

RefracGS: Novel View Synthesis Through Refractive Water Surfaces with 3D Gaussian Ray Tracing

Yiming Shao^{1*}, Qiyu Dai^{2*}, Chong Gao³, Guanbin Li^{3,5}, Yejiang Wang^{2,4},
He Sun², Qiong Zeng^{1†}, Baoquan Chen², and Wenzheng Chen^{2,4†}

¹ Shandong University, China

² Peking University, Beijing, China

³ Sun Yat-sen University, China

⁴ Beijing Academy of Artificial Intelligence, Beijing, China

⁵ Shenzhen Loop Area Institute, China

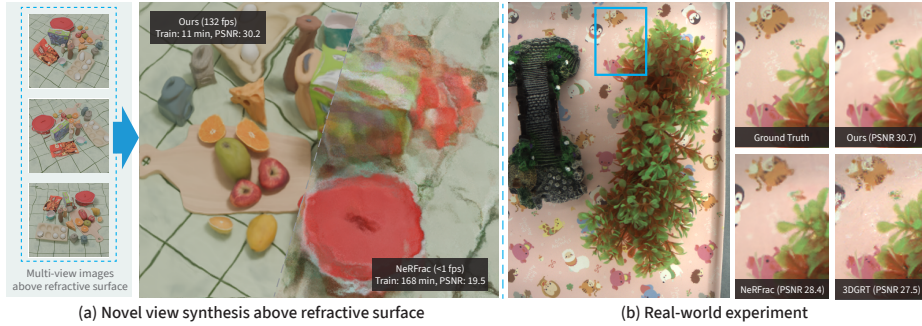


Fig. 1: (a): Our method reconstructs both the refractive water surface and the underlying scene from multi-view images captured above, achieving high-fidelity novel view synthesis with fast training and real-time rendering. (b): Experiments in real-world scenarios further demonstrate the effectiveness of our approach.

Abstract. Novel view synthesis (NVS) through non-planar refractive surfaces presents fundamental challenges due to severe, spatially varying optical distortions. While recent representations like NeRF and 3D Gaussian Splatting (3DGS) excel at NVS, their assumption of straight-line ray propagation fails under these conditions, leading to significant artifacts. To overcome this limitation, we introduce RefracGS, a framework that jointly reconstructs the refractive water surface and the scene beneath the interface. Our key insight is to explicitly decouple the refractive boundary from the target objects: the refractive surface is modeled via a neural height field, capturing wave geometry, while the underlying scene is represented as a 3D Gaussian field. We formulate a refraction-aware Gaussian ray tracing approach that accurately computes non-linear ray trajectories using Snell’s law and efficiently renders the underlying Gaussian field while backpropagating the loss gradients to the parameterized

* Equal contribution.

† Corresponding authors.

refractive surface. Through end-to-end joint optimization of both representations, our method ensures high-fidelity NVS and view-consistent surface recovery. Experiments on both synthetic and real-world scenes with complex waves demonstrate that RefracGS outperforms prior refractive methods in visual quality, while achieving $\sim 15\times$ **faster** training and real-time rendering at **200 FPS**.

Keywords: Novel view synthesis · Refraction · Ray tracing

1 Introduction

Novel view synthesis (NVS) of scenes observed through non-planar refractive surfaces (e.g., water surfaces) presents a long-standing and fundamentally challenging problem in computer vision. Such capabilities hold significant promise for a wide range of real-world applications, including shallow water bathymetry, environmental monitoring, and underwater cultural heritage preservation. In contrast to NVS in air, NVS through refractive surfaces tackles complex distortions and disrupted correspondences caused by the refractive effects of the interface. As light rays traverse interfaces between media with mismatched refractive indices, they undergo abrupt changes in direction, leading to severe optical distortions, particularly in the presence of surface undulations. This refraction-induced bending fundamentally invalidates the multi-view consistency and linear projection assumptions central to standard reconstruction, and effective solutions to this ill-posed problem remain largely absent in existing work.

Recent advances in Neural Radiance Fields (NeRF) [17] and 3D Gaussian Splatting (3DGS) [13] have enabled high-fidelity NVS for in-air scenes. However, these methods predominantly assume linear radiance transport within a homogeneous medium and thus fail under refractive conditions. To address non-linear light paths introduced by refraction, prior works have attempted explicit modeling. NeRF-based methods, such as NeRFrac [35], incorporate ray-bending to simulate refractive effects, but suffer from the high computational cost of volumetric rendering, limiting both training and inference speed. More crucially, many such approaches define the refractive interface in view-dependent spaces—e.g., parameterizing surface normals as functions of camera rays—which introduces geometric inconsistencies across views and hampers robust surface reconstruction [28, 35]. Parallel efforts have started extending 3DGS to support non-linear light transport, as seen in TransparentGS [11], which focuses on the reconstruction of transparent solid objects. Yet, these methods are not designed for through-surface imaging, where refractive distortion arises from a dynamic fluid interface.

In this work, we propose RefracGS, a novel framework that simultaneously reconstructs both the refractive water surface geometry and the underlying scene by coupling a novel refractive surface representation with 3D Gaussian ray tracing. Specifically, we introduce Water Height Map, a hybrid refractive surface representation that combines a contiguous neural height field with a recursively subdivided triangular mesh. This design provides mesh-like efficiency for ray-surface

intersection queries and field-like continuity for accurate, view-consistent surface geometry.

Built upon Water Height Map, we develop a refraction-aware Gaussian ray tracing algorithm that explicitly enforces Snell’s law at the air–water interface. For each camera pixel, we first trace a ray to intersect the reconstructed refractive surface using a hierarchical subdivision scheme, then bend the ray according to the estimated local surface normal, and finally accumulate color and opacity along the refracted path through a set of 3D Gaussian primitives modeling the underlying scene. Different from 3DGRT [18], our entire pipeline is fully differentiable: gradients flow not only to the Gaussian parameters (positions, opacities, and appearance coefficients) but also to the parameterized refractive surface, enabling joint optimization from image supervision alone. To the best of our knowledge, this is the first method to directly optimize refractive surfaces using Gaussian ray tracing. Implemented with efficient CUDA kernels and hardware-accelerated ray traversal, our method preserves much of the speed advantage of Gaussian Splatting while faithfully modeling physical refraction.

