Through the Water: Refractive Gaussian Splatting for Water Surface Scenes

Yeonghun Yoon[1], Hojoon Jung[1], Jaeyoon Lee[2], Taegwan Kim[1], Gyuhyun Kim[1], Jongwon Choi[1,2*]

[1]Dept. of Advanced Imaging, GSAIM, Chung-Ang University, Seoul, Korea
[2]Dept. of Artificial Intelligence, Chung-Ang University, Seoul, Korea
{yyhos, hjjung, leejaeyoon}@vilab.cau.ac.kr, {syukusun, gyuhyunkim, choijw}@cau.ac.kr

Abstract

Scenes with water surfaces present a significant challenge for Gaussian Splatting due to the simultaneous presence of refraction and reflection, as well as the difficulty of accurately estimating the geometry of transparent water surfaces. To address this, we propose a novel framework for reconstructing scenes involving both reflection and refraction caused by water surfaces. The water surface is modeled as a trainable plane, and 2D Gaussian ray tracing is applied to account for refraction through the water. We extend 2D Gaussian Splatting by introducing a soft mask parameter and a dual set of Gaussian primitives, which handle both reflected and refracted effects. Our method achieves state-of-the-art performance on newly constructed water surface datasets, including both synthetic and real scenes, and significantly outperforms prior approaches in water-interacting regions. Furthermore, we demonstrate the editability of our model by manipulating the index of refraction to suppress or modify refractive effects, enabling scene transformations into different liquids.

Introduction

Accurately modeling view-dependent effects such as reflection and refraction is crucial for achieving realistic novel view synthesis (NVS), especially in scenes involving reflective and refractive surfaces like water. Recent advances in novel view synthesis have been driven primarily by two representation paradigms: Neural Radiance Fields (NeRF) (Mildenhall et al. 2020) and 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023). However, these methods often fail to accurately account for view-dependent reflections, resulting in poor visual quality.

Several recent studies have proposed methods to incorporate reflection modeling within Gaussian Splatting, including methods using environment maps (Ye, Hou, and Zhou 2024; Jiang et al. 2024; Yao et al. 2025), methods employing Multi-Layer Perceptrons (MLPs) (Zhang et al. 2025), and methods leveraging ray tracing (Xie et al. 2025; Gu et al. 2025). However, existing methods do not consider refraction effects, failing to accurately reproduce scenes containing refractions, such as water surfaces.

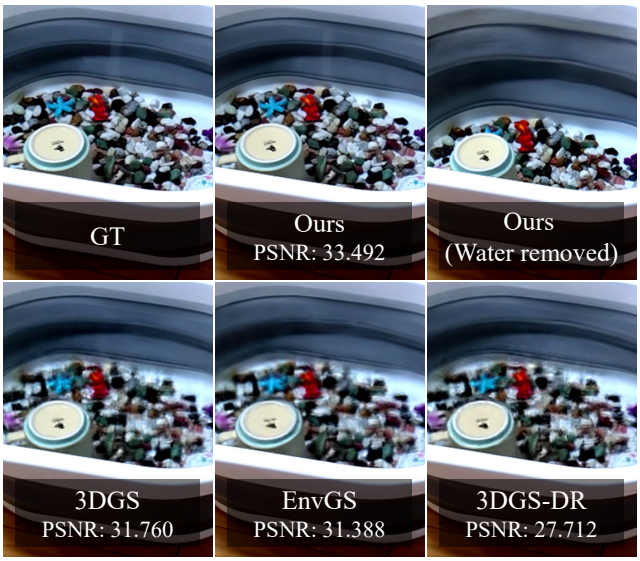

Figure 1: Novel View Synthesis on the Water Real dataset. Our framework reconstructs scenes with refraction and reflection caused by the water surface in high quality, which other methods fail to recover. By explicitly considering refraction, it also enables rendering with water removed.

Even when refraction effects are considered, existing methods still struggle to accurately model refractive surfaces like water, primarily because it is difficult to train transparent surface geometry. One contributing factor is that Gaussian Splatting's training strategy prunes low-opacity Gaussian primitives, making it difficult to maintain Gaussians on water surfaces. Another contributing factor is that depth and normal maps are often biased by Gaussian primitives beneath the water surface, which corrupts the representation of the water surface.

In this paper, we propose a novel framework capable of successfully handling both reflection and refraction effects occurring on the water surface. Our approach introduces a trainable refracting plane and employs a 2D Gaussian ray tracing technique for the water surface. Specifically, our framework uses a base Gaussian, which is a set of Gaussian primitives rendered via 2D Gaussian Splat-

*Corresponding author.

ting (2DGS) (Huang et al. 2024), with soft mask parameters added to represent regions covered by water surface. We also explicitly introduce additional Gaussian primitives called reflected Gaussian for modeling reflection effects and refracted Gaussian for modeling refraction effects. Both the reflected and refracted Gaussians are rendered using 2D Gaussian ray tracing. In addition, we propose a pseudo mask loss that leverages depth information from both the trainable plane and the base Gaussian, encouraging the soft mask to adaptively separate water and non-water regions.

To evaluate our approach, we collected two datasets, *Water Synthetic* and *Water Real*. The Water Synthetic dataset was created in Autodesk Maya by simulating water and rendering with the Arnold renderer. On these datasets, our framework outperforms prior work in both water region and entire scene comparisons, and we also demonstrate post-training edits that modify the index of refraction to simulate different liquids and enable water removal.

