



CrystalNet: Texture-Aware Neural Refraction Baking for Global Illumination

Z. Zhang¹  and E. Simo-Serra¹ 

¹Waseda University, Japan

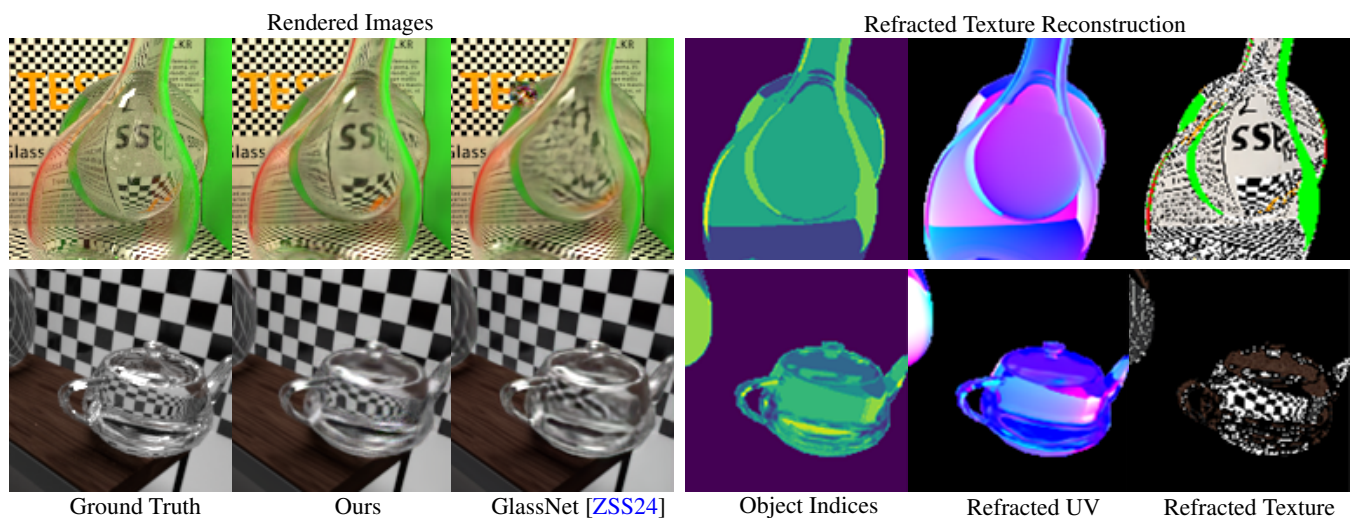


Figure 1: Rendered images by our proposed approach. We introduce our novel neural rendering framework, CrystalNet, for high quality refraction rendering. We use a neural network to bake UV coordinates and object information from refracted rays to reconstruct high frequency color information on refractive surfaces. Compared to previous baking approaches such as GlassNet [ZSS24], our method captures refraction effects more accurately with less loss of details.

Abstract

Neural rendering bakes global illumination and other computationally costly effects into the weights of a neural network, allowing to efficiently synthesize photorealistic images without relying on path tracing. In neural rendering approaches, G-buffers obtained from rasterization through direct rendering provide information regarding the scene such as position, normal, and textures to the neural network, achieving accurate and stable rendering quality in real-time. However, due to the use of G-buffers, existing methods struggle to accurately render transparency and refraction effects, as G-buffers do not capture any ray information from multiple light ray bounces. This limitation results in blurriness, distortions, and loss of detail in rendered images that contain transparency and refraction, and is particularly notable in scenes with refracted objects that have high-frequency textures. In this work, we propose a neural network architecture to encode critical rendering information, including texture coordinates from refracted rays, and enable reconstruction of high-frequency textures in areas with refraction. Our approach is able to achieve accurate refraction rendering in challenging scenes with a diversity of overlapping transparent objects. Experimental results demonstrate that our method can interactively render high quality refraction effects with global illumination, unlike existing neural rendering approaches. Our code can be found at <https://github.com/ziyangz5/CrystalNet>

CCS Concepts

• *Computing methodologies* → *Rendering*; *Neural networks*;

1. Introduction

Using neural networks to “bake” or precompute global illumination and render photorealistic images has provided a new way to solve the key challenge of real-time global illumination in computer graphics. Traditionally, global illumination is achieved by using expensive path tracing [Kaj86], which is hard to achieve real-time performance, or cheap precomputed methods [GSHG98] with limited light transportation type supported. Neural rendering instead uses end-to-end deep models to learn and bake global illumination and scene representation into the weights, supervised by path-tracing images.

Recent neural rendering methods utilize G-buffers, which simplify input handling by avoiding spatial data [GRPN20, DPD22, GMX22]. However, the usage of G-buffers has introduced a problem with transmissive object rendering. In such cases, transmissive surfaces obscure crucial shading information in the G-buffers, complicating the rendering process. Zhang and Simo-Serra [ZSS24] proposed a method that separates transmissive surfaces in G-buffer to improve the transparency quality but assumes a low index of refraction. Currently, no existing method is capable of baking specular refraction effects because the G-buffer of the transmissive surface do not provide information on incoming rays, thus losing high-frequency information from refracted images. To address this limitation, we introduce a neural rendering framework, which we denote *CrystalNet*, which consists of a refraction buffer (R-buffer) generator and rendering neural network. First, the R-buffer generator predicts UV coordinates and object indices of the incoming rays on transmissive surfaces from the G-buffers, which allows for texture lookups, allowing for recovering high frequency texture information. Afterwards, the neural renderer combines all the information to synthesize the scene. The entire framework is optimized for a specific scene by baking all the necessary information for rendering into the weights of the neural networks, and can generate high quality specular refraction effects such as caustic effects.

Our research demonstrates the practicality of using low-frequency information baked by neural models to reconstruct high-frequency textures. In comparison to directly representing texture in neural networks, our proposed method, *CrystalNet*, significantly enhances the quality of specular refraction rendering. This advancement has the potential to enable real-time global illumination in computer graphics without ray computations, making it more accessible and efficient for a wide range of applications.