We conduct extensive experiments on both synthetic and real-world scenes with complex wave dynamics. The results show that RefracGS consistently outperforms all baselines, achieving state-of-the-art performance in both visual quality and surface geometry accuracy. In addition, our method is approximately $\sim 15\times$ faster to train while supporting real-time rendering at over **200 FPS**. Finally, the explicit disentanglement of the refractive surface extends beyond novel view synthesis, enabling flexible and controllable scene manipulation, such as water removal and surface editing, without retraining.

2 Related Work

In this section, we briefly review two related topics: recent advances in novel view synthesis, and methods for reconstructing refractive surfaces.

Novel View Synthesis Novel view synthesis (NVS) has witnessed remarkable advancements in recent years. NeRF [17] models scenes with a multilayer perceptron (MLP) and synthesizes new views via volumetric rendering. Numerous subsequent works have aimed to enhance the computational efficiency of NeRF [4, 8, 19], as well as the capability of handling aliasing [1], unbounded scene [2], and other challenges.

Instead of using neural network to represent scenes, 3D Gaussian Splatting [13] uses 3D Gaussian particles, which can be rendered through highly efficient rasterization, achieving competitive results in terms of both quality and efficiency. Several follow-up studies have proposed improvements to 3D Gaussian Splatting [7, 14, 16, 32]. In addition, some works have focused on extracting high-quality geometry from Gaussian representations [5, 10, 34]. Although the rasterization-based rendering in 3D Gaussian Splatting enables extremely fast speed, it inherently limits the method’s ability to handle complex light transport phenomena such as reflection and refraction. Some methods [12, 31] represent reflective objects through material decomposition and related techniques, but they

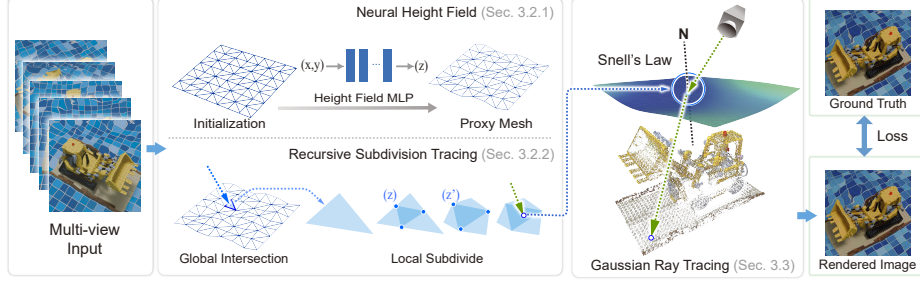


Fig. 2: Overview of the RefracGS pipeline. A proxy refractive-surface mesh is first extracted from the neural height field, after which camera rays intersect the mesh using a hierarchical global-to-local strategy to obtain intersection points and normals. The refracted directions are then computed with Snell’s law, and the resulting rays traverse the underlying scene before being rendered with Gaussian ray tracing. still lack explicit tracing of light paths and are not applicable to the reconstruction of refracted objects. 3DGRT [18] replaces rasterization with ray tracing for rendering, allowing the simulation of complex light transport phenomena. While this approach inspires our work, the direct reconstruction of refractive objects using Gaussian ray tracing remains largely unexplored.

Refractive Surface Reconstruction Reconstructing refractive surfaces and the objects behind them has long been a challenging task. Several approaches [3, 6, 15, 20] have augmented NeRF to support the modeling of refractive objects. Other methods [9, 23–25] adopt Signed Distance Function (SDF) to model refractive surfaces. Since NeRF and SDF based methods adopt ray-based rendering, they are naturally suited to handling the light path bending caused by refraction. However, the rasterization-based rendering in 3D Gaussian Splatting makes it difficult to handle refraction phenomena. TransparentGS [11] bakes refracted background content using light field probes for transparent object reconstruction. But this method approximates the Gaussian radiance field via light field probes, inherently introducing deviations relative to accurate ray tracing. Crucially, these approaches are primarily designed for reconstructing isolated transparent objects, rendering them ill-equipped for through-surface imaging where refractions arise from a continuous, non-planar interface.

In addition, several studies aim to reconstruct both the refractive water surface and the underwater scene from images captured above water. Qian *et al.* [22] reconstruct both the water surface and the underwater scene by leveraging multi-view consistency and variational optical flow estimation. Xiong *et al.* [30] reconstruct both the water surface and the underwater scene from a single-camera video sequence. NeReF [28] reconstructs fluid surfaces from multi-view images, but it assumes access to a known underwater pattern, limiting its applicability in real-world settings. More recently, WSGS [33] addresses refractive underwater reconstruction; however, it relies on an idealized assumption of a perfectly planar water surface. NeRFrac [35] simultaneously reconstructs non-planar refractive surfaces and underlying scenes from multi-view images, providing inspiration for our work.

3 Methodology

Given a series of images captured from multiple viewpoints above a refractive surface, along with their corresponding camera parameters, our method aims to reconstruct both the Refractive surface geometry and the underlying scene. The overall pipeline of our proposed RefracGS framework is illustrated in Fig. 2. The core of our approach lies in two key components: (1) a Water Height Map for accurate and differentiable modeling of the refractive surface, and (2) a tailored refraction-aware ray tracing mechanism compatible with 3D Gaussian Splatting [13] (3DGS). In Sec. 3.1, we first review 3D Gaussian Splatting and the physical law of refraction, followed by a comprehensive explanation on our Water Height Map (Sec. 3.2) and the refraction-aware Gaussian ray tracing algorithm (Sec. 3.3). Implementation details including loss function and optimization hyper-parameters are detailed in Sec. 3.4 to further enhance the reproducibility.