The key contributions of our work are as follows:

- The proposed framework introduces reflected and refracted Gaussian primitives, rendered through 2D Gaussian ray tracing, to explicitly capture reflection and refraction effects, respectively.
- A pseudo mask loss is designed to leverage depth information from both the trainable plane and base Gaussian, enabling adaptive separation of water and non-water regions via soft mask learning.
- We introduce *Water Synthetic* and *Water Real*, datasets we created and collected to validate both synthetic and real-world performance.
- We outperform baselines in reconstruction quality on both the water region and the entire scene, and we enable post-training edits that modify the index of refraction and remove the water surface.

Related Work

Neural Radiance Fields. Neural Radiance Fields (NeRF) (Mildenhall et al. 2020) methods have significantly advanced the field of novel view synthesis through implicit representation, demonstrating that MLPs can effectively model density and radiance for photorealistic view synthesis. Subsequent methods explored various strategies to enhance NeRF's speed (Müller et al. 2022; Fridovich-Keil et al. 2022) and quality (Sun, Sun, and Chen 2022; Barron et al. 2022). On the other hand, for sparse-view reconstruction scenarios, RegNeRF (Niemeyer et al. 2022) implemented geometric and color regularization, significantly improving reconstruction quality with limited input views.

Several methods have been proposed to handle reflective materials within the NeRF framework. Ref-NeRF (Verbin et al. 2022) modified the MLP structure to output material properties, effectively modeling view-dependent reflections. SpecNeRF (Ma et al. 2023) incorporated Gaussian directional encoding, capturing both distant and nearby lighting and reflection effects more accurately. ENVIDR (Liang et al. 2023) further improved realism by employing multiple specialized MLPs to predict various physically based rendering components, mixing them appropriately for enhanced rendering quality. However, they fail to handle the refractive effect for the water surface.

Limited research has addressed the modeling of refractions within the NeRF paradigm. NeRFrac (Zhan et al. 2023) models refractive effects by predicting ray–surface intersections using an implicit refractive field and applying Snell's law to compute refracted rays. However, it is only applicable under highly constrained experimental settings, requiring fixed camera arrays.

Gaussian Splatting. Recently, 3DGS has emerged as an explicit alternative to NeRF, offering fast and high-quality rendering. Building on this idea, 2DGS flattened 3D Gaussians into 2D, enhancing surface alignment. MLP based approaches have also been explored. Ref-GS (Zhang et al. 2025) predicts reflection effects by learning an appearance field with a small neural network that takes geometric features from Gaussian primitives as input.

Reflections in Gaussian Splatting have been addressed through various strategies. MirrorGaussian (Liu et al. 2024) modeled reflections by generating mirrored Gaussians across estimated mirror planes, while Mirror-3DGS (Meng et al. 2024) rendered reflections from virtual viewpoints behind the mirrors. Environment map based reflection methods such as 3DGS-DR (Ye, Hou, and Zhou 2024), GaussianShader (Jiang et al. 2024), and Ref-Gaussian (Yao et al. 2025) aimed to approximate reflections efficiently using environment maps. However, due to the limitation of the environment map approach which lacks the ability to represent high-frequency detail, these methods failed to represent refractions that require such fine detail.

Furthermore, ray tracing based reflection approaches have also been explored in Gaussian Splatting. 3D Gaussian Ray Tracing (3DGRT) (Moenne-Loccoz et al. 2024) incorporated ray tracing after rendering to add reflection and refraction effects as a post-processing step, without reconstructing refractive geometry. In contrast, EnvGS (Xie et al. 2025) introduced a separate set of Gaussian primitives, called environment Gaussian, to model reflections within the rendering pipeline via ray tracing. IRGS (Gu et al. 2025) also modeled reflections during rendering by tracing reflected rays and generating additional appearance information, which is blended with the original Gaussian. While these methods are effective for realistic reflection modeling, they fail to handle refractions accurately caused by water surfaces.

Preliminary

2D Gaussian Splatting

First, we introduce 2DGS, which serves as the foundation for 2D Gaussian ray tracing. 2DGS represents each splat by a planar Gaussian disk living in its tangent plane. Concretely, each disk is defined by a center $\mathbf{p}_k$, two orthonormal tangent axes $\mathbf{t}_u, \mathbf{t}_v$, and scale factors s_u, s_v. These are combined into the homogeneous transform of:

$$H = \begin{bmatrix} s_u\,\mathbf{t}_u & s_v\,\mathbf{t}_v & 0 & \mathbf{p}_k \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1}$$