Our main contributions in this paper include:

1. A novel neural rendering framework, *CrystalNet*, for rendering of scenes with high quality specular refraction effects with full global illumination including soft shadows and caustics.
2. Refraction buffers (R-buffers) baking information into data that is usable by neural networks based on refracted UV coordinates on transmissive surfaces.
3. In-depth evaluation and comparison with existing approaches.

2. Related Work

In this section, we review the most related works in traditional refraction rendering and neural rendering.

2.1. Traditional Refraction Rendering

Specular refraction rendering has been a challenging problem in computer graphics. Early attempts focus on refraction effects on arbitrary shapes [Wym05a] under distant lighting. Wyman [Wym05b] extended this method to handle nearby geometries. Later, Oliveira and Brauwiers [OB07] improved it to handle more complex geometries. G  nevaux *et al.* [GLD06] used pre-computed rays to approximate light paths, but such baking procedure is usually high-cost. To avoid expensive precomputation and increase the capability to render deformable objects, the ray-object intersection in 2D texture space methods are proposed [ED06]. Despite the efficiency and stability of traditional refraction rendering approaches, they typically suffer from limited lighting types, constrained geometries’ complicity, unsupported overlapping transmissive objects, and the incapability of combining with other global illumination. In this paper, we propose a neural rendering method that takes advantage of the flexibility of neural rendering methods to solve the problems above and simultaneously handle refraction effects.

Refraction with roughness and BTDF is also a major research topic in refraction rendering [ED06, DRBS*11, GP16]. However, it is out of the scope of this paper as neural rendering methods are often highly capable of capturing low-frequency global illumination. We include an analysis on rough surface refraction in Sec. 4.3.

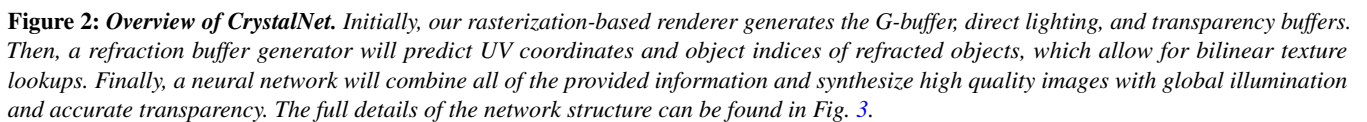
2.2. Neural Rendering

As a group of data-driven rendering methods, neural rendering can theoretically bake various light transportation types under area lighting and effectively and efficiently unify refraction rendering with other global illumination.

Ren *et al.* [RWG*13] initiated the usage of neural networks to bake relightable global illumination with fixed objects and limited types of lighting. Encoding scenes to latent vectors was introduced by Eslami *et al.* [EJRB*18]. Later, Granskog *et al.* [GRPN20] expands the encoding into explainable compositional latent vectors.

Several later works empathize with the improvement of inputs to the neural rendering models. Gao *et al.* [GMX22] explored the usage of radiance cues to improve the rendering quality. *Active Exploration* proposed by Diolatzis *et al.* [DPD22] improved the sampling procedure of the training images. Zhang and Simo-Serra [ZSS24] recognized the limitation of using G-buffer as inputs to neural rendering models caused by the occlusion by transparent objects and proposed *GlassNet* to improve the transparency rendering quality. However, not all previous works have addressed the effects of refraction on transmissive objects. Even the closest work, *GlassNet*, only considers transmissive objects with refraction index close to 1. Instead, our proposed method, *CrystalNet*, is able to capture the refraction effects even on smooth refractive objects with high-frequency refracted information. To losslessly preserve all information of transparent objects, our method utilizes the permutation-invariant network structure proposed in *GlassNet*.

Real-time path tracing and denoising, aided by neural networks, are rapidly developing areas. Although these methods are generally orthogonal to neural rendering techniques, they still have some limitations that neural rendering can address. Neural radiance



Recently, Neural Radiance Fields (NeRF) based methods have significantly contributed to the neural rendering field. We refer the development of NeRF to a comprehensive survey [GGH*22]. Various latest approaches [BMRF*22, PSH*22, LLW*23, CLZ*23, DCS*24] attempt to extend the ability of NeRF into refractive objects. However, such methods often restricted to static objects. Furthermore, NeRF-based methods also require ray marching during rendering, which increases the computational burden. Compared to NeRF-based approaches, our method does not require ray marching, and supports dynamic scenes.

We introduce our neural rendering framework, *CrystalNet*. Notations used in this paper are summarized in Tab. 1. Our approach consists of two major components: a refraction buffer (R-buffer) generator, and a rendering model. The R-buffer generator predicts UV coordinates and object indices of refracted objects from direct lighting information, which allow for texture lookups. Afterwards, all the computed information is used by the rendering model to synthesize a high quality image of the scene. Our network mainly utilizes the structure from UNet [RFB15] and *GlassNet*, and the implementation can be found in the code and Fig. 3.

The general objective of neural rendering is to use a neural network to bake the outgoing radiance, L_o at position \mathbf{p} . L_o can be computed by the rendering equation [Kaj86]:

To compute L_o , instead of doing expensive path tracing, neural rendering methods use a neural network F to represent the global illumination in latent space:

To more accurately capture the refraction effects, we explicitly separate the reflection scattering over the upper hemisphere Ω^+ , and the refraction under the lower hemisphere Ω^- :

In the case of baking reflection scattering, it is possible to leverage G-buffers from direct rendering, as neural networks do not need to encode high-frequency information such as textures, but only need to encode global illumination.

