Neural Directional Encoding for Efficient and Accurate View-Dependent Appearance Modeling

Liwen Wu[1*] Sai Bi[2] Zexiang Xu[2] Fujun Luan[2] Kai Zhang[2]
Iliyan Georgiev[2] Kalyan Sunkavalli[2] Ravi Ramamoorthi[1]
[1]UC San Diego [2]Adobe Research

Abstract

Novel-view synthesis of specular objects like shiny metals or glossy paints remains a significant challenge. Not only the glossy appearance but also global illumination effects, including reflections of other objects in the environment, are critical components to faithfully reproduce a scene. In this paper, we present Neural Directional Encoding (NDE), a view-dependent appearance encoding of neural radiance fields (NeRF) for rendering specular objects. NDE transfers the concept of feature-grid-based spatial encoding to the angular domain, significantly improving the ability to model high-frequency angular signals. In contrast to previous methods that use encoding functions with only angular input, we additionally cone-trace spatial features to obtain a spatially varying directional encoding, which addresses the challenging interreflection effects. Extensive experiments on both synthetic and real datasets show that a NeRF model with NDE (1) outperforms the state of the art on view synthesis of specular objects, and (2) works with small networks to allow fast (real-time) inference. The source code is available at: https://github.com/lwwu2/nde

1. Introduction

Some of the most compelling appearances in our visual world arise from specular objects like metals, plastics, glossy paints, or silken cloth. Faithfully reproducing these effects from photographs for novel-view synthesis requires capturing both geometry and view-dependent appearance. Recent neural radiance field (NeRF) [37] methods have made impressive progress on efficient geometry representation and encoding using learnable spatial feature grids [5, 7, 29, 39, 45, 53]. However, modeling high-frequency view-dependent appearance has achieved much less attention. Efficient encoding of directional information is just as important, for modeling effects such as specular highlights and glossy interreflections. In this paper, we present a feature-grid-like *neural directional encoding* (NDE) that can accurately model the appearance of shiny objects.

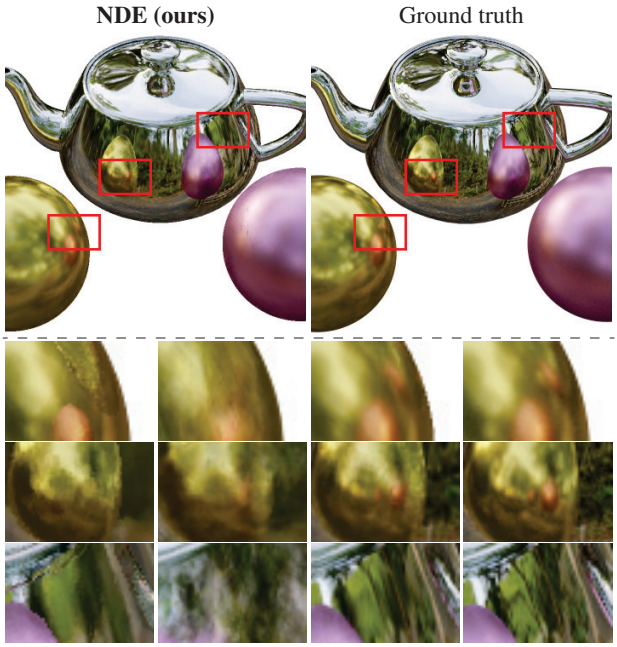

Figure 1. **Ours vs. analytical encoding.** Methods like Ref-NeRF [48] use an analytical function to encode viewing directions in large MLPs, failing to model complex reflections (column 1-2 of the insets). Instead, we encode view-dependent effects into feature grids with better interreflection parameterization, successfully reconstructing the details on the teapot and even multi-bounce reflections of the pink ball (3rd column of the insets) with little computational overhead (75 FPS on an NVIDIA 3090 GPU).

View-dependent colors in NeRFs (*e.g.* [48]) are commonly obtained by decoding spatial features and encoded direction. This approach necessitates a large multi-layer perceptron (MLP) and exhibits slow convergence with analytical directional encoding functions. To that end, we bring feature-grid-based encoding to the directional domain, representing reflections from distant sources via learnable feature vectors stored on a global environment map (Sec. 4.1). Features localize signal learning, reducing the MLP size required to model high-frequency far-field reflections.