3.1 Preliminaries

3D Gaussian Splatting 3D Gaussian Splatting [13] represents a scene as a collection of anisotropic 3D Gaussian primitives $\mathcal{G} = \{\mathcal{G}_i\}_{i=1}^N$, where each Gaussian $\mathcal{G}_i$ is defined by its central position $\boldsymbol{\mu}_i \in \mathbb{R}^3$, covariance matrix $\boldsymbol{\Sigma}_i \in \mathbb{R}^{3 \times 3}$, opacity $o_i \in [0, 1]$, color $\mathbf{c}_i \in [0, 1]^3$, and optionally spherical harmonic coefficients. Each Gaussian primitive models a volumetric density distribution:

$$\alpha_i(\mathbf{x}) = o_i \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)\right), \quad (1)$$

where $\boldsymbol{\Sigma}_i = \mathbf{R}_i \mathbf{S}_i \mathbf{S}_i^\top \mathbf{R}_i^\top$ is decomposed into a rotation matrix $\mathbf{R}_i \in \mathbb{R}^{3 \times 3}$ and a scale matrix $\mathbf{S}_i \in \mathbb{R}^{3 \times 3}$ to ensure the positive semi-definite nature of the covariance matrix. During rendering, an efficient rasterization is conducted to project each Gaussian onto the image plane, resulting in a 2D elliptical footprint. The final color $\mathbf{C}$ of pixel $\mathbf{p}$ is computed by alpha compositing these projected Gaussians in front-to-back order:

$$\mathbf{C} = \sum_{i=1}^N T_i(\mathbf{p}) \alpha_i(\mathbf{p}) \mathbf{c}_i, \quad (2)$$

where $T_i(\mathbf{p}) = \prod_{j < i} (1 - \alpha_j(\mathbf{p}))$ denotes transmittance.

3D Gaussian Ray Tracing Despite the remarkable efficiency of the rasterization process, it inherently assumes linear radiance propagation, an assumption that often fails in complex real-world scenarios involving reflective surfaces or refractive interfaces. To address this limitation, 3DGRT [18] introduces a ray tracing-based rendering technique specifically designed for 3DGS, enabling the modeling of directional deviations caused by reflection and refraction while maintaining real-time performance. However, the proposed ray tracing formulation is not differentiable with respect to the reflective or refractive properties of surfaces, and thus remains incapable of reconstructing objects with unknown reflection or refraction characteristics.

The Law of Refraction In nature, the refraction of light is governed by Snell’s law, which describes how a ray of light changes direction when crossing media boundaries:

$$n_1 \sin \theta_1 = n_2 \sin \theta_2, \quad (3)$$

where n_1 and n_2 are the refractive indices of the two media, and θ_1 and θ_2 denote the angles of incidence and refraction, respectively. For computational convenience, we adopt the vector formulation of Snell’s law:

$$\omega_o = \eta(\omega_i + c_1 \mathbf{n}) - c_2 \mathbf{n}, \quad (4)$$

where $\omega_i \in \mathbf{SO}(3)$ is the incident ray direction, $\mathbf{n} \in \mathbf{SO}(3)$ is the surface normal, and $\omega_o \in \mathbf{SO}(3)$ is the refracted ray direction. Here, $\eta = n_1/n_2$ is the ratio of refractive indices, $c_1 = \mathbf{n} \cdot \omega_i$, and $c_2 = \sqrt{1 - \eta^2(1 - c_1^2)}$.

3.2 Water Height Map

Overview A precise and differentiable representation of the Refractive surface is crucial for accurately simulating refraction. For any given ray defined by an origin $\mathbf{o} \in \mathbb{R}^3$ and direction $\mathbf{d} \in \mathbf{SO}(3)$, our goal is to determine two key quantities: the intersection point $\mathbf{p} \in \mathbb{R}^3$ on the surface where refraction occurs, and the resulting refracted ray direction $\omega_o \in \mathbf{SO}(3)$ as it propagates into the underlying scene. Therefore, an effective refractive surface model $\mathcal{W}$ must provide the intersection depth $t \in \mathbb{R}^+$ for intersection position computation $\mathbf{p} = \mathbf{o} + t\mathbf{d}$ and the surface normal $\mathbf{n} \in \mathbf{SO}(3)$ at $\mathbf{p}$ for refracted direction computation, as formulated in Eq. 5:

$$(t, \mathbf{n}) = \mathcal{W}(\mathbf{o}, \mathbf{d}). \quad (5)$$

Counterintuitively, modeling a refractive surface both accurately and efficiently remains a challenge.

Existing representations for refractive surfaces, such as explicit meshes, struggle to model high-frequency geometry like water waves due to limited resolution. Alternatively, implicit field methods like NeRFrac [35] leverage MLPs to model contiguous surfaces but suffer from view-dependent queries that lack multi-view consistency (Fig. 3) and require the computationally expensive plane fitting to estimate normal directions from the implicitly represented surfaces. To address these limitations, we propose Water Height Map, a novel representation that models the interface as a neural height field. By querying intersections via recursive subdivision tracing, our method combines the efficient intersection queries of meshes with the contiguous geometry modeling of fields, inherently achieving both spatial consistency and computational efficiency.

Neural Height Field Following NeRFrac [35], we define the Z -axis as the height direction, pointing upward from the underlying medium. This allows the refractive surface—approximately perpendicular to the Z -axis and aligned with the XY -plane—can be represented as a height field $\mathcal{H}(x, y)$:

$$\mathcal{H}(x, y) = z, \quad (6)$$

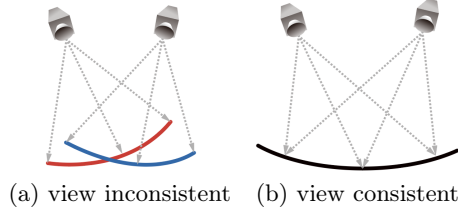


Fig. 3: Surface modeling strategy. NeRFRac (a) predicts ray–surface intersections using an MLP conditioned on ray origin and direction, which fails to enforce multi-view geometric consistency. In contrast, our method (b) explicitly models the refractive surface in world space, ensuring consistent intersection positions across viewpoints.

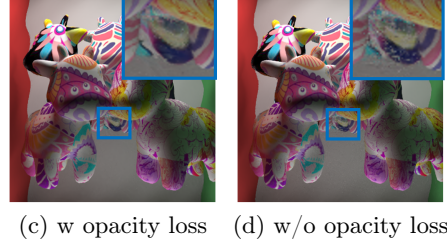


Fig. 4: Effectiveness of opacity loss. We compare rendering results (c) with and (d) without our proposed opacity loss. The visualization demonstrates that applying this regularization effectively eliminates floating artifacts in complexly occluded regions, yielding significantly cleaner scene reconstructions.

where $\mathcal{H}(x, y) - z = 0$ defines an implicit function that formulates the refractive surface as its zero isosurface. The height function $\mathcal{H}(x, y)$ is parameterized by a multi-layer perceptron (MLP), providing a continuous, smooth, and view-independent representation of the surface geometry. Compared to NeRFRac’s view-dependent refractive field, this formulation improves spatial consistency, facilitating stable multi-view optimization.