In contrast, the refraction component, especially on smooth surfaces, often consists of distorted images of the objects behind the

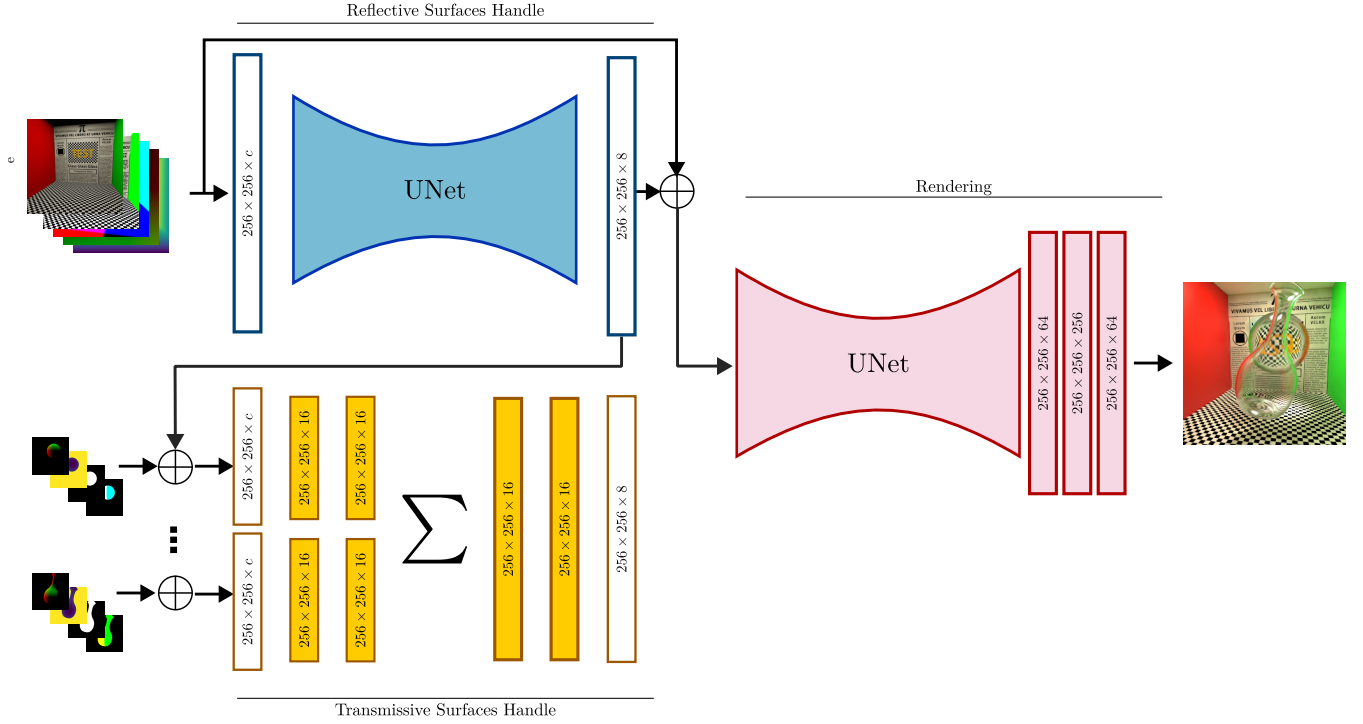


Figure 3: Network structure of CrystalNet. Our network utilizes UNet [RFB15], and GlassNet structure [ZSS24]. All activation functions are leaky ReLU. We use the same network structure for both R-buffer generator and the neural renderer with only the difference of the hidden layers. Specific implementation details can be found in the released code.

refracting surface, with high frequency texture information. Such refraction can not be rendered simply using G-buffers, and it is necessary to either provide high-frequency information of the refracted objects into the rendering neural network through expensive ray tracing, or encode it into the network itself. Specifically, in addition to all the rendering information regarding the transmissive object itself, a neural network, F_t , requires information regarding the high-frequency information of the refracted objects γ . Accordingly, most of the neural rendering models bake γ directly by using an implicit deep model T :

$$\begin{aligned} \int_{\Omega^+} L_i(\mathbf{p}, \omega_i) f_r(\omega_o, \omega_i) |\mathbf{n} \cdot \omega_i| d\omega_i &\approx F_r(\mathbf{p}, \mathbf{n}_r, m_r, \omega_i) \\ \int_{\Omega^-} L_i(\mathbf{p}, \omega_i) f_t(\omega_o, \omega_i) |\mathbf{n} \cdot \omega_i| d\omega_i &\approx F_t(\mathbf{p}, \mathbf{n}_t, m_t, \omega_i, \gamma) \\ &\approx F_t(\mathbf{p}, \mathbf{n}_t, m_t, \omega_i, T(\sigma)) \end{aligned} \quad (4)$$

Note that T can only rely on the neural scene latent representation to bake the high-frequency information of the refraction, leading to blurry and loss of details on high-frequency color or normal textures.

Instead of directly baking the high-frequency information, our method, *CrystalNet*, bakes the lower frequency UV coordinates and then reconstructs the textures directly. Specifically, our refraction baking model, \mathcal{T} , reconstructs γ by generating the R-buffer containing UV coordinates and normals. The overall *CrystalNet* structure is shown in Fig. 2. By applying such an approach, we avoid using

neural networks to directly bake the high-frequency texture and achieve high-quality refraction through texture look-ups instead.

Similar to recent neural rendering approaches, we also use a G-buffer as input to our approach and path tracing images as the ground truth. To maximally retain the information occluded by the transmissive objects, we apply the permutation invariant G-buffer encoding method used in [ZSS24]. By doing so, the G-buffer provided to the rendering model contains all the information behind the refractive models.

3.2. Permutation Invariant Network Structure

We address the occlusion problem by transparent objects on G-buffers by separating the G-buffers of transparent objects into individual buffers. As previous works proposed [QSMG17, ZSS24], a permutation invariant neural network, f , can be approximated by a deep function g combined with a symmetric operator, and then post-processed by an outer neural network k . As described below, we choose addition as the symmetric operator:

$$f(\{x_1, x_2, \dots, x_{\text{num}_t}\}, \hat{\gamma}) = k\left(\sum_{i=1}^{\text{num}_t} h(x_i, \sigma, \hat{\gamma})\right) \quad (5)$$

By applying this structure, our model can process the transparent objects without losing input information, and does not require specific input order of the transparent objects.