Besides far-field reflections, spatially varying near-field

*This work was partially done during an internship at Adobe Research.

interreflections are also key effects in rendering glossy objects. These effects cannot be accurately modeled by NeRF's spatio-angular parameterization whose directional encoding does not depend on the position. In contrast, we propose a novel spatio-spatial parameterization by *cone-tracing a spatial feature grid* (Sec. 4.2) to encode near-field reflections. The cone tracing accumulates spatial encodings along the queried direction and position, thus it is spatially varying. While prior works consider only single-bounce or diffuse interreflections [26], our representation is able to model general multi-bounce reflection effects.

Overall, our neural directional encoding (NDE) achieves both high-quality modeling of view-dependent effects and fast evaluation. Figure 1 demonstrates NDE incorporated into NeRF, showing (1) accurate rendering of specular objects—a difficult challenge for the state of the art (Sec. 5.1), and (2) high inference speed that can be pushed to real-time without obvious quality loss (Sec. 5.2).

2. Related work

Novel-view synthesis aims to render a 3D scene from unseen views given a set of image captures with camera poses. Neural radiance fields (NeRF) [37] has recently emerged as a promising solution to this task, utilizing an implicit scene representation and volume rendering to synthesize photorealistic images. Follow-up works achieve state-of-the-art results in this area, for unbounded scenes [1, 59], in-the-wild captures [34], and sparse- or single-view reconstruction [6, 14, 28, 46, 47, 51]. While the original NeRF method [37] is computationally inefficient, it can be visualized in real-time by baking the reconstruction into voxel- [12, 15, 43, 57] or feature-grid-based representations (discussed below). The volumetric representation has been extended to work with signed distance fields (SDF) [50, 55] for better geometry acquisition, and the volume-rendering concept has also been applied to other 3D-related tasks such as object generation [4, 5, 27, 30, 42].

Feature-grid-based NeRF. NeRF's positional encoding [37] is a key component for the underlying multi-layer perceptron (MLP) network to learn high-frequency spatial and directional signals. However, the MLP size needs to be large, which leads to slow training and inference. Instead, methods like NSVF [29] and DVGO [45] interpolate a 3D volume of learnable feature vectors to encode the spatial signal, showing faster training and inference with even better spatial detail. Addressing the sparsity in typical scene geometry, later works avoid maintaining a large dense 3D grid via volume-compression techniques such as hash grids [39] and tensor factorization [5, 7, 11]. These methods are compact and scale up the feature grid to large scenes [2, 39] and even work with SDF-based models [25, 56]. The essence of feature-grid encoding is to interpolate feature vectors attached to geometry primitives, and similar ideas have also been applied to irregular 3D grids [22, 44], point clouds [19, 20, 54, 62], and meshes [8]. Operations like mip-mapping are trivial on feature grids, enabling efficient anti-aliasing and range query of NeRF models [2, 16, 53]—something we also leverage in this paper to encode rough reflection.

Rendering specular objects. Apart from geometry, view-dependent effects like reflections from rough surfaces are a crucial component in photorealistic novel-view synthesis. Reflections are conventionally modeled by fitting local light-field functions [10, 17, 36]. A 4D light field presents more degrees of freedom than the constraints from input images, which necessitates additional regularization to avoid overfitting. Inverse-rendering approaches introduce such a constraint by solving for parametric BRDFs and lighting, then using forward rendering to reconstruct the light field. Spherical-basis lighting [60] or split-sum approximation [31, 40] are usually used to tamper the Monte Carlo variance of specular-reflection derivatives [3]. ENVIDR [26] and NMF [33] further explicitly consider global-illumination effects by ray-tracing one or few bounces of indirect lighting. On the other hand, Ref-NeRF [48] uses an integrated directional encoding (IDE) to directly improve NeRF's view-dependent effects. IDE encodes the reflected direction rather than viewing direction to let the network learn an environment-map-like function and is pre-filtered to account for rough reflection effects. Our neural directional encoding, similar to IDE, can model general view-dependent appearance without assuming simplified lighting or reflections but with smaller computation cost.