To model refraction, the height field should derive the intersection depth t and surface normal $\mathbf{n}$ for each ray query, as formulated in Eq. 5. However, directly computing intersections with the implicit MLP-defined isosurface is computationally expensive. To enable efficient geometric queries, we propose the Recursive Subdivision Tracing tailored for our Neural Height Field, as detailed in the following section.

Recursive Subdivision Tracing Given the ray query $\mathbf{r} = \mathbf{o} + t\mathbf{d}$ on a neural height field $\mathcal{H}$, our goal is to efficiently solve t , so that

$$\mathcal{H}(o_x + td_x, o_y + td_y) = o_z + td_z. \quad (7)$$

However, as discussed in Sec. 3.2, the implicit function nature of $\mathcal{H}$ hinders a direct solution of Eq. 7, and the conventional ray marching-based binary search is computationally impractical, especially when conducted during each rendering process.

Proxy-based Tracing To this end, we employ a proxy mesh surface as a piecewise linear approximation of $\mathcal{H}$ for accelerated ray-surface intersection. As illustrated in Fig. 2 (1), we predefine a 2D triangular mesh $\mathcal{M}_{2D}(\mathcal{V}_{2D}, \mathcal{F})$ with $\mathcal{V}_{2D} = \{(x_i, y_i) \in \mathbb{R}^2\}_{i=1}^M$ denoting its vertex set and $\mathcal{F}$ representing the triangular connectivity. For each 2D vertex (x_i, y_i) , we lift it to 3D by querying its

height $z_i = \mathcal{H}(x_i, y_i)$. This results in a 3D proxy mesh serving as the piecewise linear approximation of $\mathcal{H}$, thereby transforming the ray–height field intersection problem into a more tractable ray–mesh intersection problem, where the computation naturally benefits from modern hardware (*e.g.*, RT Cores) and can be accelerated by Bounding Volume Hierarchies (BVHs).

Recursive Subdivision Despite the efficiency, the limited resolution of the mesh proxy hinders accurate refraction modeling when confronted with complex interface involving detailed surface geometry. To this end, we propose a coarse-to-fine strategy for progressive approximation of $\mathcal{H}$. As outlined in Algorithm 1, we introduce the Recursive Subdivision Tracing, where $\mathcal{M}_{2D}$ will be recursively subdivided and the newly added vertices will updated their heights via $\mathcal{H}$ queries. Furthermore, this tracing approach can fully leverage locality by gradually pruning non-intersected triangles, eliminating the need for repeated BVH construction and bypassing redundant MLP evaluations. As a result, our proposed Recursive Subdivision Tracing achieves excellent efficiency while benefiting from great surface modeling ability of the neural height field.

Normal Estimation In this section, we detail the calculation of the normal direction in Algorithm 1. Given the ray–mesh intersection point $\mathbf{p}$ and the hit triangle facet $\mathcal{F} = (i, j, k)$, a straightforward strategy is to use the facet normal $\mathbf{n}_{\mathcal{F}} = \frac{(\mathbf{v}_j - \mathbf{v}_i) \times (\mathbf{v}_k - \mathbf{v}_i)}{\|(\mathbf{v}_j - \mathbf{v}_i) \times (\mathbf{v}_k - \mathbf{v}_i)\|_2}$ for refraction computation in Eq. 4. However, these face normals create discontinuity at triangle boundaries, adversely degrading rendering quality.

Inspired by Phong shading [21], for spatially consistent normal directions, we employ barycentric interpolation and interpolate the vertex normals via barycentric coordinates ($\alpha + \beta + \gamma = 1$, $\alpha\mathbf{v}_i + \beta\mathbf{v}_j + \gamma\mathbf{v}_k = \mathbf{p}$):

$$\mathbf{n}_p = \frac{\alpha\mathbf{n}_{v_i} + \beta\mathbf{n}_{v_j} + \gamma\mathbf{n}_{v_k}}{\|\alpha\mathbf{n}_{v_i} + \beta\mathbf{n}_{v_j} + \gamma\mathbf{n}_{v_k}\|_2}, \quad (8)$$

where the vertex normals are computed by averaging the adjacent face normals. This strategy ensures a smooth normal field across the surface, resulting in improved view synthesis quality, as evidenced by the ablation studies (Tab. 2).

3.3 Refraction-Aware Gaussian Ray Tracing

Our ray tracing process integrates the hierarchical surface model with the 3DGS framework. Similar to other ray-based methods, we begin by casting a ray from each pixel.

Forward Each ray first intersects the refractive surface, yielding an intersection point $\mathbf{p}$ and normal $\mathbf{n}$ from the method described in Sec. 3.2. Using these, the refracted ray direction $\omega_{\mathbf{o}}$ is computed via Snell’s law (Eq. 4). This new ray, originating from $\mathbf{p}$ with direction $\omega_{\mathbf{o}}$, then traverses the underlying scene. It

Algorithm 1 Recursive Subdivision Tracing

Input: An initial mesh proxy $\mathcal{M}_0 = (\mathcal{V}_0, \mathcal{F}_0)$, the ray queries $\mathbf{r}$, the height field $\mathcal{H}$.
Output: The ray depth t with intersection normal $\mathbf{n}$.

- 1: // Start with a coarse mesh for lightweight computing.
- 2: $\mathcal{V}_0 \leftarrow \text{QueryHeightField}(\mathcal{H}, \mathcal{V}_0)$
- 3: $\mathcal{B} \leftarrow \text{BuildBVH}(\mathcal{V}_0, \mathcal{F}_0)$
- 4: // Identify the intersected triangle and produce a mask.
- 5: $M_{\text{hit}} \leftarrow \text{GlobalIntersectionCompute}(\mathcal{B}, \mathbf{r})$
- 6: **for** $i = 1$ to K **do**
- 7: // Remove unhit vertices to reduce MLP overhead.
- 8: $(\mathcal{V}'_{i-1}, \mathcal{F}'_{i-1}) \leftarrow \text{RemoveUnhit}(\mathcal{V}_{i-1}, \mathcal{F}_{i-1}, M_{\text{hit}})$
- 9: // Each triangle is subdivided into four.
- 10: $(\mathcal{V}_i, \mathcal{F}_i) \leftarrow \text{SubdivideTriangles}(\mathcal{V}'_{i-1}, \mathcal{F}'_{i-1})$
- 11: // Refine the positions of newly added vertices.
- 12: $\mathcal{V}_i \leftarrow \text{QueryHeightField}(\mathcal{H}, \mathcal{V}_i)$
- 13: // Only the four sub-triangles are checked.
- 14: $t, \mathbf{n}, M_{\text{hit}} \leftarrow \text{LocalIntersectionCompute}(\mathcal{V}_i, \mathbf{r})$
- 15: **end for**
- 16: **return** $t, \mathbf{n}$

intersects a series of Gaussian primitives, accumulating color and opacity as in standard 3DGS [13]. These accumulated colors are subsequently used for loss computation noted in Eq. 10.