Notation	Description
L_o	Outgoing radiance
L_i	Incoming radiance
L_e	Emission radiance
ω_o	Outgoing direction
ω_i	Incoming direction
\mathbf{n}	Normal
m	Material parameters
Ω	Hemisphere
$\Omega^+ \Omega^-$	Upper and lower hemisphere
\mathbf{t}	Object index
\mathbf{p}	Position in 3D space
$\mathbf{p}_{uv}, \mathbf{p}_{uv}^1$	UV coordinates with \mathbf{t} indicating specific object
\mathbf{p}^{UV}	UV maps as 2D tensors with channels
F, \mathcal{F}	Neural networks predicting radiance
T, \mathcal{T}	Neural networks predicting refraction information
σ	Neural scene representation in latent space
γ	Refraction buffer
c, c_p	Color with subscripts indicating specific location
$I[x, y]$	Color on texture or image at coordinate (x, y)
u, v	UV coordinates
w, h	Image width and height
\mathcal{L}	Loss function
λ_{tv}	Weight of total variation loss
num_t	Total number of transmissive objects
$\hat{\cdot}$	All hat notations indicate symbols are predictions
\cdot_r	Subscripts indicate symbols are for reflective objects
\cdot_t	Subscripts indicate symbols are for transmissive objects

Table 1: Glossary of notations used in this paper.

3.3. Refraction Baking: R-Buffer

Our R-buffer generator, \mathcal{T} , takes a G-buffer as an input and generates a tuple of normal and UV coordinates for a refracted objects, \mathbf{t} :

$$\mathcal{T}(\mathbf{p}, \mathbf{n}, m, \omega_i, \sigma, \gamma) = \{\hat{\mathbf{n}}_t, \hat{\mathbf{p}}_{uv}^1, \hat{\mathbf{t}}\} \quad (6)$$

We take advantage of the fact that modern rendering engines use UV coordinates and bilinear interpolation to reconstruct 2D texture of color, normal, *etc.*, to 3D objects. We used the predicted refraction UV coordinates, $\hat{\mathbf{p}}_{uv}^1$, to reconstruct the color information, \hat{c} , from textures of objects by bilinear interpretation:

$$\begin{aligned} \hat{c}_{\mathbf{p}_{uv}}^1 \approx & I[x, y] \cdot (1 - x') \cdot (1 - y') + \\ & I[x + 1, y] \cdot x' \cdot (1 - y') + \\ & I[x, y + 1] \cdot (1 - x') \cdot y' + \\ & I[x + 1, y + 1] \cdot x' \cdot y' \end{aligned} \quad (7)$$

where $x' = \lfloor w \cdot u + 1 \rfloor - w \cdot u$ and $y' = \lfloor h \cdot v + 1 \rfloor - h \cdot v$. Then, the predicted R-buffer, $\hat{\gamma}$, can be used in the rendering model.

Given that UV mapping is highly sensitive to the stability of the generated UV coordinates, we use the total variation loss [ROF92] to enforce the smoothness of generated UV maps. Specifically, the loss function is the combination of L1 reconstruction loss and the total variation loss with a balance coefficient λ_{tv} :

$$\mathcal{L}_{\mathcal{T}}(\mathbf{P}^{UV}, \hat{\mathbf{P}}^{UV}) = \|\mathbf{P}^{UV} - \hat{\mathbf{P}}^{UV}\|_1 + \lambda_{tv} \mathcal{L}_{tv}(\hat{\mathbf{P}}^{UV}) \quad (8)$$

where the total variation loss, \mathcal{L}_{tv} , is defined as:

$$\mathcal{L}_{tv}(\hat{\mathbf{P}}^{UV}) = \sum_{i,j} |\hat{\mathbf{p}}_{i+1,j}^{UV} - \hat{\mathbf{p}}_{i,j}^{UV}| + |\hat{\mathbf{p}}_{i,j+1}^{UV} - \hat{\mathbf{p}}_{i,j}^{UV}| \quad (9)$$

The R-buffer generator is implemented using a permutation invariant multi-layer convolutional neural network described in Sec. 3.2.

3.4. Rendering Model

We next introduce our neural rendering model. It contains three building blocks: a reflection baking network, \mathcal{F}_r , a transmissive baking network, \mathcal{F}_t , and final rendering network \mathcal{R} . For the network architectures, \mathcal{F}_r and \mathcal{R} use a U-Net model [IZZE17], and \mathcal{F}_t uses the permutation invariant multi-layer convolutional neural network described in [ZSS24]. We use G-buffers with separated transmissive objects as inputs to prevent the occlusion to the G-buffer by transmissive objects and maximally retain inputting information as shown in Fig. 2.

The reflection baking network \mathcal{F}_r accepts all information required by rendering equation. As with most other neural rendering approaches, besides \mathbf{p} , n_r , m_r , and ω_i , we take advantage of a rasterization engine to also directly generate the depth buffer, z_r , and direct lighting, d_r , as network inputs. We use Linearly Transformed Cosines [HDHN16] to approximate the direct lighting. \mathcal{F}_r also generates a neural scene representation latent vector, σ used in \mathcal{F}_t .

The transmissive baking network \mathcal{F}_t takes the rendering information of only refraction objects (\mathbf{p} , n_t , m_t , z_t , and ω_i) and, as described in Sec. 3.3, the R-buffer $\hat{\gamma}$ generated by \mathcal{T} the inputs. As we use separated groups of G-buffers for transmissive objects, we need a permutation-invariant network structure, as mentioned in Sec. 3.2, to handle the G-buffer groups with uncertain ordering.

Finally, the final rendering network uses the outputs from \mathcal{F}_r and \mathcal{F}_t to approximate L_o . We use positional encoding to process all the input data, which has proven to be beneficial for neural networks to learn high-frequency information [MST*21]. The full implementation details of the network structure can be found in the released code and Fig. 3.

