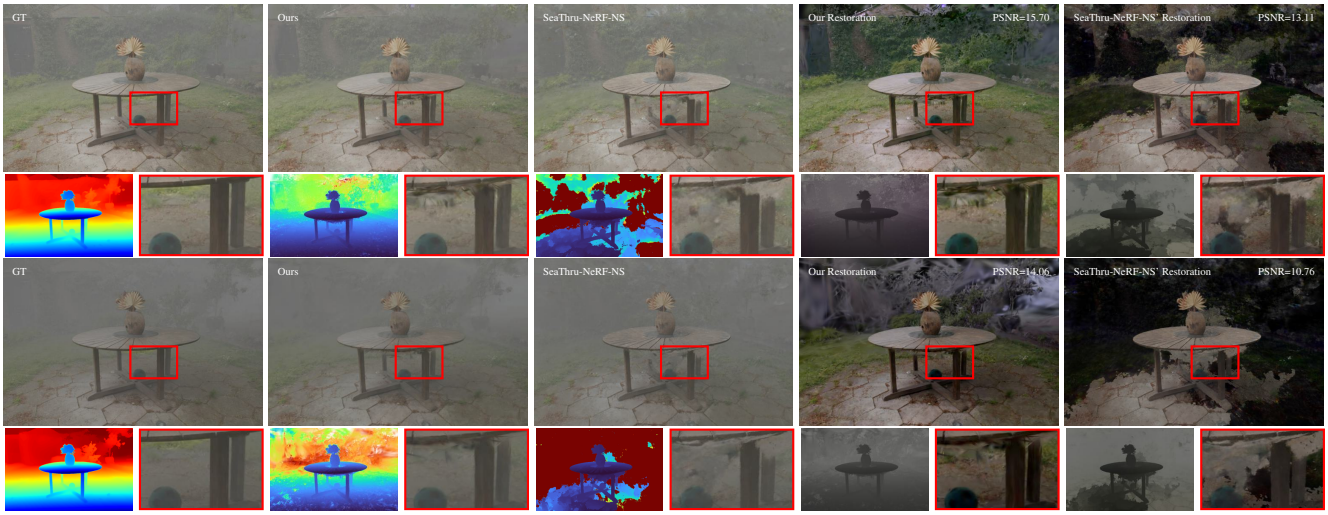

Figure 5. Simulated scene rendering with the easy foggy scene (upper) and hard foggy scene (lower). We compare our method with SeaThru-NeRF by showing both the full image and the rendering without the attenuation (restoration). Furthermore, under each image, we show the depth maps (for GT the depth map from pre-trained model [41]), and highlighted region from the image. For restoration, we further show the rendered medium without rendering objects. Our results exhibit better restoring quality and reasonable depth map compared to SeaThru-NeRF-NS' results.

ther from the camera. While our method's predictions may appear blurry and the object map unclear in the upper right corner for the hard foggy scene, the results from SeaThru-NeRF are considerably worse. The restoration quality comparison presented in Table 2 further quantitatively demonstrates the superiority of our method in simulated scenes. Overall, our method surpasses SeaThru-NeRF in both underwater and simulated scenes.

Table 2. **Restoration Performance.** (PSNR↑/SSIM↑/LPIPS↓)

Dataset/Metric	**Foggy-Easy**			**Foggy-Hard**		
Method	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
SeaThru-NeRF-NS	13.11	0.32	0.58	10.76	0.29	0.63
Ours	15.70	0.37	0.56	14.06	0.45	0.54

4.3. Ablation Study

We isolate the different contributions to address the key one leading to the increment. We conduct a quantitative analysis on different combination of loss functions, between pixel-wise component $\{\mathcal{L}_1, \mathcal{L}_{\text{Reg-}\mathcal{L}_2}, \mathcal{L}_{\text{Reg-}\mathcal{L}_1}\}$ and frame-wise $\{\mathcal{L}_{\text{DSSIM}}, \mathcal{L}_{\text{Reg-DSSIM}}\}$, as well as removing the medium effect and removing both the medium and our proposed loss function $\mathcal{L}_{\text{Reg}}$ as a reimplementation of 3DGS. These comparisons are made across validation sets for the SeaThru-NeRF dataset in Table 3.

Fig. 6 clearly shows that our frame-level $\mathcal{L}_{\text{Reg-DSSIM}}$ provides better details in low light. When used alone, $\mathcal{L}_{\text{Reg-DSSIM}}$ can also improve the reconstruction quality and structural similarity of distant objects. Our pixel-level $\mathcal{L}_{\text{Reg-}\mathcal{L}_1}$ can further sharpen the edges. The supervision ability of $\mathcal{L}_{\text{Reg-}\mathcal{L}_2}$ [25] for 3DGS-based models is relatively insufficient. Our proposed $\mathcal{L}_{\text{Reg}}$ shows superiority over other configurations in leading the 3DGS-based model to better fit HDR scenes and removing the medium component (basically 3DGS) significantly hurms the performance of our method, which indicates the necessity of our approach.