3. Preliminaries

We assume opaque objects with diffuse and specular components and demonstrate our directional encoding using a surface-based model that represents a scene using a signed distance field (SDF) $s(\mathbf{x})$ and a color field $\mathbf{c}(\mathbf{x}, \boldsymbol{\omega})$ (dependent on the viewing direction $\boldsymbol{\omega}$). The SDF is converted to NeRF's density field σ following VolSDF [55] with a learnable parameter β controlling the boundary smoothness:

$$\sigma(\mathbf{x}) = \begin{cases} \frac{1}{2\beta} \exp\left(\frac{s(\mathbf{x})}{\beta}\right) & \text{if } s(\mathbf{x}) \leq 0, \\ \frac{1}{\beta}\left(1 - \frac{1}{2}\exp\left(-\frac{s(\mathbf{x})}{\beta}\right)\right) & \text{otherwise.} \end{cases} \tag{1}$$

The color $\mathbf{C}(\mathbf{x}, \boldsymbol{\omega})$ of a ray with origin $\mathbf{x}$ and direction $\boldsymbol{\omega}$ can thus be volume-rendered [35]:

$$\mathbf{C}(\mathbf{x}, \boldsymbol{\omega}) = \sum_i w(\sigma(\mathbf{x}_i))\mathbf{c}(\mathbf{x}_i, \boldsymbol{\omega}), \text{ where} \tag{2}$$

$$w(\sigma(\mathbf{x}_i)) = \left(1 - e^{-\sigma(\mathbf{x}_i)\delta_i}\right) \prod_{j<i} e^{-\sigma(\mathbf{x}_j)\delta_j}, \tag{3}$$

with $\delta_i = \|\mathbf{x}_i - \mathbf{x}_{i-1}\|_2$ and $\mathbf{x}_i$ denoting the i^{th} sample point along the ray. Like Ref-NeRF [48], we decompose the color $\mathbf{c}$ into a diffuse color $\mathbf{c}_d$, specular tint $\mathbf{k}_s$, and specular color $\mathbf{c}_s$ queried in reflected direction $\boldsymbol{\omega}_r$ with surface normal $\mathbf{n}$ given by the SDF gradient:

$$\begin{aligned} \mathbf{c}(\mathbf{x}, \boldsymbol{\omega}) &= \mathbf{c}_d(\mathbf{x}) + \mathbf{k}_s(\mathbf{x})\mathbf{c}_s(\mathbf{x}, \boldsymbol{\omega}_r), \quad \text{where} \\ \boldsymbol{\omega}_r &= \text{reflect}(\boldsymbol{\omega}, \mathbf{n}), \quad \mathbf{n} = \text{normalize}(\nabla_{\mathbf{x}} s(\mathbf{x})). \end{aligned} \tag{4}$$