3.5. Training

We randomly choose the camera position and targeting center in a bounding box with sizes depending on the scenes. Each of the 3D coordinates of the position of movable objects follows a uniform distribution with a given range. We generated our ground truth by using Mitsuba 3 [JSRV22] path tracer. The ground truth for refracted UV maps is generated by our custom renderer using a ray-casting algorithm. \mathcal{F}_r , \mathcal{F}_t and \mathcal{R} are trained as one neural network. To achieve efficiency, \mathcal{T} is separately trained because UV coordinates ground truth is much easier to get, and, therefore, \mathcal{T} can be trained with much more data than the networks using path tracing images as ground truth. The loss function of \mathcal{F}_r , \mathcal{F}_t and \mathcal{R} is the combination



Figure 4: Random selection of images from the dataset.

Parameter	Value
Training set size	1380
Validation set size	120
Test set size	128
Samples per pixel	4096
Learning rate of the renderer	1×10^{-4}
Learning rate of the R-buffer generator	7.5×10^{-5}
Weight decay	1×10^{-4}
Total variation weight, λ_{TV}	1.5×10^{-8}

Table 2: Dataset settings and hyperparameter choices.

of L1 loss and DSSIM [LMCB06], and the loss function for \mathcal{T} is described in Eq. (8). We trained all our models end-to-end and used Adam [KB14] as the optimizer.

4. Experiments

4.1. Experiment Settings

We choose 256×256 as the input and output resolution for the experiments. Note that resolution is not the main focus of this paper. As indicated by previous works, it is feasible to enhance resolution by placing a supersampler [ZSS24] or by using the PixelGenerator as the network backbone to support multi-resolution [GRPN20]. We test our models under three scenes: CORNELLBOX, DESK, and BEDROOM. BEDROOM is created from the Bitterli datasets [Bit16]. CORNELLBOX and DESK are created using existing models and textures in public domain. The setting of datasets and hyperparameter chosen during training are shown in Fig. 4 and Tab. 2. The training time of the neural renderer and R-buffer generator \mathcal{T} are around 6 hours and 16 hours.

In this section, we use L1 reconstruction error, LPIPS [ZIE*18], SSIM [LMCB06], and Peak Signal-to-Noise Ratio (PSNR) as the evaluation metrics. We evaluate the models on transmissive only area to further investigate the performance of our method focusing on the refraction quality. Such areas are evaluated by L1 error and PSNR (T.L1 and T.PSNR). As our method focuses on reconstruct

high frequency texture, for transmissive parts, we also include the L2 error of the amplitude spectrum on frequency domain (T.Amp.):

$$\mathcal{L}_{\text{Amp.}}(I, \hat{I}) = \frac{\sum_{i,j}^{w,h} (\|\text{FFT}(I)_{i,j}\| - \|\text{FFT}(\hat{I})_{i,j}\|)^2}{w \cdot h} \quad (10)$$

We convert the rendered images into the frequency domain by Fast Fourier Transform (FFT).

4.2. Comparison with the State-of-the-Art

We compare our method to other state-of-the-art methods: *GlassNet* [ZSS24], which only focuses on transparent objects with low refraction index, and *Active Exploration* [DPD22] which bakes scene global illumination based on active sampling processes. The qualitative and quantitative results can be found in Fig. 5 and Tab. 3.

In summary, our proposed methods achieve the best specular refraction rendering quality among the compared methods and testing scenes. In CORNELLBOX, other methods completely failed to capture the refracted texture of the background wall, whereas our method reconstructed it successfully. For smaller refraction objects in DESK, *GlassNet* failed to render the checkerboard texture on the teapot. Although *Active Exploration* was able to reconstruct the checkerboard texture, the geometric structure of the refraction was inaccurate. Our method achieves both better texture rendering quality and the precision of the refracted geometry. In BEDROOM, our method also generates the refracted texture much better than other methods.

Furthermore, we compare the performance of the models under the exact training time in Fig. 7 under CORNELLBOX. The experiment shows *GlassNet* is unable to accurately render the scene regardless the length of the training time.

To further demonstrate the advantage of using G-buffers as inputs, we test the ability of our model under variable lighting and index of refraction (IoR) conditions. As shown in Fig. 8 and Tab. 4, *CrystalNet* still significantly outperforms the *GlassNet* even under such challenging conditions. Additionally, our video demonstration can be found the supplementary documents.

We also compared to the real-time path tracing denoising algorithm proposed by Chaitanya *et al.* [CKS*17]. As shown in Fig. 6, we achieve a much better refraction quality than the path tracing denoiser under 32 sample-per-pixel under a similar frame time around 18 to 20 milliseconds.

4.3. Ablation Study

One important decision is what types of information should be included in the R-buffer. We explore whether including refracted normal in R-buffer or not can improve the rendering quality. As shown in Tab. 5, using predicted normals can positively affect the rendering quality. We also investigated the effectiveness of combining texture reconstruction loss and UV coordinates loss when training the R-buffer generator. As shown in Tab. 6, using a combination of color space loss and UV space loss does not improve but worsens the rendering quality. Therefore, we choose not to use