Backward During backpropagation, the gradients are twofold. First, color gradients are propagated along each ray to all Gaussian primitives that contributed to the pixel, optimizing the underlying scene’s parameters (position, opacity, SH, etc.). Different from 3DGRUT [18], our method additionally computes the gradients of the rendered color with respect to the refracted ray’s origin $\mathbf{p}$ and direction ω_o . These gradients are then propagated back to the parameters of our hierarchical surface model (i.e., the weights of the MLP $\mathcal{H}(x, y)$). This joint optimization allows the framework to simultaneously reconstruct the underlying scene and the refractive surface geometry that best explains the observed images. We detail the gradient computation in the supplementary material.

3.4 Implementation Details

We build our method upon the open-source 3DGRUT [18, 29] codebase, adapting it for refractive ray tracing. We implement our water height map in PyTorch and accelerate the ray–mesh intersection and subdivision using NVIDIA OptiX and custom CUDA kernels. We train our model for 15k steps across all experimental scenes. More details are in the supplementary material.

Loss Functions Consistent with prior works [13, 18], we conduct optimization under multi-view image supervision, employing a combination of $\mathcal{L}_{L1} = \|\mathbf{C} - \mathbf{C}_{gt}\|_1$ loss and SSIM loss $\mathcal{L}_{\text{SSIM}} = 1 - \text{SSIM}(\mathbf{C}, \mathbf{C}_{gt})$ as the primary supervision signal

between the rendered image $\mathbf{C}$ and the ground-truth image $\mathbf{C}_{gt}$. During training, we observe that sparse-view regions produce high-opacity floaters, which occlude subsequent Gaussians and cause gradient vanishing as analyzed in [26]. To mitigate this, we additionally introduce an opacity loss $\mathcal{L}_\alpha$ that penalizes high opacity values across all N Gaussians:

$$\mathcal{L}_\alpha = \frac{1}{N} \sum_{i=1}^N \alpha_i^2. \quad (9)$$

As illustrated in Fig. 4, this simple regularization effectively encourages transparency and stabilizes optimization, especially in sparsely-viewed regions. Finally, our total loss is a weighted sum of these components, where we set $\lambda_1 = 0.8$, $\lambda_2 = 0.2$ and $\lambda_3 = 0.007$:

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_{\text{L1}} + \lambda_2 \mathcal{L}_{\text{SSIM}} + \lambda_3 \mathcal{L}_\alpha. \quad (10)$$

4 Experiments

4.1 Experimental Setup

Datasets To rigorously validate our method, we conduct evaluation on datasets where multi-view observation of an underwater scene is captured from above the non-planar water surface. Specifically, we select two datasets covering both synthetic and real-world scenarios:

NeRFrac Dataset. Our experiments cover the entire NeRFrac [35] benchmark, which includes **4 real-world scenes** and **3 synthetic scenes**, each with 8 training views and 1 test view looking vertically down onto the water surface. The image resolution of the real dataset is 384×512 , while that of the synthetic dataset is 392×392 .

RefracGS Dataset. To further evaluate our method under stronger refractive effects, we construct the RefracGS dataset. Compared with NeRFrac’s narrow-baseline setup, RefracGS features scenes captured with larger camera angular coverage, where rays hit the water surface at higher incidence angles, amplifying refraction and making reconstruction more challenging. Each scene consists of 24 training views in a surround-view configuration and 6 test views. In addition, RefracGS offers ground-truth water-surface geometry, which is not publicly accessible in the NeRFrac dataset. Further dataset details and rendering settings are provided in the supplementary material.

Baselines We benchmark our method against both refractive reconstruction and general scene reconstruction approaches. For refractive reconstruction, we include WSGS [33], which is constrained to planar refractive water surfaces, and NeRFrac [35], which represents the state-of-the-art in modeling arbitrary, non-planar refractive fluid surface. For general radiance field reconstruction, we include representative methods across different paradigms: Mip-NeRF [1], TensorRF [4], Plenoxels [8], 3DGS [13], and 3DGRT [18]. This selection covers both

Table 1: Quantitative experimental results. Our method achieves state-of-the-art novel view synthesis quality and water surface reconstruction accuracy, while maintaining fast training speed and real-time rendering frame rates. († indicates methods specifically designed for refractive scenes.)

Dataset	Method	PSNR↑	SSIM↑	LPIPS↓	Train↓	FPS↑	RMSE(cm)↓
NeRFrac Real	Mip-NeRF	10.385	0.221	0.916	~15 h	< 1	-
	TensoRF	22.973	0.809	0.139	11 min	< 1	-
	Plenoxels	14.867	0.389	0.631	9 min	61	-
	3DGS	27.711	0.864	0.130	3 min	857	-
	3DGRT	27.732	0.842	0.164	16 min	157	-
	WSGS†	22.449	0.679	0.279	106 min	129	-
	NeRFrac†	28.146	0.876	0.167	149 min	< 1	-
	Ours†	30.671	0.913	0.129	10 min	242	-
NeRFrac Synthetic	Mip-NeRF	12.414	0.382	0.870	~15 h	< 1	-
	TensoRF	16.223	0.532	0.601	11 min	< 1	-
	Plenoxels	13.334	0.407	0.806	12 min	72	-
	3DGS	22.186	0.730	0.300	3 min	704	-
	3DGRT	21.929	0.689	0.363	27 min	89	-
	WSGS†	20.868	0.726	0.271	107 min	153	-
	NeRFrac†	34.381	0.944	0.149	77 min	< 1	-
	Ours†	35.816	0.952	0.136	12 min	203	-
RefracGS	Mip-NeRF	13.874	0.479	0.808	~15 h	< 1	-
	TensoRF	14.125	0.362	0.658	13 min	< 1	-
	Plenoxels	13.983	0.383	0.790	12 min	23	-
	3DGS	17.482	0.428	0.486	5 min	593	-
	3DGRT	18.019	0.424	0.545	14 min	182	-
	WSGS†	17.915	0.424	0.484	105 min	124	-
	NeRFrac†	17.837	0.405	0.624	164 min	< 1	3.655
	Ours†	30.224	0.933	0.098	11 min	124	0.115

volumetric and point-based representations, providing a comprehensive baseline for evaluating our approach. All baselines were trained with their default settings. For fairness, we also report 3DGS and 3DGRT results trained with fewer iterations (under the same setting as ours, 15k instead of default 30k), provided in the supplementary material.