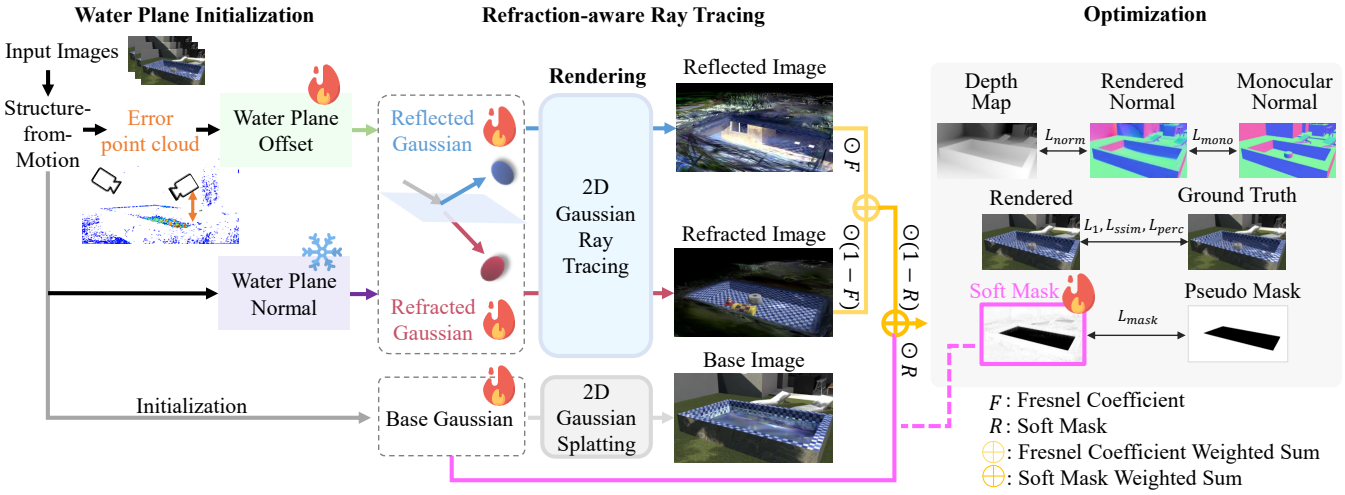

Figure 2: Framework Overview. The pipeline begins with water plane initialization, followed by refraction-aware ray tracing that employs base, reflected, and refracted Gaussians to render water surface effects. The final optimization stage combines RGB losses, normal consistency, and pseudo mask loss to achieve accurate water surface reconstruction.

Then, any point on the disk is located upon the plane of:

$$P(u, v) = H\,(u, v, 1, 1)^{\top}. \tag{2}$$

Finally, the Gaussian weight in local (u, v) coordinates is estimated as follows:

$$G(u, v) = \exp\left(-\frac{u^2 + v^2}{2}\right). \tag{3}$$

The rendering proceeds by ray-splat intersection to find the exact $\mathbf{u} = (u, v)$ for each pixel ray $\mathbf{x}$, followed by alpha-compositing in image space as:

$$\mathbf{c}(\mathbf{x}) = \sum_{i=1} \mathbf{c}_i\, \alpha_i\, \hat{G}_i\big(\mathbf{u}(\mathbf{x})\big) \prod_{j=1}^{i-1} \big(1 - \alpha_j\, \hat{G}_j\big(\mathbf{u}(\mathbf{x})\big)\big), \tag{4}$$

where $\hat{G}$ includes a low-pass fix for degenerate views, and $\mathbf{c}_i$, α_i represents the color and opacity of the i-th Gaussian, respectively. For more details, please refer to 2DGS (Huang et al. 2024).

2D Gaussian Ray Tracing

To overcome the limitations of 2DGS and support accurate specular reflections, 2D Gaussian ray tracing (2DGRT) (Xie et al. 2025; Gu et al. 2025) was developed. In particular, we build on EnvGS, which converts each 2D Gaussian primitive into two triangles by sampling four extremal points in the local tangent plane:

$$\mathbf{V}_{\text{local}} = \{(\pm r, \pm r)\}, \quad r = 3\sigma. \tag{5}$$

These are transformed into world-space triangles via the 4×4 pose matrix and inserted into a Bounding-Volume Hierarchy (BVH) for fast intersection queries. Custom OptiX (Parker et al. 2010) kernels handle rendering by efficiently tracing rays against the BVH-structured scene. At each intersection point $\mathbf{x}_i$, the local Gaussian weight is evaluated as:

$$G_i(\mathbf{u}_i) = G_i\big(\mathbf{H}^{-1}\mathbf{x}_i\big), \tag{6}$$

where $\mathbf{H}^{-1}$ denotes the inverse transformation into the local tangent plane.

Importantly, EnvGS enables both the forward and backward passes in a single front-to-back traversal by re-casting rays and computing gradients with respect to the ray origin $\partial L/\partial \mathbf{o}$ and direction $\partial L/\partial \mathbf{d}$ on the fly. For more details, please refer to EnvGS (Xie et al. 2025).

Method

We introduce Water Scene Gaussian Splatting (WSGS), which is a novel framework that enables reconstruction in scenes with refraction caused by water surfaces. We initialize the trainable plane using results obtained from Structure-from-Motion (SfM) (Schönberger and Frahm 2016) and define the water region on this plane by using a soft mask. Reflected and refracted rays are then computed according to Snell's law with respect to this plane, after which we perform 2D Gaussian ray tracing along these rays. The reflected and refracted colors are then blended using Fresnel equations (Born et al. 1999), enabling accurate rendering of the water region. Finally, we combine the water region color with the non-water region color from 2DGS using the soft mask to create the final image. An overview of the WSGS framework is shown in Figure 2.

Trainable Water Surface Plane

When modeling highly transmissive water surfaces, Gaussian Splatting's opacity-based pruning makes it difficult for primitives to represent the thin water surface. Reflection-aware Gaussian Splatting methods built on 2D Gaussian ray tracing (Gu et al. 2025; Xie et al. 2025) determine the direction of reflected rays by leveraging the normal and depth

information of 2D Gaussian primitives, but the approach is infeasible for the transparent water surface. This is because normal and depth maps obtained via alpha compositing are often disrupted by objects beneath the water surface, which makes it difficult to recover accurate water surface geometry.

To overcome the limitations, we replace the reliance on Gaussian primitives with a single trainable plane that explicitly defines the water surface. This plane is parameterized by a surface normal vector $\mathbf{n}_p$ and a plane offset o_p, ensuring a stable and precise representation of the water surface as:

$$\mathbf{n}_p \cdot \mathbf{z} = o_p, \tag{7}$$

where $\mathbf{z}$ represents the points of the water surface.

However, jointly learning both the plane normal vector $\mathbf{n}_p$ and the offset o_p introduces too many degrees of freedom and makes convergence difficult. Therefore, we pre-estimate the plane normal vector by detecting line segments (Gioi et al. 2010), clustering their intersections via RANSAC (Fischler and Bolles 1981) into three orthogonal vanishing points, and back-projecting these through each camera's intrinsics to extract the vertical axis, and then hold this normal constant during training.