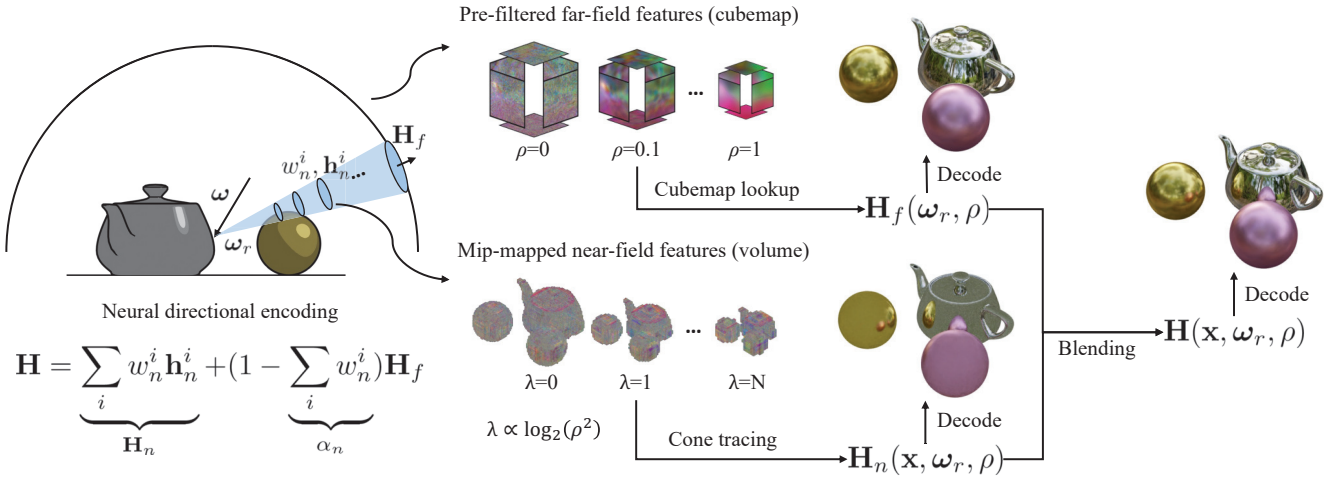

Figure 2. **Pipeline of our neural directional encoding (NDE).** We encode far-field reflections into a cubemap and near-field interreflections into a volume. Both representations store learnable feature vectors to encode direction and are mip-mapped to account for rough reflections. Given a reflected ray, the features are combined by tracing a cone of size proportional to the surface roughness to aggregate spatial features with cubemap features blended as the background. The result is fed into an MLP to output the specular color (Eq. (5)).

Here, the specular color $\mathbf{c}_s$ is decoded from an MLP that conditions on spatial feature $\mathbf{f}(\mathbf{x})$, directional encoding $\mathbf{H}$ controlled by surface roughness ρ, and the cosine term $\mathbf{n}\cdot\boldsymbol{\omega}$:

$$\mathbf{c}_s(\mathbf{x}, \boldsymbol{\omega}_r) = \text{MLP}(\mathbf{f}(\mathbf{x}), \mathbf{H}(\mathbf{x}, \boldsymbol{\omega}_r, \rho(\mathbf{x})), \mathbf{n}\cdot\boldsymbol{\omega}). \quad (5)$$

$\mathbf{c}_d, \mathbf{k}_s, \mathbf{f}, \rho$ come from a spatial MLP (Sec. 4.3).

Discussion on directional encoding. Previous works [37, 48] use an analytical function for $\mathbf{H}$ dependent only on $\boldsymbol{\omega}_r$ (and optionally ρ), which has several limitations: (1) the encoding function is fixed (not learnable), and (2) the spatial context only comes from $\mathbf{f}(\mathbf{x})$. Both require the decoder MLP to be large to fit the spatio-angular details of the specular color, which can be expensive and slow.

4. Neural directional encoding

To minimize the MLP complexity, we use a learnable neural directional encoding that also depends on the spatial location. Specifically, our NDE encodes different types of reflection by different representations, which include a cubemap feature grid $\mathbf{h}_f$ for far-field reflections and a spatial volume $\mathbf{h}_n$ that models near-field interreflections. As shown in Fig. 2, we compute $\mathbf{H}$ by first cone-tracing $\mathbf{h}_n$ accumulated along the reflected ray, yielding near-field feature $\mathbf{H}_n$ (Sec. 4.2), and blending the far-field feature $\mathbf{H}_f$ queried from $\mathbf{h}_f$ in the same direction (Sec. 4.1):

$$\mathbf{H}(\mathbf{x}, \boldsymbol{\omega}_r, \rho) = \mathbf{H}_n(\mathbf{x}, \boldsymbol{\omega}_r, \rho) + (1 - \alpha_n)\mathbf{H}_f(\boldsymbol{\omega}_r, \rho), \quad (6)$$

where α_n is the cone-traced opacity [24], and both features are mip-mapped with ρ deciding the mip level.

4.1. Far-field features

Feature-grid-based representations [7, 29, 39, 45, 53] speed-up spatial signal learning by storing feature vectors in voxels for local signal control. Similarly, we place feature vectors $\mathbf{h}_f$ at every pixel of a global cubemap to encode ideal specular reflections. The cubemap is pre-filtered to model reflections under rough surfaces in the split-sum [18] style, where the k^{th} level mip-map $\mathbf{h}_f^k$ is created by convolving the downsampled $\mathbf{h}_f$ using a GGX kernel [49] D with canonical roughness ρ_k evenly spaced in $[0, 1]$:

$$\mathbf{h}_f^k = \text{convolution}(\text{downsample}(\mathbf{h}_f, k), D(\rho_k)). \quad (7)$$

Given the surface roughness, we perform a cubemap lookup in the reflected direction and interpolate between mip levels to get the far-field feature:

$$\mathbf{H}_f(\boldsymbol{\omega}_r, \rho) = \text{lerp}\left(\mathbf{h}_f^k(\boldsymbol{\omega}_r), \mathbf{h}_f^{k+1}(\boldsymbol{\omega}_r), \frac{\rho - \rho_k}{\rho_{k+1} - \rho_k}\right), \quad (8)$$

where $\text{lerp}(\cdot)$ denotes linear interpolation and $\rho \in [\rho_k, \rho_{k+1}]$.