Metrics We evaluate the performance from two aspects: *novel-view synthesis quality* and *refractive surface reconstruction accuracy*. For novel view synthesis quality, we report PSNR, SSIM [27], and LPIPS [36] scores. For refractive surface reconstruction accuracy, we report the root mean square error (RMSE) of the minimum distance between the intersection point of each ray from the test views and the ground-truth water surface. All experiments were conducted on a single RTX 4090 GPU.

4.2 Experimental Results

Quantitative Results Tab. 1 presents the quantitative results. On the NeRFrac dataset, our method significantly outperforms all existing approaches. Compared to the previous state-of-the-art, it achieves a PSNR increase of over 2+ on

Table 2: Ablation study results. The ablations show that the components of our method collectively provide a favorable balance between reconstruction quality and computational efficiency.

Method Metric	PSNR↑	SSIM↑	LPIPS↓	Train↓	FPS↑
(a) w/o Water Height Map	27.782	0.869	0.168	22 min	57
(b) w/o Normal Smoothing	32.643	0.928	0.133	11 min	232
(c) w/o Recursive Intersection	32.966	0.929	0.132	20 min	218
(d) w/o Opacity Loss	31.927	0.920	0.125	12 min	224
Full Model	32.876	0.929	0.132	11 min	223

real scenes and 1.4+ on synthetic scenes. While WSGS formulates an physics-based refraction model, its performance inevitably degrades on these non-planar datasets due to its strong reliance on a perfectly planar water surface. On the more challenging RefracGS dataset, all prior methods, including NeRFrac, struggle to handle severe refractive distortions, while our approach consistently delivers high-fidelity novel views and precise refractive-surface reconstruction.

Benefiting from our efficient refractive-surface intersection and the rendering efficiency of Gaussian representations, our method trains substantially faster than NeRFrac while maintaining real-time rendering. It also converges more quickly than 3DGRT, as the absence of refraction modeling in 3DGRT leads to inaccurate scene learning and excessive cloning and splitting of Gaussians, which slow down optimization. Its longer default training schedule further increases runtime.

Qualitative Results Fig. 5 presents qualitative comparisons demonstrating the superiority of our method. On the NeRFrac dataset, both our method and NeRFrac achieve comparable reconstruction quality, but our approach better preserves fine textures and high-frequency details, whereas NeRFrac produces noticeably smoother results. On the more challenging RefracGS dataset, where refractionve distortions are severe, existing methods fail to reconstruct consistent views. In contrast, our method achieves photorealistic rendering across all test scenes.

Ablation Studies We perform ablation studies to analyze the contribution of each module to reconstruction quality. Specifically, we conduct four independent experiments: replacing our **(a) water height map** (3.2) with NeRFrac’s refractive field, disabling **(b) normal smoothing** (3.2), disabling **(c) recursive intersection** (3.2), and disabling the **(d) opacity loss** (3.4). Results are summarized in Tab. 2, all conducted on the NeRFrac real and synthetic dataset. Our refractive-surface representation provides significant improvements in both accuracy and efficiency compared to the refraction-field representation. The normal-smoothing strategy enhances rendering quality by promoting smoother surface shading, with only minor computational overhead. Recursive subdivision slightly reduces rendering quality, as a few rays may miss micro-triangles, but it shortens training time by nearly 50%, offering a favorable efficiency–accuracy trade-off.