In contrast, o_p is a trainable parameter designed to estimate the precise position of the unknown water surface. Since an incorrect estimation of the water surface can lead to significant errors during ray tracing with refractive effects, we initialize the water offset based on SfM points with high reconstruction error and the positions of the captured cameras. Specifically, o_p is initialized at the midpoint between the average height of the top 5% of SfM points with the highest reconstruction error and the height of the lowest camera.

Refraction-aware Ray Tracing

Since spherical harmonics based color representation in Gaussian Splatting and its standard rendering pipeline fail to capture view-dependent effects such as reflection and refraction, we introduce an extended set of Gaussian primitives: the base Gaussian $\mathbf{P}_{base}$ trained via 2D Gaussian Splatting, a reflected Gaussian $\mathbf{P}_{refl}$ for representing reflection, and a refracted Gaussian $\mathbf{P}_{refr}$ for representing refraction. We then perform 2D Gaussian ray tracing on each Gaussian to render reflection and refraction separately.

For a camera ray with origin $\mathbf{o}_{cam}$ and direction $\mathbf{d}_{cam}$, we compute its intersection with the water surface plane to obtain the ray origins $\mathbf{o}_{refl}$ and $\mathbf{o}_{refr}$. The reflection ray direction $\mathbf{d}_{refl}$ is given by:

$$\mathbf{d}_{refl} = \mathbf{d}_{cam} \; - \; 2\left(\mathbf{d}_{cam} \cdot \mathbf{n}_p\right) \cdot \mathbf{n}_p. \tag{8}$$

Then, we perform ray tracing on reflected ray $(\mathbf{o}_{refl}, \mathbf{d}_{refl})$ through $\mathbf{P}_{refl}$ to obtain the reflection color $\mathbf{c}_{refl}$.

The refraction ray direction $\mathbf{d}_{refr}$ is computed using Snell's law, taking into account the Index of Refraction (IOR) as follows:

$$\mathbf{d}_{refr} = \eta\, \hat{\mathbf{d}}_{cam} + \left(\eta\, \cos\theta_i - \cos\theta_t\right) \mathbf{n}_p, \tag{9}$$

where $\eta = \frac{IOR_{in}}{IOR_{out}}$ is the refractive ratio, with IOR_{in} denoting the refractive index of air (1.000293) and IOR_{out} denoting the refractive index of water (1.333), $\hat{\mathbf{d}}_{cam} \equiv \mathbf{d}_{cam}/|\mathbf{d}_{cam}|$ is the normalized direction vector, $\cos\theta_i \equiv -\,\hat{\mathbf{d}}_{cam} \cdot \mathbf{n}_p$, and $\cos\theta_t \equiv \sqrt{1-\eta^2\left(1-\cos^2\theta_i\right)}$. Adjusting IOR_{out} allows the model to handle other refractive interfaces such as oil without changing the overall architecture.

Rendering Model

The reflection color $\mathbf{c}_{refl}$ and refraction color $\mathbf{c}_{refr}$ obtained in the previous stages are combined to compute the water region color $\mathbf{c}_{water}$ by considering the angle-dependent ratio between reflection and refraction using the Fresnel equations (Born et al. 1999). For computational efficiency, we approximate the Fresnel coefficient F using the Schlick approximation (Schlick 1994) as follows:

$$\begin{aligned} F &= F_0 + \left(1 - F_0\right)\left(1 - \cos\theta_i\right)^5, \\ where\ F_0 &= \left(\frac{IOR_{in} - IOR_{out}}{IOR_{in} + IOR_{out}}\right)^2. \end{aligned} \tag{10}$$

Using F, the final water color is computed as a linear blend of the reflection and refraction colors as follows:

$$\mathbf{c}_{water} = F \cdot\, \mathbf{c}_{refl} + \left(1 - F\right) \cdot\, \mathbf{c}_{refr}. \tag{11}$$

For non-water regions, the color $\mathbf{c}_{out}$ is obtained by rendering the base Gaussian $\mathbf{P}_{base}$.

To ensure that the refraction effect is applied only to water regions, we introduce a soft mask R to distinguish between water and non-water regions. The soft mask R is defined such that values close to 1 indicate the absence of refraction, while values close to 0 indicate the presence of refraction. Taking these considerations into account, the final color $\mathbf{c}_{final}$ is computed as:

$$\mathbf{c}_{final} = \left(1 - R\right) \cdot\, \mathbf{c}_{water} + R \cdot\, \mathbf{c}_{out}. \tag{12}$$

Training

We begin by training the base Gaussian $\mathbf{P}_{base}$ for a sufficiently long number of iterations to ensure high-quality reconstruction in regions unaffected by water refraction. This phase significantly reduces the undesired spread of $\mathbf{c}_{water}$ into non-water regions.

Subsequently, we train the reflected Gaussian $\mathbf{P}_{refl}$ and refracted Gaussian $\mathbf{P}_{refr}$. Unlike $\mathbf{P}_{base}$, $\mathbf{P}_{refl}$ and $\mathbf{P}_{refr}$ are initialized without access to an SfM point cloud. Following the strategy proposed in EnvGS (Xie et al. 2025), we divide the region covered by the SfM point cloud into N^3 sub-grids and randomly sample K points within each grid. We set $N = 32$ and $K = 5$. During the training of $\mathbf{P}_{refl}$ and $\mathbf{P}_{refr}$, we periodically initialize the soft mask parameters of the $\mathbf{P}_{base}$ located below the water surface plane to 0.5 and reset its color to $RGB(0, 0, 0)$. Through this periodic initialization, we can prevent $\mathbf{P}_{base}$ from representing content that should instead be represented by $\mathbf{P}_{refl}$ or $\mathbf{P}_{refr}$.