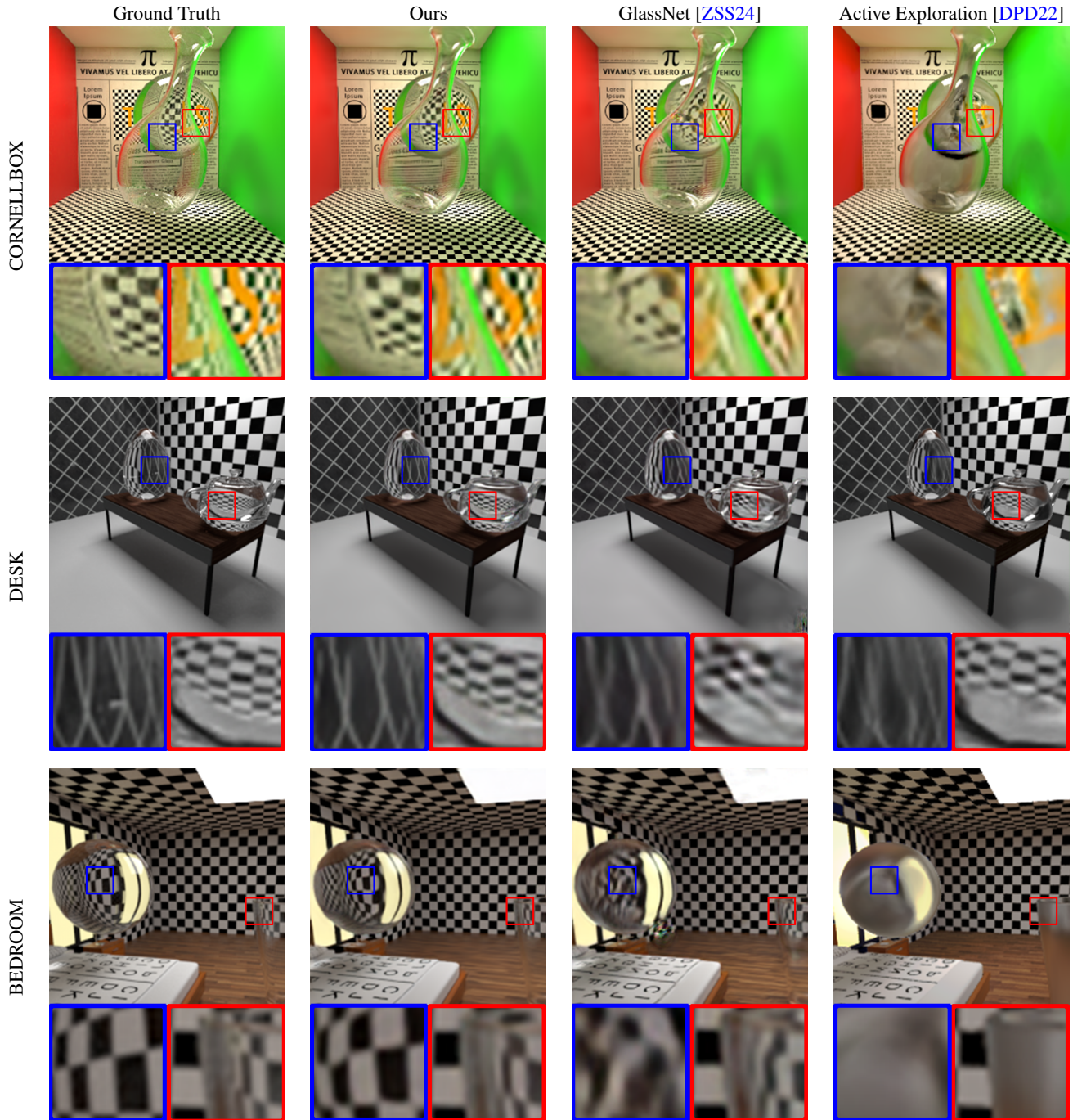


Figure 5: Qualitative comparison. We compare with GlassNet and Active Exploration [DPD22] in three challenging scenes. Our method achieves the best visual refraction rendering quality among all compared methods.

Scene	Method	Metrics						
		L1 ↓	LPIPS ↓	SSIM ↑	PSNR ↑	T.L1 ↓	T.PSNR ↑	T.Amp. ↓
CORNELLBOX	GlassNet	0.02988	0.04444	0.89189	23.85371	0.22757	13.52789	0.11564
	Active Exp.	0.07726	0.17964	0.75572	17.59048	0.391709	10.20211	0.16840
	Ours	0.02836	0.03381	0.90364	24.63194	0.21195	14.30030	0.11139
BEDROOM	GlassNet	0.03279	0.07125	0.88129	27.04549	0.18837	17.22410	0.08531
	Active Exp.	0.03721	0.13940	0.86013	24.30689	0.31550	12.35947	0.13601
	Ours	0.02709	0.05951	0.89872	28.60590	0.16426	18.00709	0.08389
DESK	GlassNet	0.03691	0.08818	0.88115	25.07125	0.29159	12.78175	0.23917
	Active Exp.	0.03515	0.07266	0.90637	21.61073	0.26436	14.85235	0.35330
	Ours	0.02330	0.05344	0.93148	27.69721	0.20789	15.25814	0.23184

Table 3: Quantitative comparison. Results show that our method, CrystalNet, has better performance on scenes with refraction objects compared to GlassNet and Active Exploration. Best result is denoted in **bold**.

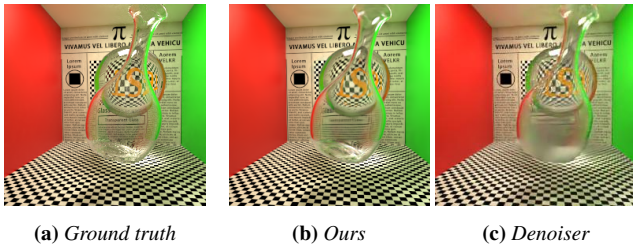


Figure 6: Comparison with a path tracing denoiser. The refraction rendering quality of real-time path tracing denoiser is unsatisfactory even with 32 samples per pixel.

Metrics	Variable Lighting		Variable IoR	
	Ours	GlassNet	Ours	GlassNet
L1 ↓	0.02432	0.03495	0.03784	0.03964
LPIPS ↓	0.02203	0.05687	0.05375	0.11011
T.L1 ↓	0.16413	0.25086	0.17037	0.21048
T.Amp. ↓	0.09240	0.11860	0.11688	0.14551

Table 4: Quantitative comparison under more scene parameters. Our experiments show that our method still performs strong under variable lighting and IoR. Best result is highlighted in **bold**.

color space loss, which also decreases the required training time as converting to during training is expensive.

We further demonstrate that our method is capable of handling refraction with roughness in Fig. 9. As shown in various previous works [XZX19, MST*21], neural networks tend to learn low-frequency information well, whereas ignores high-frequency information. Refraction on rough surface contains much lower frequency, and thus can be baked well in neural rendering models.