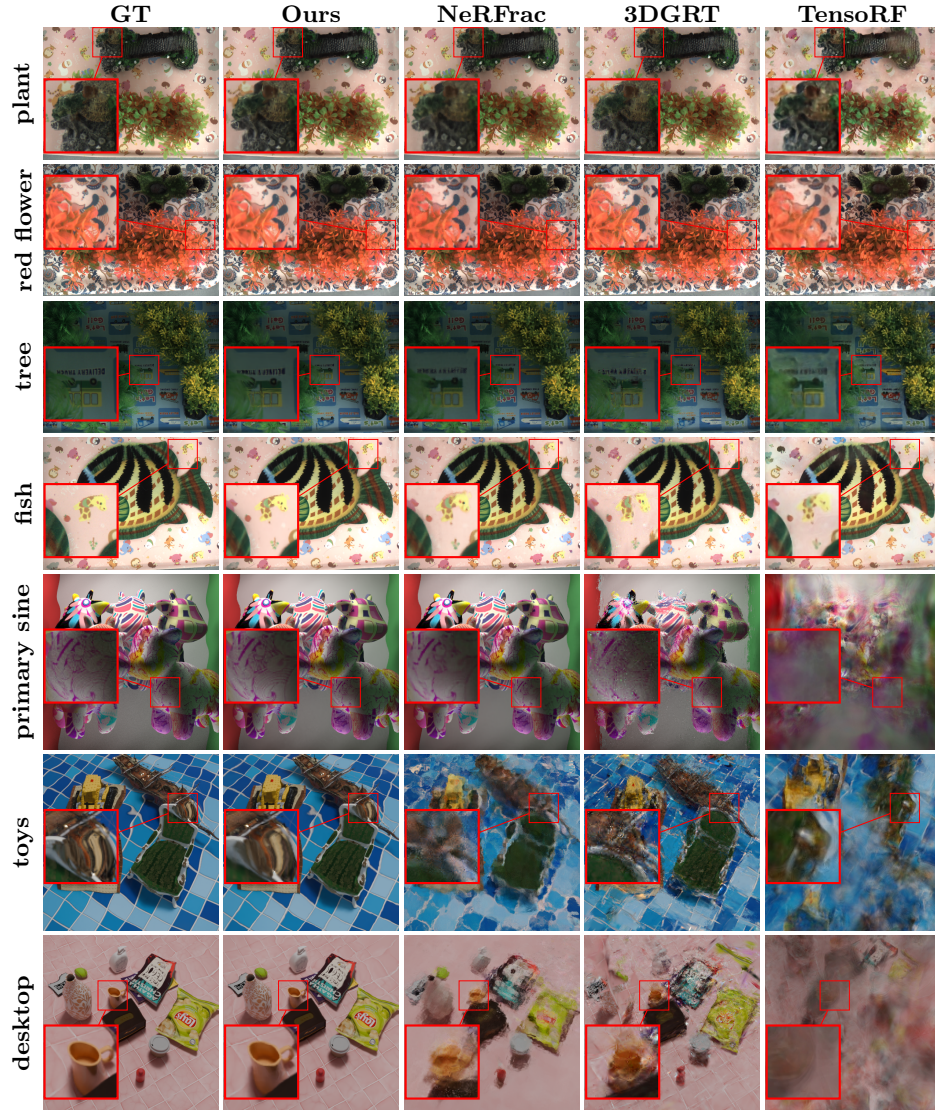


Fig. 5: Qualitative comparison with existing methods. On the NeRFrac dataset, our method preserves fine textures and high-frequency details better than NeRFrac. On the more challenging RefracGS dataset—featuring wider viewpoint coverage and stronger refractive effects—our method consistently produces high-fidelity renderings, whereas existing methods fail to learn accurate scene representations under such conditions, leading to noticeable artifacts and incorrect novel-view synthesis.

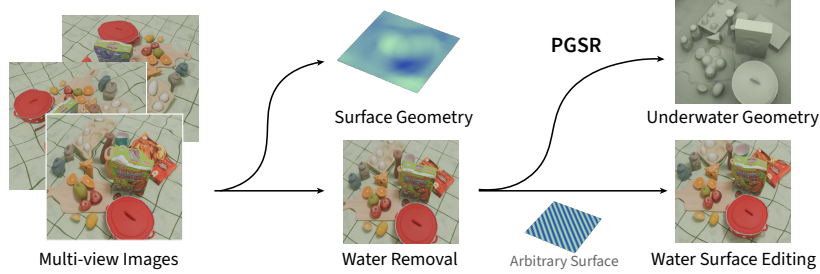


Fig. 6: Water removal and surface editing Benefiting from our physically-based modeling approach, our method naturally decouples the water surface geometry from underwater objects. This capability enables the removal of surface refraction, as well as water surface editing using arbitrary meshes. Crucially, our precise refraction modeling allows these refraction-removed views to serve directly as inputs for methods like PGSR to achieve complete geometric reconstruction of the underlying scene.

Finally, the opacity loss effectively suppresses local floaters, resulting in noticeably improved visual quality.

4.3 Applications

Water Removal Similar to NeRFrac, our approach is also capable of removing the refractive water-surface effect. After training, this can be achieved by disabling the refraction computation that bends ray trajectories during rendering. We additionally evaluate the water removal image quality of our proposed method and NeRFrac on *RefracGS dataset*, as shown in Tab. 3. Benefiting from our high-precision water surface representation, we achieve high-quality water removal results, which further enable us to extract the geometry of underwater objects from the dewatered images using geometry extraction methods such as PGSR [5], as shown in Fig. 6.

Water Surface Editing Compared with NeRFrac, an additional advantage of our explicit water surface representation is its support for surface editing. Once a scene has been trained, the water surface can be replaced with any arbitrary mesh without the need for retraining. Fig. 6 illustrates this capability, where we substitute the original surface with a triangular wave surface.

Table 3: Metrics of water removal quality. Our method yields more reliable results compared with NeRFrac.

Method	Metric	PSNR↑	SSIM↑	LPIPS↓
NeRFrac		14.950	0.380	0.563
Ours		23.151	0.869	0.110

5 Conclusions and Limitations

Conclusions We present RefracGS, a framework for reconstructing scenes through refractive surfaces with 3D Gaussian ray tracing. By introducing an explicit, differentiable refractive-surface model and refraction-aware Gaussian ray tracing,

our method enables accurate and efficient modeling of both the interface and the scene. Experiments on both real and synthetic datasets show that RefracGS outperforms prior methods in rendering quality, reconstruction accuracy, and training efficiency. The explicit surface also supports flexible applications such as water removal and surface editing.

Limitations While our approach demonstrates encouraging performance in handling refractive water surfaces, it inherently charts several promising avenues for future research. First, our formulation prioritizes refraction-induced geometric distortions; accounting for features like reflection and scattering will further enhance the method’s application in a wider range of real-world scenarios. Second, our current framework is tailored for static scenes, serving as a foundational stepping stone toward the more challenging modeling of dynamic underwater environments. Third, we adopt a single-value height field assumption for the water surface, which leaves rare, highly turbulent phenomena like curling or breaking waves as an open invitation for more complex geometric representations. Fourth, although 3DGS guarantees high-fidelity view synthesis, exploring the integration of 2DGS [10] offers a compelling path forward to balance rendering quality with precise surface geometry for exact reconstruction tasks. We provide a more expansive discussion of these forward-looking directions in the supplementary material.

6 Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 62372271 and No. 62595772) and the projects of Beijing Science and Technology Program (Z251100008125028).