For the loss functions, we adopt the normal consistency loss $\mathcal{L}_{\text{norm}}$ introduced in 2DGS to guide the learning of normals without ground truth. We also incorporate commonly used losses in the Gaussian Splatting domain: perceptual

Metric	Method	Water Synthetic								Water Real					
		pool		pond		basin		kitchen		fishbowl		bucket		basin real	
		water	entire	water	entire	water	entire	water	entire	water	entire	water	entire	water	entire
PSNR ↑	2DGS	23.971	31.143	24.914	29.384	21.135	29.759	32.960	37.492	24.078	29.797	34.600	35.529	31.535	32.811
	3DGS	26.496	33.161	25.758	30.346	22.028	30.379	33.920	37.999	25.235	31.760	36.883	**36.939**	32.381	33.160
	EnvGS	25.057	31.794	24.995	29.473	21.805	30.380	33.867	37.878	24.670	31.388	36.048	36.374	33.069	33.090
	GaussianShader	19.508	22.905	21.366	26.157	19.473	26.282	29.038	33.332	20.504	23.691	34.625	34.241	24.531	29.455
	3DGS-DR	26.375	33.232	25.101	28.040	21.058	29.080	34.269	37.467	23.355	27.712	36.295	35.709	30.927	33.009
	Ref-GS	21.292	28.613	22.636	21.443	22.077	30.189	31.415	31.465	14.248	27.298	32.846	28.385	30.133	31.629
	Ours	**34.632**	**36.419**	**33.779**	**31.716**	**34.252**	**38.142**	**42.423**	**39.476**	**31.626**	**33.492**	**38.017**	36.534	**35.020**	**33.942**
SSIM ↑	2DGS	0.969	0.930	0.946	0.913	0.967	0.954	0.989	0.979	0.976	0.945	0.996	0.967	0.992	0.951
	3DGS	0.979	0.949	0.956	0.933	0.972	0.961	0.989	0.980	0.980	0.955	0.996	**0.973**	0.992	0.953
	EnvGS	0.974	0.935	0.946	0.913	0.971	0.958	0.989	0.979	0.979	0.949	0.996	0.970	0.992	0.953
	GaussianShader	0.924	0.765	0.899	0.850	0.956	0.912	0.981	0.955	0.961	0.844	0.995	0.951	0.981	0.907
	3DGS-DR	0.978	0.939	0.953	0.890	0.963	0.938	0.990	0.972	0.975	0.920	0.996	0.959	0.990	0.935
	Ref-GS	0.945	0.883	0.921	0.792	0.974	0.944	0.985	0.952	0.990	0.911	0.932	0.936	0.990	0.929
	Ours	**0.996**	**0.958**	**0.990**	**0.943**	**0.997**	**0.983**	**0.997**	**0.984**	**0.994**	**0.964**	**0.997**	0.971	**0.996**	**0.954**
LPIPS ↓	2DGS	0.041	0.141	0.068	0.124	0.043	0.078	0.025	0.048	0.033	0.124	0.006	0.085	0.016	0.105
	3DGS	0.030	0.108	0.061	0.094	0.039	0.063	0.022	0.039	0.027	0.089	0.005	0.068	0.013	0.085
	EnvGS	0.034	0.116	0.063	0.106	0.037	0.067	0.022	0.042	0.028	0.100	0.005	0.068	0.013	0.088
	GaussianShader	0.086	0.376	0.106	0.186	0.055	0.159	0.043	0.105	0.052	0.303	0.008	0.093	0.043	0.176
	3DGS-DR	0.035	0.122	0.068	0.162	0.051	0.103	0.023	0.064	0.190	0.160	0.006	0.084	0.019	0.117
	Ref-GS	0.063	0.199	0.087	0.244	0.087	0.107	0.033	0.097	0.074	0.169	0.008	0.108	0.020	0.122
	Ours	**0.002**	**0.083**	**0.015**	**0.079**	**0.002**	**0.033**	**0.004**	**0.027**	**0.006**	**0.080**	**0.004**	**0.066**	**0.007**	**0.083**

Table 1: Quantitative comparison on the Water Synthetic and Water Real datasets. "water" columns denote results within the interior of the water region, and "entire" columns show results over the entire image.

loss $\mathcal{L}_{\text{perc}}$ (Zhang et al. 2018), SSIM loss $\mathcal{L}_{\text{SSIM}}$ (Wang et al. 2004), and pixel-wise L1 loss $\mathcal{L}_1$:

$$\mathcal{L}_{\text{rgb}} = \lambda_1 \mathcal{L}_1 + \lambda_{SSIM} \mathcal{L}_{\text{SSIM}} + \lambda_{perc} \mathcal{L}_{\text{perc}}. \quad (13)$$

Next, we leverage StableNormal (Ye et al. 2024) as the monocular normal estimation model to provide pseudo normal labels for supervising the normal map prediction. We define the monocular normal loss $\mathcal{L}_{\text{mono}}$ for more stable convergence of the base Gaussian $\mathbf{P}_{base}$ as follows:

$$\mathcal{L}_{\text{mono}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \left(1 - \mathbf{n}_i^\top \mathbf{N}_m\right), \quad (14)$$

where N_p is the number of pixels, $\mathbf{n}$ is the normal map obtained by rendering $\mathbf{P}_{base}$, and $\mathbf{N}_m$ is the normal map predicted by the StableNormal. Since highly transparent water surfaces result in inaccurate monocular normal, we apply $\mathcal{L}_{\text{mono}}$ only for a short number of iterations before starting the learning of reflection and refraction.