The cubemap-based encoding allows signals in different directions to be optimized independently by tuning the feature vectors. This is easier to optimize than globally solving the MLP parameters, making it more suitable to model high-frequency details in the angular domain (Fig. 3). The coarse level feature is a consistently filtered version of the fine level, which is empirically found to be better constrained than using independent feature vectors at each mip level [23, 58].

4.2. Near-field features

Parameterizing the specular color by a spatial and angular feature is sufficient for distant reflections, but lacks expressivity for near-field interreflections: different points query

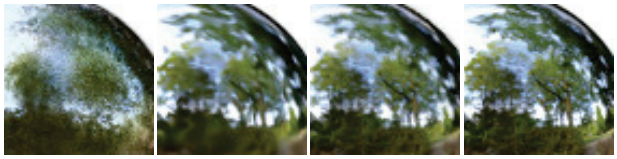

Figure 3. **Our cubemap-based feature encoding** requires only a small MLP (2 layers, 64 width) to model details in mirror reflections (3rd image) comparable with IDE [48] (2nd image; 8 layers, 256 width MLP) that fails when the MLP is small (1st image).

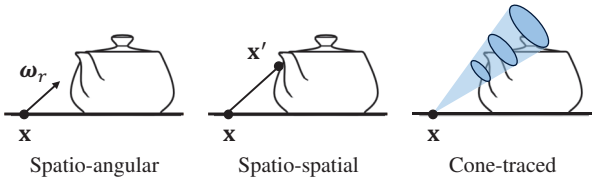

Figure 4. **Spatio-spatial encoding** (middle) is equivalent to the common spatio-angular encoding (left) of mirror reflections, but it captures the variation of $\mathbf{x}'$ across different $\mathbf{x}$. The idea can be extended to model rough reflections by cone tracing mip-mapped spatial features covered by the reflection cone (right).

the same $\mathbf{h}_f$, so spatially varying components can end up being averaged out during optimization. Our insight is that the spatio-angular reflection can also be parameterized as a spatio-spatial function of current and next bounce location (Fig. 4). Therefore, an MLP can decode the second bounce spatial feature with $\mathbf{f}(\mathbf{x})$ in Eq. (5) to get mirror reflections.

For rough reflections, we aggregate the averaged second bounce feature under the reflection lobe by cone tracing [9] (Fig. 4, right), which volume renders the mip-mapped spatial features $\mathbf{h}_n$ using the mip-mapped density σ_n along the reflected ray $\mathbf{x}+\boldsymbol{\omega}_r t$ with mip level $\lambda_i = \log_2(2r_i)$ at sample point $\mathbf{x}'_i$ decided by the cone's footprint $r_i = \sqrt{3}\rho^2\|\mathbf{x}-\mathbf{x}'_i\|_2$:

$$\mathbf{H}_n(\mathbf{x},\boldsymbol{\omega}_r,\rho) = \sum_i w_n^i \mathbf{h}_n^i, \quad \text{where}$$
$$w_n^i = w(\sigma_n(\mathbf{x}'_i,\lambda_i)), \quad \mathbf{h}_n^i = \mathbf{h}_n(\mathbf{x}'_i,\lambda_i). \tag{9}$$

The cone's footprint is selected to cover the GGX lobe at $\mathbf{x}$ (see supplemental document). Note that we do not use the SDF-converted σ in Eq. (1) as it cannot be mip-mapped; instead, we optimize a separate σ_n to match σ (Sec. 4.3) jointly with the indirect feature $\mathbf{h}_n$. Both are decoded from a tri-plane [5] $\mathbf{T}_n$, whose each 2D plane is mip-mapped similar to Tri-MipRF [16]:

$$\sigma_n(\mathbf{x}'_i,\lambda_i), \mathbf{h}_n(\mathbf{x}'_i,\lambda_i) = \text{MLP}(\text{mipmap}(\mathbf{T}_n(\mathbf{x}'_i),\lambda_i)). \tag{10}$$