4.4. Performance Analysis

We implement our method into a custom CUDA/OpenGL rendering framework. Under this framework, our renderer is able to render

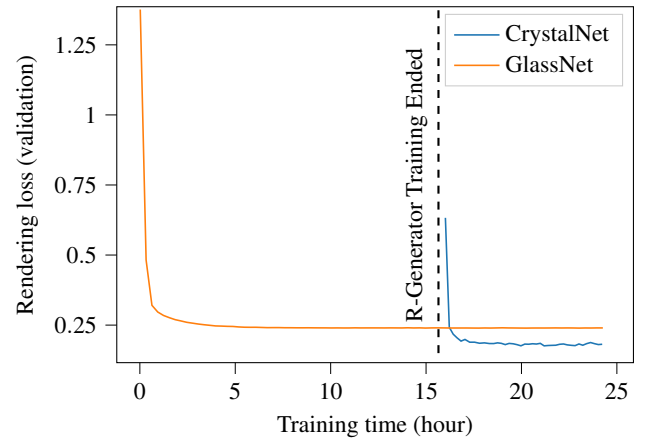


Figure 7: Evolution of the validation loss w.r.t. training time.

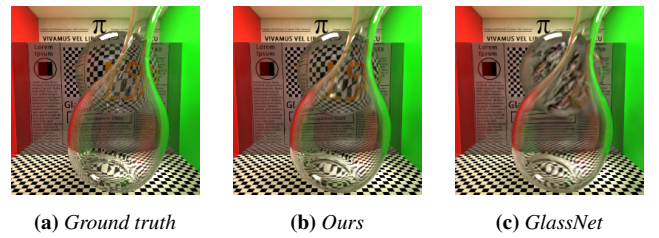


Figure 8: Qualitative results on three refractive objects with variable IoR. All three refractive objects have different IoR, and the IoR of the glass plane varies from 1.6 to 2.

256 × 256 images with two refraction objects at 55 FPS using a RTX 4090 GPU, achieving real-time performance. We also demonstrate the performance advantage to other methods. Under the same hardware condition, *Active Exploration* requires averagely 40 milliseconds frame time, whereas our method is significantly more efficient, requiring only averagely 18 milliseconds frame time. Furthermore, by utilizing a supersampler, CARN [AKS18], our method can render 1024 × 1024 with two refraction objects at 25 FPS.

	L1 ↓	SSIM ↑	T.L1 ↓	T.Amp. ↓
$\hat{\gamma} = \{\hat{\mathbf{n}}\}$	0.03220	0.88409	0.22862	0.12305
$\hat{\gamma} = \{\hat{c}\}$	0.02520	0.92736	0.17414	0.10046
$\hat{\gamma} = \{\hat{\mathbf{n}}, \hat{c}\}$	0.02439	0.93371	0.16471	0.09735

Table 5: Impact of compositions of the R-buffer. Results show including refracted normal, $\hat{\mathbf{n}}$, and refracted texture, \hat{c} , in R-buffers increases rendering quality. Best result is highlighted in **bold**.

L_1 Refraction Rendering Error	
L_1^{UV}	0.1649
$L_1^{UV} + L_1^{Tex}$	0.1657
$L_2^{UV} + L_1^{Tex}$	0.1710

Table 6: Impact of the texture reconstruction loss. Our experiments show the combination of texture reconstruction loss and UV space loss does not improve the performance of the network. Best result is highlighted in **bold**.

Additionally, as we are using the permutation-invariant structure as described in Eq. (5), the memory consumption does not grow with the number of transmissive objects because the transmissive objects are handled one by one.

5. Limitations and Discussion

Due to computation time scaling with resolution size, most neural rendering approaches are limited to lower resolutions. In this work, we have focused on improving the quality of the results, without significant improvements in computational speed. To increase the rendering resolution while maintaining similar performance, it is necessary to employ supersampling-based approaches. We believe that through optimization and breakthroughs in supersampling will allow our proposed approach to run in real-time at acceptable resolutions, allowing for general usage.

Our approach is able to accurately reconstruct high frequency textures through refraction thanks to the R-buffer, greatly improving

results over existing approaches. However, in one special situation where total reflection occurs, the approximation error may become significant due to R-buffer only capturing refraction rays. Although it is not possible to directly extend the R-buffer concept to handle total reflection, it should be possible to use a similar approach to approximate the reflective component through baking it into another neural network component.

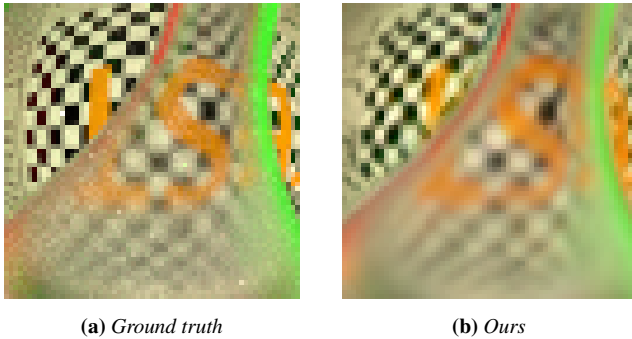


Figure 9: Refraction with roughness. Our method is able to capture the roughness effect.