Finally, we propose a pseudo mask loss $\mathcal{L}_{\text{mask}}$ to enhance the separation of water regions by supervising the soft mask obtained from $\mathbf{P}_{base}$ using a pseudo mask R_p as follows:

$$\mathcal{L}_{\text{mask}} = \frac{1}{N_p} \|R - R_p\|_2^2. \quad (15)$$

To estimate R_p, we leverage two observations: 1) the base depth map from $\mathbf{P}_{base}$ tends to capture underwater depths due to water's transparency, and 2) water is typically enclosed by closer surrounding objects. Using these, we extract the largest enclosed region on the estimated plane as a 2D pseudo mask R_p.

Then, the final loss function $\mathcal{L}$ is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{rgb}} + \lambda_{mono} \mathcal{L}_{\text{mono}} + \lambda_{norm} \mathcal{L}_{\text{norm}} + \lambda_{mask} \mathcal{L}_{\text{mask}}, \quad (16)$$

where λ_1, λ_{SSIM}, λ_{perc}, λ_{mono}, λ_{norm}, and λ_{mask} are the corresponding loss scaling factors.

Experiments

Experimental Setup

Datasets. Due to the near absence of datasets for novel view synthesis that include refraction caused by water, we created and collected two datasets named *Water Synthetic* and *Water Real*. The Water Synthetic dataset was produced by constructing scenes in Autodesk Maya, adding water dynamics via physical simulation, and rendering the outputs with the Arnold renderer. The Water Real dataset was collected with a Galaxy S25 Ultra in full HD resolution. Each scene was recorded from at least three heights to make refractive differences clear, providing about 200–400 images per scene. Half of the images are used for training, and the remaining half are used for validation. The camera parameters are not provided for both the synthetic and the real datasets, which should be estimated through SfM.

Implementation details. On average, our method requires 212 minutes for training and uses approximately 14.8 GiB of GPU memory, and during inference the model renders at 31 FPS. Additional implementation details can be found in the supplementary material.

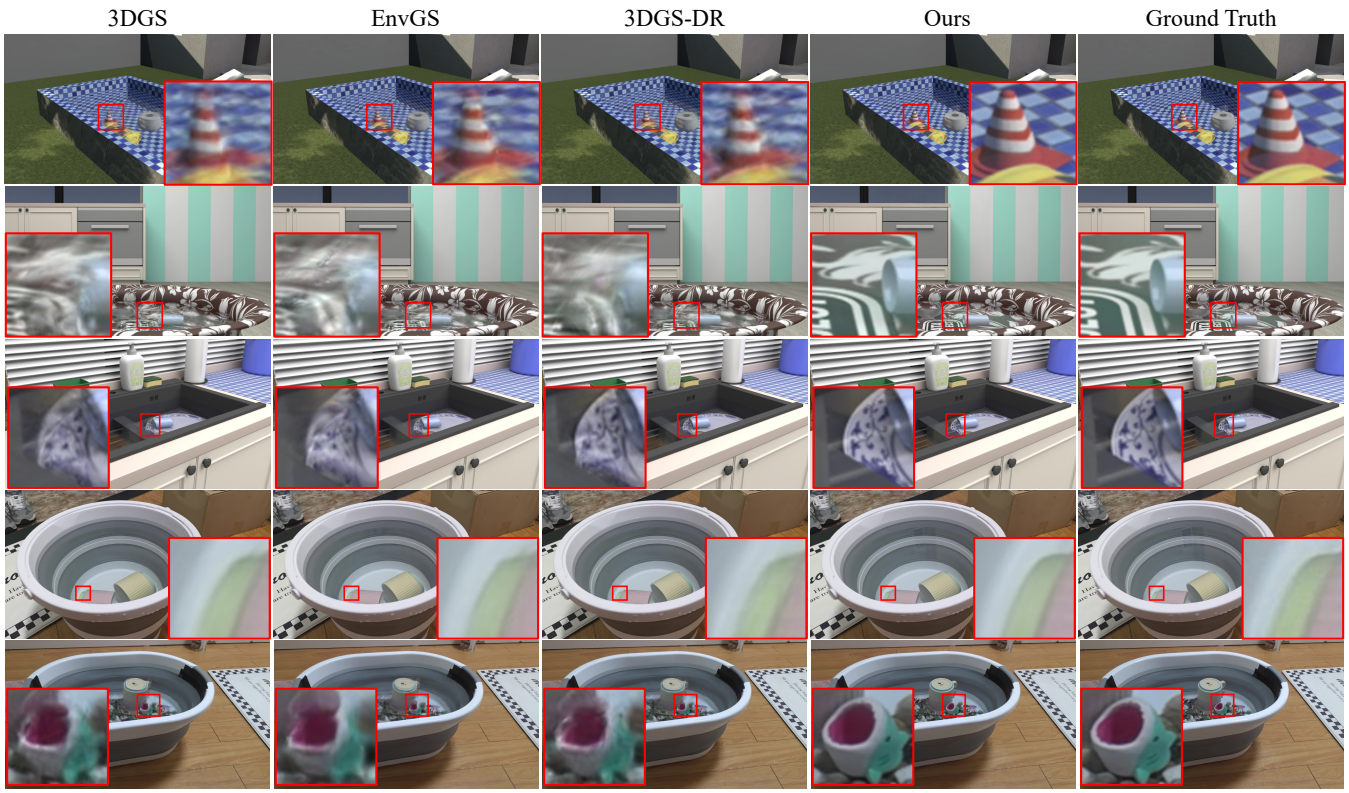

Figure 3: Qualitative comparison on the Water Synthetic and Water Real datasets.