The indirect rays are spatially varying, hence the cone-traced near-field features are spatially varying too. This has advantages over the angular-only feature for learning inter-reflections and is empirically less likely to overfit (Fig. 5). This is because the same $\mathbf{h}_n$ is traced from different rays in training, such that the underlying representation is well-constrained. $\mathbf{H}_n$ and $\mathbf{H}_f$ are similar to the foreground and background colors in regular volume rendering, so $\mathbf{H}_f$ can be naturally composited with $\mathbf{H}_n$ using the opacity $\alpha_n = 1 - \prod_i e^{-\sigma_n(\mathbf{x}'_i,\lambda_i)\delta_i} = \sum_i w_n^i$ as in Eq. (6).

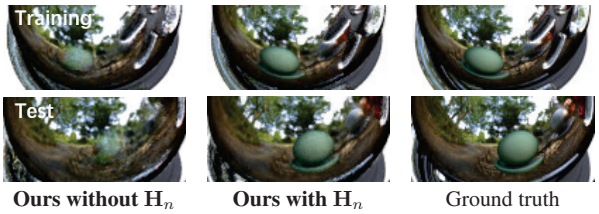

Figure 5. **Our cone-traced near-field features** successfully reconstruct the reflected spheres (2nd column) under novel views, which are overfitted by the angular-only encoding (1st column).

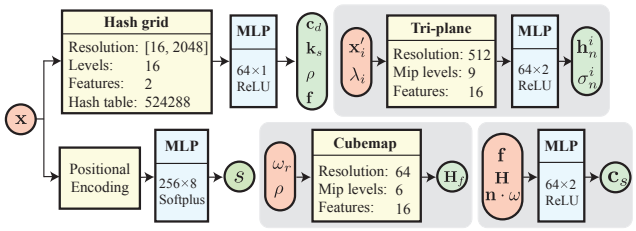

Figure 6. **Network architectures.** $N \times M$ denotes an M-layer MLP of width N.

4.3. Optimization

Figure 6 shows our network architectures. Stable geometry optimization is essential for modeling specular objects, so we use the positional-encoded MLP from VolSDF [55] to output the SDF. To reduce computation cost, a hash grid is used to encode other spatial features $(\mathbf{c}_d, \mathbf{k}_s, \rho, \mathbf{f})$, and all other MLPs are tiny. The representation is optimized through the Charbonnier loss [1] between ground truth pixel color $\mathbf{C}_{\text{gt}}$ and our rendering $\mathbf{C}$ in tone-mapped space:

$$L = \sum_{\mathbf{x},\boldsymbol{\omega}} \sqrt{\|\Gamma(\mathbf{C}(\mathbf{x},\boldsymbol{\omega})) - \mathbf{C}_{\text{gt}}(\mathbf{x},\boldsymbol{\omega})\|_2^2 + 0.001}, \tag{11}$$

where Γ is the tone-mapping function [40].

Occupancy-grid sampling. Eqs. (3) and (9) are accelerated by an occupancy-grid estimator [24] to get rid of computations in empty space. This is especially important for the efficient near-field feature evaluation, since we trace a reflected ray for each primary ray sample. The primal ray rendering uses a fixed ray marching step of 0.005. Following [9], we choose the cone tracing step proportional to its footprint: $\max(0.5r_i, 0.005)$, and query a mip-mapped occupancy grid for the correct occupancy information.

Regularization. Given the primary samples $\mathbf{x}_i$, Eikonal loss [55] L_{eik} is applied to regularize the SDF, and we implicitly regularize σ_n to match σ by encouraging the rendering using σ_n at mip level 0 to be close to the ground truth:

$$L_\sigma = \sum_{\mathbf{x},\boldsymbol{\omega}} \|\mathbf{C}_\sigma(\mathbf{x},\boldsymbol{\omega}) - \mathbf{C}_{\text{gt}}(\mathbf{x},\boldsymbol{\omega})\|_2^2, \quad \text{where}$$
$$\mathbf{C}_\sigma(\mathbf{x},\boldsymbol{\omega}) = \sum_i w(\sigma_n(\mathbf{x}_i,0))\mathring{\mathbf{c}}(\mathbf{x}_i,\boldsymbol{\omega}), \tag{12}$$

$\mathring{\square}$ denotes stop-gradient to prevent σ_n affecting appearance. The total loss is $L + 0.1L_{\text{eik}} + 0.01L_\sigma$.