References

- [AKS18] AHN N., KANG B., SOHN K.-A.: Fast, accurate, and lightweight super-resolution with cascading residual network. In *Proceedings of the European conference on computer vision (ECCV)* (2018), pp. 252–268. 8
- [Bit16] BITTERLI B.: Rendering resources, 2016. <https://benedikt-bitterli.me/resources/>. 6
- [BMRF*22] BEMANA M., MYSZKOWSKI K., REVAL FRISVAD J., SEIDEL H.-P., RITSCHER T.: Eikonal fields for refractive novel-view synthesis. In *ACM SIGGRAPH 2022 Conference Proceedings* (2022), pp. 1–9. 3
- [CKS*17] CHAITANYA C. R. A., KAPLAYAN A. S., SCHIED C., SALVI M., LEFOHN A., NOWROUZEZAHRAI D., AILA T.: Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12. 3, 6
- [CLZ*23] CHEN X., LIU J., ZHAO H., ZHOU G., ZHANG Y.-Q.: Nerf: 3d reconstruction and view synthesis for transparent and specular objects with neural refractive-reflective fields. *arXiv preprint arXiv:2309.13039* (2023). 3
- [DCS*24] DENG W., CAMPBELL D., SUN C., KANITKAR S., SHAFFER M., GOULD S.: Ray deformation networks for novel view synthesis of refractive objects. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2024), pp. 3118–3128. 3
- [DPD22] DIOLATZIS S., PHILIP J., DRETTAKIS G.: Active exploration for neural global illumination of variable scenes. *ACM Transactions on Graphics (TOG)* 41, 5 (2022), 1–18. 2, 6, 7
- [DRBS*11] DE ROUSIERS C., BOUSSEAU A., SUBR K., HOLZSCHUCH N., RAMAMOORTHY R.: Real-time rendering of rough refraction. *IEEE Transactions on Visualization and Computer Graphics* 18, 10 (2011), 1591–1602. 2
- [ED06] EISEMANN E., DÉCORET X.: Fast scene voxelization and applications. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games* (2006), pp. 71–78. 2
- [EJRB*18] ESLAMI S. A., JIMENEZ REZENDE D., BESSE F., VIOLA F., MORCOS A. S., GARNELO M., RUDERMAN A., RUSU A. A., DANIELKA I., GREGOR K., ET AL.: Neural scene representation and rendering. *Science* 360, 6394 (2018), 1204–1210. 2
- [GGH*22] GAO K., GAO Y., HE H., LU D., XU L., LI J.: Nerf: Neural radiance field in 3d vision, a comprehensive review. *arXiv preprint arXiv:2210.00379* (2022). 3
- [GLD06] GÉNEVAUX O., LARUE F., DISCHLER J.-M.: Interactive refraction on complex static geometry using spherical harmonics. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games* (2006), pp. 145–152. 2
- [GMX22] GAO D., MU H., XU K.: Neural global illumination: Interactive indirect illumination prediction under dynamic area lights. *IEEE Transactions on Visualization and Computer Graphics* (2022). 2
- [GP16] GUO J., PAN J.-G.: Real-time rendering of refracting transmissive objects with multi-scale rough surfaces. *The Visual Computer* 32 (2016), 1579–1592. 2
- [GRPN20] GRANSKOG J., ROUSSELLE F., PAPAS M., NOVÁK J.: Compositional neural scene representations for shading inference. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 135–1. 2, 6
- [GSHG98] GREGER G., SHIRLEY P., HUBBARD P. M., GREENBERG D. P.: The irradiance volume. *IEEE Computer Graphics and Applications* 18, 2 (1998), 32–43. 2
- [HDHN16] HEITZ E., DUPUY J., HILL S., NEUBELT D.: Real-time polygonal-light shading with linearly transformed cosines. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–8. 5
- [IZZE17] ISOLA P., ZHU J.-Y., ZHOU T., EFROS A. A.: Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 1125–1134. 5
- [JSRV22] JAKOB W., SPEIERER S., ROUSSEL N., VICINI D.: Dr.jit: A just-in-time compiler for differentiable rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)* 41, 4 (July 2022). [doi:10.1145/3528223.3530099](https://doi.org/10.1145/3528223.3530099). 5
- [Kaj86] KAJIYA J. T.: The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques* (1986), pp. 143–150. 2, 3
- [KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 6
- [LLW*23] LI Z., LONG X., WANG Y., CAO T., WANG W., LUO F., XIAO C.: Neto: neural reconstruction of transparent objects with self-occlusion aware refraction-tracing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 18547–18557. 3
- [LMCB06] LOZA A., MIHAYLOVA L., CANAGARAJAH N., BULL D.: Structural similarity-based object tracking in video sequences. In *2006 9th International Conference on Information Fusion* (2006), pp. 1–6. 6
- [MRNK21] MÜLLER T., ROUSSELLE F., NOVÁK J., KELLER A.: Real-time neural radiance caching for path tracing. *arXiv preprint arXiv:2106.12372* (2021). 3
- [MST*21] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHY R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM* 65, 1 (2021), 99–106. 5, 8
- [OB07] OLIVEIRA M. M., BRAUWERS M.: Real-time refraction through deformable objects. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games* (2007), pp. 89–96. 2
- [PSH*22] 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* (2022), pp. 1–4. 3
- [QSMG17] QI C. R., SU H., MO K., GUIBAS L. J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 652–660. 4
- [RFB15] RONNEBERGER O., FISCHER P., BROX T.: U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention—MICCAI 2015: 18th international conference, Munich, Germany, October 5–9, 2015, proceedings, part III 18* (2015), Springer, pp. 234–241. 3, 4
- [ROF92] RUDIN L. I., OSHER S., FATEMI E.: Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena* 60, 1–4 (1992), 259–268. 5
- [RWG*13] REN P., WANG J., GONG M., LIN S., TONG X., GUO B.: Global illumination with radiance regression functions. *ACM Trans. Graph.* 32, 4 (2013), 130–1. 2
- [Wym05a] WYMAN C.: An approximate image-space approach for interactive refraction. *ACM transactions on graphics (TOG)* 24, 3 (2005), 1050–1053. 2
- [Wym05b] WYMAN C.: Interactive image-space refraction of nearby geometry. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia* (2005), pp. 205–211. 2
- [XZX19] XU Z.-Q. J., ZHANG Y., XIAO Y.: Training behavior of deep neural network in frequency domain. In *Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part I 26* (2019), Springer, pp. 264–274. 8
- [ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 586–595. 6
- [ZSS24] ZHANG Z., SIMO-SERRA E.: Neural scene baking for permutation invariant transparency rendering with real-time global illumination. *arXiv preprint arXiv:2405.19056* (2024). 1, 2, 3, 4, 5, 6, 7