Metrics. To evaluate the quality of the rendered images, we adopted Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM) (Wang et al. 2004), and Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al. 2018).

Comparisons. We compare WSGS against several other Gaussian Splatting models, including **1.** 3DGS and 2DGS which do not model view-dependent effects, **2.** Gaussian-Shader and 3DGS-DR which handle reflection via environment maps, **3.** Ref-GS which models reflection using MLP and **4.** EnvGS which performs reflection through ray tracing.

Comparison Results

We first focused on the water regions to evaluate how well our framework and the baseline methods handle view-dependent effects, specifically reflection and refraction on water surfaces. As represented in the "water" columns of Table 1, even for both the synthetic and real scenes, our method consistently outperformed all baseline methods for the water regions. To assess overall scene quality, we extended the evaluation to entire images, including both water and static regions. This tests the model's ability to reconstruct realistic scenes beyond water effects. As shown in the "entire" columns of Table 1, our method outperformed others in most cases, demonstrating robustness across both regions. An exception was the "bucket" scene from the Water Real dataset, where small incident angles diminished refractive effects and reduced the advantage of our approach.

Method	PSNR↑	SSIM↑	LPIPS↓
w/o trainable plane	30.121	0.943	0.105
w/o soft mask	31.124	0.953	0.083
w/o pseudo mask loss	35.785	0.973	0.057
Ours	**35.817**	**0.973**	**0.056**

Table 2: Component-wise ablation studies.

Method	PSNR↑	SSIM↑	LPIPS↓
w/o pseudo mask loss	17.604	0.887	0.132
Ours	**30.267**	**0.984**	**0.063**

Table 3: Effectiveness of pseudo mask on non-water region.

Qualitatively, as shown in Fig. 3, the baseline approaches that either ignored view-dependent effects or modeled only reflections often produced blurry results with visible visual artifacts in the water regions. In contrast, our method produced sharper and higher-quality reconstructions, successfully capturing the detailed appearance of water surfaces.

Ablation Study and Analysis

To demonstrate the effectiveness of our proposed methods, we conducted ablation studies on the basin data from the Water Synthetic dataset and the fishbowl data from the Water Real dataset. The study comprised two experiments. First, to quantify overall reconstruction quality, we compared the

w/o trainable plane | w/o soft mask | w/o pseudo mask loss | Ours | Ground Truth

Figure 4: Component-wise ablation studies on the Water Synthetic dataset.

Figure 5: Qualitative effectiveness of pseudo mask loss.

Data	PSNR↑	SSIM↑	LPIPS↓
basin	38.099±0.05	0.983±0.00	0.034±0.00
fishbowl	33.466±0.08	0.964±0.00	0.081±0.00

Table 4: Hyperparameter sensitivity analysis.

Ground Truth | Oil (IOR=1.46) | Water Removed

Figure 6: Results of varying the IOR and removing water on the Water Real bucket and Water Synthetic kitchen data.

final rendered output to the ground truth. Second, in cases where the performance gap in the first experiment was small, we carried out an additional analysis to demonstrate how well water and non-water regions were separated by comparing the ground truth with the rendering of $\mathbf{P}_{base}$ in the non-water regions. Further details are provided in the supplementary material.

Pseudo water mask loss. To measure how much the pseudo water mask loss $\mathcal{L}_{\text{mask}}$ contributes to separating water and non-water regions, we conducted an ablation by removing $\mathcal{L}_{\text{mask}}$. Without $\mathcal{L}_{\text{mask}}$, the soft mask fails to properly separate water and non-water regions, as can be seen in Figure 5, degrading performance. The degradation is not obvious in Table 2, since the train and test views are not far apart, so the performance drop caused by incorrect refraction is obscured. However, when we re-evaluate only the non-water regions in Table 3 by comparing the rendering of $\mathbf{P}_{base}$ representing those regions with the ground truth, a clear quantitative decline is observed.

Hyperparameter sensitivity analysis. To evaluate the stability of our method, we conducted experiments by adjusting hyperparameters. As shown in Table 4, even when shifting the plane height by $\pm 5\%$ of the point cloud height range or changing the mask loss weight from the 0.05 to 0.03 and 0.07, the performance remained largely unchanged, demonstrating the robustness of our approach.

IOR Editing. Since the refracted Gaussian is explicitly trained to model refraction, adjusting IOR enables the simulation of different liquids or the removal of water by eliminating both refraction and reflection. As shown in Figure 6, increasing the IOR to 1.46 results in oil-like refraction, whereas setting the IOR to the same value as air and disabling reflection effectively removes the water surface.

Conclusion

In this paper, we introduce WSGS, a novel framework for accurately reconstructing scenes with refraction caused by calm water surfaces. To address the difficulty of representing the water surface with Gaussian primitives alone, we introduce a trainable water surface plane and a soft mask. Our framework extends 2DGS by adding a set of Gaussian primitives called reflected Gaussian and refracted Gaussian and by performing refraction-aware ray tracing on them through the trainable water surface plane. The reflection and refraction effects obtained through ray tracing are combined using the soft mask and the Fresnel equation to produce realistic novel view synthesis in calm water scenes. Experiments on the Water Synthetic and Water Real datasets demonstrate that WSGS outperforms vanilla and reflection-aware Gaussian Splatting in PSNR, SSIM, and LPIPS, while allowing post-training IOR editing and refraction removal.

Limitations and Future Work. Since our method assumes a planar water surface, it cannot handle scenes with dynamic water surfaces during capture. However, it can be extended by incorporating a suitable surface detection module. As future work, we aim to recover complex water surfaces with dynamic Gaussian Splatting.