Table 3. **Ablation Study** Avg. over SeaThru-NeRF Scenes

Configuration	**PSNR↑**	**SSIM↑**	**LPIPS↓**
$\mathcal{L}_1$+$\mathcal{L}_{\text{DSSIM}}$	28.98	0.903	0.172
$\mathcal{L}_1$+$\mathcal{L}_{\text{Reg-DSSIM}}$	29.58	0.918	0.128
$\mathcal{L}_{\text{Reg-}\mathcal{L}_2}$+$\mathcal{L}_{\text{DSSIM}}$	28.18	0.900	0.180
$\mathcal{L}_{\text{Reg-}\mathcal{L}_2}$+$\mathcal{L}_{\text{Reg-DSSIM}}$	29.31	0.917	0.129
$\mathcal{L}_{\text{Reg-}\mathcal{L}_1}$+$\mathcal{L}_{\text{DSSIM}}$	28.91	0.906	0.160
$\mathcal{L}_{\text{Reg-}\mathcal{L}_1}$+$\mathcal{L}_{\text{Reg-DSSIM}}$	29.60	0.918	0.129
$\mathcal{L}_{\text{Reg-}\mathcal{L}_1}$+$\mathcal{L}_{\text{Reg-DSSIM}}$ w/o Med.	29.10	0.912	0.141
$\mathcal{L}_1$+$\mathcal{L}_{\text{DSSIM}}$ w/o Medium	28.96	0.903	0.175

5. Limitations

Although our method achieves good reconstruction quality, there are some limitations to consider. Firstly, our method, similar to NeRF-based approaches [18], has some difficulties with distinguishing the background-like object and the medium far in the distance, as illustrated on the top of Fig. 3 and Fig. 7. However, in the foreground, our method prunes medium-role primitives well while SeaThru-NeRF cannot prevent the geometrical field from fitting the medium, resulting in wave-like artifacts. Secondly, same as other NVS methods [15, 24], our method relies on the camera poses

Figure 6. **Ablation Study: loss function alignment.** Our proposed $\mathcal{L}_{\text{Reg-DSSIM}}$ improves the reconstruction quality of distant details in dark areas, and the benefit is obvious even when used alone. Our proposed $\mathcal{L}_{\text{Reg-}\mathcal{L}_1}$ further improves the fineness of the reconstruction.

Figure 7. **Limitation: distant medium represented by Gaussians.** Our method (**left**) models distant medium with Gaussians; SeaThru-NeRF [18] (**right**) also struggles with the background.

Figure 8. **Limitation: insufficient supervision.** Our method (**left**) has low details visuals in regions not sufficiently covered by training views. SeaThru-NeRF [18] (**right**) suffers from blurring.

being available which might prove difficult to obtain in underwater 3D scenes. Thirdly, our 3DGS-based method has artifacts in regions lacking observation [15], which is also suffered by NeRF-based models, as illustrated in the left side of Fig. 8 and the top part of Fig. 4, while the NeRF-based SeaThru-NeRF approach (right image) will introduces some blurring, distortion and interpolation. Lastly, the restored color of the scene from the scene is not ensured to be precise (especially for the background-like object), as under the effect of medium, the color of object and the attenuation attribute are entangled during training, which is shown by Fig. 5.

6. Conclusions

In our work, we focused on the problem of underwater reconstruction, previously tackled by fully volumetric representations that are slow to train and render. Therefore, we proposed to fuse the explicit point-splatting method (3DGS) with volume rendering to achieve both fast training and real-time rendering speed. Our method interleaves alpha compositing of splatted Gaussians with integrated ray segments passing through the scattering medium. We have demonstrated that our method achieves state-of-the-art results while enabling real-time rendering. Furthermore, the explicit scene representation enables disentanglement of geometry and the scattering medium. In future work, we would like to extend our method for larger scenes with both water and fog.

Acknowledgements. This work was supported by the Czech Science Foundation (GAČR) EXPRO (grant no. 23-07973X), and by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254). Jonas Kulhanek acknowledges travel support from the European Union's Horizon 2020 research and innovation programme under ELISE (grant no. 951847).

References

[1] Derya Akkaynak and Tali Treibitz. A revised underwater image formation model. In *CVPR*, 2018. 3, 4

[2] Derya Akkaynak and Tali Treibitz. Sea-thru: A method for removing water from underwater images. In *CVPR*, 2019. 2

[3] 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, 2021. 2

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 2, 5

[5] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. *ICCV*, 2023. 2, 5

[6] Jiazhong Cen, Zanwei Zhou, Jiemin Fang, Chen Yang, Wei Shen, Lingxi Xie, Dongsheng Jiang, Xiaopeng Zhang, and Qi Tian. Segment anything in 3d with nerfs. In *NeurIPS*, 2023. 2