References

1. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields (2021)
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. CVPR (2022)
3. Bemana, M., Myszkowski, K., Revall Frisvad, J., Seidel, H.P., Ritschel, T.: Eikonal fields for refractive novel-view synthesis. In: Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings. pp. 1–9 (2022)
4. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: European Conference on Computer Vision (ECCV) (2022)
5. Chen, D., Li, H., Ye, W., Wang, Y., Xie, W., Zhai, S., Wang, N., Liu, H., Bao, H., Zhang, G.: Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. arXiv preprint arXiv:2406.06521 (2024)
6. Deng, W., Campbell, D., Sun, C., Kanitkar, S., Shaffer, M., Gould, S.: Ray deformation networks for novel view synthesis of refractive objects. In: 2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 3106–3116 (2024). <https://doi.org/10.1109/WACV57701.2024.00309>

7. Fan, Z., Wang, K., Wen, K., Zhu, Z., Xu, D., Wang, Z.: Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps (2023)
8. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: CVPR (2022)
9. Gao, F., Zhang, L., Wang, L., Cheng, J., Zhang, J.: Transparent object reconstruction via implicit differentiable refraction rendering. In: SIGGRAPH Asia 2023 Conference Papers. SA '23, Association for Computing Machinery, New York, NY, USA (2023). <https://doi.org/10.1145/3610548.3618236>, <https://doi.org/10.1145/3610548.3618236>
10. Huang, B., Yu, Z., Chen, A., Geiger, A., Gao, S.: 2d gaussian splatting for geometrically accurate radiance fields. In: SIGGRAPH 2024 Conference Papers. Association for Computing Machinery (2024). <https://doi.org/10.1145/3641519.3657428>
11. Huang, L., Ye, D., Dan, J., Tao, C., Liu, H., Zhou, K., Ren, B., Li, Y., Guo, Y., Guo, J.: Transparentgs: Fast inverse rendering of transparent objects with gaussians. *ACM Transactions on Graphics (TOG)* **44**(4), 1–17 (July 2025), <https://doi.org/10.1145/3730892>
12. Jiang, Y., Tu, J., Liu, Y., Gao, X., Long, X., Wang, W., Ma, Y.: Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. *arXiv preprint arXiv:2311.17977* (2023)
13. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics* **42**(4) (July 2023), <https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/>
14. Kheradmand, S., Rebain, D., Sharma, G., Sun, W., Tseng, Y.C., Isack, H., Kar, A., Tagliasacchi, A., Yi, K.M.: 3d gaussian splatting as markov chain monte carlo. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2024), spotlight Presentation
15. Kim, W., Fukiage, T., Oishi, T.: Ref²-nerf: Reflection and refraction aware neural radiance field (2023)
16. Lee, J.C., Rho, D., Sun, X., Ko, J.H., Park, E.: Compact 3d gaussian representation for radiance field. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 21719–21728 (2024)
17. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis (2020)
18. Moenne-Loccoz, N., Mirzaei, A., Perel, O., de Lutio, R., Esturo, J.M., State, G., Fidler, S., Sharp, N., Gojcic, Z.: 3d gaussian ray tracing: Fast tracing of particle scenes. *ACM Transactions on Graphics and SIGGRAPH Asia* (2024)
19. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.* **41**(4), 102:1–102:15 (Jul 2022). <https://doi.org/10.1145/3528223.3530127>, <https://doi.org/10.1145/3528223.3530127>
20. Pan, J.I., Su, J.W., Hsiao, K.W., Yen, T.Y., Chu, H.K.: Sampling neural radiance fields for refractive objects. In: *SIGGRAPH Asia 2022 Technical Communications. SA '22 Technical Communications*, Association for Computing Machinery, New York, NY, USA (2022). <https://doi.org/10.1145/3550340.3564234>, <https://doi.org/10.1145/3550340.3564234>
21. Phong, B.T.: Illumination for computer generated pictures, p. 95–101. Association for Computing Machinery, New York, NY, USA (1998), <https://doi.org/10.1145/280811.280980>
22. Qian, Y., Zheng, Y., Gong, M., Yang, Y.H.: Simultaneous 3d reconstruction for water surface and underwater scene. In: Ferrari, V., Hebert, M., Sminchisescu, C.,

- Weiss, Y. (eds.) Computer Vision – ECCV 2018. pp. 776–792. Springer International Publishing, Cham (2018)
23. Sun, J.M., Wu, T., Yan, L.Q., Gao, L.: Nu-nerf: Neural reconstruction of nested transparent objects with uncontrolled capture environment. *ACM Transactions on Graphics (ACM SIGGRAPH Asia)* (2024)
 24. Tong, J., Muthu, S., Maken, F.A., Nguyen, C., Li, H.: Seeing Through the Glass: Neural 3D Reconstruction of Object Inside a Transparent Container . In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12555–12564. IEEE Computer Society, Los Alamitos, CA, USA (Jun 2023). <https://doi.org/10.1109/CVPR52729.2023.01208>, <https://doi.ieeecomputersociety.org/10.1109/CVPR52729.2023.01208>
 25. Wang, D., Zhang, T., Süssstrunk, S.: NEMTO: Neural Environment Matting for Novel View and Relighting Synthesis of Transparent Objects. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2023)
 26. Wang, L., Ren, Q., Liao, K., Wang, H., Chen, Z., Tang, Y.: Stablegs: A floater-free framework for 3d gaussian splatting (2025), <https://arxiv.org/abs/2503.18458>
 27. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* **13**(4), 600–612 (2004). <https://doi.org/10.1109/TIP.2003.819861>
 28. Wang, Z., Yang, W., Cao, J., Hu, Q., Xu, L., Yu, J., Yu, J.: Neref: Neural refractive field for fluid surface reconstruction and rendering. In: 2023 IEEE International Conference on Computational Photography (ICCP). pp. 1–11 (2023). <https://doi.org/10.1109/ICCP56744.2023.10233838>
 29. Wu, Q., Martinez Esturo, J., Mirzaei, A., Moenne-Loccoz, N., Gojcic, Z.: 3dgut: Enabling distorted cameras and secondary rays in gaussian splatting. *Conference on Computer Vision and Pattern Recognition (CVPR)* (2025)
 30. Xiong, J., Heidrich, W.: In-the-wild single camera 3d reconstruction through moving water surfaces. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 12538–12547 (2021). <https://doi.org/10.1109/ICCV48922.2021.01233>
 31. Yao, Y., Zeng, Z., Gu, C., Zhu, X., Zhang, L.: Reflective gaussian splatting. In: *ICLR* (2025)
 32. Ye, Z., Li, W., Liu, S., Qiao, P., Dou, Y.: Absgs: Recovering fine details for 3d gaussian splatting (2024)
 33. Yeonghun, Y., Hojoon, J., Jaeyoon, L., Taegwan, K., Gyuhyun, K., Choi, J.: Through the water: Refractive gaussian splatting for water surface scenes. *AAAI* (2026)
 34. Yu, Z., Sattler, T., Geiger, A.: Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics* (2024)
 35. Zhan, Y., Nobuhara, S., Nishino, K., Zheng, Y.: Nerfrac: Neural radiance fields through refractive surface. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 18402–18412 (2023)
 36. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: *CVPR* (2018)