Acknowledgements

This work was partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [IITP-2023(2024)-RS-2024-00418847, Graduate School of Metaverse Convergence support program; RS-2021-II211341, Artificial Intelligence Graduate School Program (Chung-Ang University)].

References

Barron, J. T.; Mildenhall, B.; Verbin, D.; Srinivasan, P. P.; and Hedman, P. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5470–5479.

Born, M.; Wolf, E.; Bhatia, A. B.; Clemmow, P. C.; Gabor, D.; Stokes, A. R.; Taylor, A. M.; Wayman, P. A.; and Wilcock, W. L. 1999. *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Cambridge University Press, 7 edition.

Fischler, M. A.; and Bolles, R. C. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6): 381–395.

Fridovich-Keil, S.; Yu, A.; Tancik, M.; Chen, Q.; Recht, B.; and Kanazawa, A. 2022. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5501–5510.

Gioi, R.; Jakubowicz, J.; Morel, J.-M.; and Randall, G. 2010. LSD: A Fast Line Segment Detector with a False Detection Control. *IEEE transactions on pattern analysis and machine intelligence*, 32: 722–32.

Gu, C.; Wei, X.; Zeng, Z.; Yao, Y.; and Zhang, L. 2025. IRGS: Inter-Reflective Gaussian Splatting with 2D Gaussian Ray Tracing. In *CVPR*.

Huang, B.; Yu, Z.; Chen, A.; Geiger, A.; and Gao, S. 2024. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery.

Jiang, Y.; Tu, J.; Liu, Y.; Gao, X.; Long, X.; Wang, W.; and Ma, Y. 2024. GaussianShader: 3D Gaussian Splatting with Shading Functions for Reflective Surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5322–5332.

Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4).

Liang, R.; Chen, H.; Li, C.; Chen, F.; Panneer, S.; and Vijaykumar, N. 2023. Envidr: Implicit differentiable renderer with neural environment lighting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 79–89.

Liu, J.; Tang, X.; Cheng, F.; Yang, R.; Li, Z.; Liu, J.; Huang, Y.; Lin, J.; Liu, S.; Wu, X.; et al. 2024. MirrorGaussian: Reflecting 3D Gaussians for Reconstructing Mirror Reflections. In *ECCV*.

Ma, L.; Agrawal, V.; Turki, H.; Kim, C.; Gao, C.; Sander, P.; Zollhöfer, M.; and Richardt, C. 2023. SpecNeRF: Gaussian Directional Encoding for Specular Reflections. arXiv:2312.13102.

Meng, J.; Li, H.; Wu, Y.; Gao, Q.; Yang, S.; Zhang, J.; and Ma, S. 2024. Mirror-3DGS: Incorporating Mirror Reflections into 3D Gaussian Splatting. arXiv:2404.01168.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

Moenne-Loccoz, N.; Mirzaei, A.; Perel, O.; de Lutio, R.; Esturo, J. M.; State, G.; Fidler, S.; Sharp, N.; and Gojcic, Z. 2024. 3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes. *ACM Transactions on Graphics and SIGGRAPH Asia*.

Müller, T.; Evans, A.; Schied, C.; and Keller, A. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.*, 41(4): 102:1–102:15.

Niemeyer, M.; Barron, J. T.; Mildenhall, B.; Sajjadi, M. S.; Geiger, A.; and Radwan, N. 2022. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5480–5490.

Parker, S. G.; Bigler, J.; Dietrich, A.; Friedrich, H.; Hoberock, J.; Luebke, D.; McAllister, D.; McGuire, M.; Morley, K.; Robison, A.; and Stich, M. 2010. OptiX: a general purpose ray tracing engine. *ACM Trans. Graph.*, 29(4).

Schlick, C. 1994. An Inexpensive BRDF Model for Physically-based Rendering. *Computer Graphics Forum*, 13(3): 233–246.

Schönberger, J. L.; and Frahm, J.-M. 2016. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sun, C.; Sun, M.; and Chen, H. 2022. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. In *CVPR*.

Verbin, D.; Hedman, P.; Mildenhall, B.; Zickler, T.; Barron, J. T.; and Srinivasan, P. P. 2022. Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields. *CVPR*.

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612.

Xie, T.; Chen, X.; Xu, Z.; Xie, Y.; Jin, Y.; Shen, Y.; Peng, S.; Bao, H.; and Zhou, X. 2025. EnvGS: Modeling View-Dependent Appearance with Environment Gaussian. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, 5742–5751.

Yao, Y.; Zeng, Z.; Gu, C.; Zhu, X.; and Zhang, L. 2025. Reflective Gaussian Splatting. In *ICLR*.

Ye, C.; Qiu, L.; Gu, X.; Zuo, Q.; Wu, Y.; Dong, Z.; Bo, L.; Xiu, Y.; and Han, X. 2024. StableNormal: Reducing Diffusion Variance for Stable and Sharp Normal. *ACM Transactions on Graphics (TOG)*.

Ye, K.; Hou, Q.; and Zhou, K. 2024. 3D Gaussian Splatting with Deferred Reflection. In *ACM SIGGRAPH 2024 Conference Papers*, SIGGRAPH '24. New York, NY, USA: Association for Computing Machinery. ISBN 9798400705250.

Zhan, Y.; Nobuhara, S.; Nishino, K.; and Zheng, Y. 2023. Nerfrac: Neural radiance fields through refractive surface. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 18402–18412.

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 586–595.

Zhang, Y.; Chen, A.; Wan, Y.; Song, Z.; Yu, J.; Luo, Y.; and Yang, W. 2025. Ref-GS: Directional Factorization for 2D Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.