[7] Eric R. Chan, Koki Nagano, Matthew A. Chan, Alexander W. Bergman, Jeong Joon Park, Axel Levy, Miika Aittala, Shalini De Mello, Tero Karras, and Gordon Wetzstein. GeNVS: Generative novel view synthesis with 3D-aware diffusion models. In *arXiv*, 2023. 2

[8] 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. *arXiv preprint arXiv:2103.15595*, 2021. 2

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

[10] Jaeyoung Chung, Suyoung Lee, Hyeongjin Nam, Jaerin Lee, and Kyoung Mu Lee. Luciddreamer: Domain-free generation of 3d gaussian splatting scenes, 2023. 2

[11] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 2

[12] Salma P. González-Sabbagh and Antonio Robles-Kelly. A survey on underwater computer vision. *ACM Comput. Surv.*, 55(13s), 2023. 2

[13] James T. Kajiya and Brian P Von Herzen. Ray tracing volume densities. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, page 165–174, New York, NY, USA, 1984. Association for Computing Machinery. 3

[14] Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathan Luiten. Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. *arXiv*, 2023. 2

[15] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM TOG*, 2023. 1, 2, 3, 4, 5, 7, 8

[16] Chung Min* Kim, Mingxuan* Wu, Justin* Kerr, Matthew Tancik, Ken Goldberg, and Angjoo Kanazawa. Garfield: Group anything with radiance fields. In *arXiv*, 2024. 2

[17] Jonas Kulhanek and Torsten Sattler. Tetra-NeRF: Representing neural radiance fields using tetrahedra. *arXiv preprint arXiv:2304.09987*, 2023. 2

[18] Deborah Levy, Amit Peleg, Naama Pearl, Dan Rosenbaum, Derya Akkaynak, Simon Korman, and Tali Treibitz. Seathru-nerf: Neural radiance fields in scattering media, 2023. 1, 2, 3, 4, 5, 7, 8

[19] Mingrui Li, Shuhong Liu, Heng Zhou, Guohao Zhu, Na Cheng, Tianchen Deng, and Hongyu Wang. Sgs-slam: Semantic gaussian splatting for neural dense slam, 2024. 2

[20] Chen-Hsuan Lin, Chaoyang Wang, and Simon Lucey. Sdf-srn: Learning signed distance 3d object reconstruction from static images. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[21] Jiaqi Lin, Zhihao Li, Xiao Tang, Jianzhuang Liu, Shiyong Liu, Jiayue Liu, Yangdi Lu, Xiaofei Wu, Songcen Xu, Youliang Yan, and Wenming Yang. Vastgaussian: Vast 3d gaussians for large scene reconstruction. In *CVPR*, 2024. 2

[22] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object, 2023. 2

[23] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4):65:1–65:14, 2019. 2

[24] 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, 2020. 1, 2, 7

[25] Ben Mildenhall, Peter Hedman, Ricardo Martin-Brualla, Pratul Srinivasan, and Jonathan T. Barron. Nerf in the dark: High dynamic range view synthesis from noisy raw images, 2021. 4, 7

[26] 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, 2022. 2

[27] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 2

[28] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. Langsplat: 3d language gaussian splatting. *arXiv preprint arXiv:2312.16084*, 2023. 2

[29] Chen Quei-An. $nerf_pl$: a pytorch-lightning implementation of nerf, 2020. 2

[30] Andrea Ramazzina, Mario Bijelic, Stefanie Walz, Alessandro Sanvito, Dominik Scheuble, and Felix Heide. Scatternerf: Seeing through fog with physically-based inverse neural rendering, 2023. 2

[31] Sara Fridovich-Keil and Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, 2023. 2

[32] Johannes L Schonberger and Jan-Michael Frahm. Structure from-motion revisited. *CVPR*, 2016. 5

[33] Advaith Venkatramanan Sethuraman, Manikandasriram Srinivasan Ramanagopal, and Katherine A. Skinner.

Waternerf: Neural radiance fields for underwater scenes, 2023. 2

[34] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 2

[35] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P. Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8248–8258, 2022. 2

[36] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 2, 5

[37] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *CVPR*, 2022. 5

[38] Peng Wang, Yuan Liu, Zhaoxi Chen, Lingjie Liu, Ziwei Liu, Taku Komura, Christian Theobalt, and Wenping Wang. F2-nerf: Fast neural radiance field training with free camera trajectories. *CVPR*, 2023. 2

[39] Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong Park, Ruiqi Gao, Daniel Watson, Pratul P. Srinivasan, Dor Verbin, Jonathan T. Barron, Ben Poole, and Aleksander Holynski. Reconfusion: 3d reconstruction with diffusion priors. *arXiv*, 2023. 2

[40] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 2

[41] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv:2406.09414*, 2024. 6, 7

[42] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. Absgs: Recovering fine details for 3d gaussian splatting, 2024. 2, 5

[43] 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. 2

[44] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 2

[45] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. *arXiv:2311.16493*, 2023. 2

[46] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient and compact surface reconstruction in unbounded scenes, 2024. 5

[47] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, Los Alamitos, CA, USA, 2018. IEEE Computer Society. 5