Implementation details. We implement our code using PyTorch [41], NerfAcc [24], and CUDA. The optimization takes 400k steps using the Adam optimizer [21] with 0.0005 learning rate and dynamic batch size [39] targeting for 32k primary point samples. We use the scheduler from BakedSDF [15] to anneal β in Eq. (1) for more stable convergence. Because the SDF uses a positional-encoded MLP, each scene still requires 10∼18 hours to train on an NVIDIA 3090 GPU with 15GB GPU memory usage.

5. Experiments

We evaluate our method on view synthesis of specular objects using synthetic and real scenes. The synthetic scenes include the Shinny Blender dataset [48] and the Materials scene from the NeRF Synthetic dataset [37], all rendered without background; the real scenes come from NeRO [31] which contain backgrounds and reflections of the capturer in the images. The rendering quality is compared in terms of PSNR, SSIM [52], LPIPS [61], and the inference speed in FPS is recorded on an NVIDIA 3090 GPU.

Background and capturer. For real scenes, we use a separate Instant-NGP [39] with coordinate contraction [1] to render backgrounds. Similarly to NeRO [31], the reflection of the capturer is encoded by blending a capturer plane feature $\mathbf{h}_c$ of opacity α_c between $\mathbf{H}_f$ and $\mathbf{H}_n$:

$$\mathbf{H} = \mathbf{H}_n + (1-\alpha_n)(\alpha_c\mathbf{h}_c + (1-\alpha_c)\mathbf{H}_f), \text{ where}$$
$$\alpha_c, \mathbf{h}_c = \text{MLP}(\text{mipmap}(\mathbf{T}_c(\mathbf{u}), \lambda_c)) \tag{13}$$

are decoded from a mip-mapped 2D feature grid $\mathbf{T}_c$; $\mathbf{u}, \lambda_c$ are the ray-plane intersection coordinate and the mip-level derived from the intersection footprint. Jointly optimizing foreground and background networks can be unstable, so we apply stabilization loss from NeRO [31] and modify the specular color computation for the first 200k steps: $\mathbf{h}_f, \mathbf{h}_n, \mathbf{h}_c$ are sampled and decoded into colors first, then the colors are blended to get $\mathbf{c}_s$. Compared to blending the feature and decoding, we find the decoding-then-blending strategy provides better geometry optimization.

5.1. View synthesis

We compare against NeRO [31], ENVIDR [26], and Ref-NeRF [48] on synthetic scenes. All methods except for Ref-NeRF use SDFs, and we evaluate NeRO after the BRDF estimation as it shows better performance. Ideally, both backgrounds and reflections from the capturer should be removed when evaluating renderings of specular objects, which is difficult for the real scenes. Therefore, we only qualitatively compare real scenes against NeRO with PSNR computed on the foreground zoom-ins without the capturer.

Method	Mat.	Teapot	Toaster	Car	Ball	Coffee	Helmet	Mean
PSNR ↑								
NeRO	24.85	40.29	*27.31*	26.98	31.50	33.76	29.59	30.61
ENVIDR	29.51	46.14	26.63	29.88	41.03	*34.45*	*36.98*	34.95
Ref-NeRF	**35.41**	*47.90*	25.70	**30.82**	**47.46**	34.21	29.68	*35.88*
NDE (ours)	*31.53*	**49.12**	**30.32**	*30.39*	*44.66*	**36.57**	**37.77**	**37.19**
SSIM ↑								
NeRO	0.878	0.993	0.891	0.926	0.953	0.960	0.953	0.936
ENVIDR	0.971	**0.999**	*0.955*	**0.972**	**0.997**	**0.984**	**0.993**	**0.982**
Ref-NeRF	**0.983**	0.998	0.922	0.955	*0.995*	0.974	0.958	*0.969*
NDE (ours)	*0.972*	**0.999**	**0.968**	*0.968*	*0.995*	*0.979*	*0.990*	**0.982**
LPIPS ↓								
NeRO	0.138	0.017	0.162	0.064	0.179	0.099	0.102	0.109
ENVIDR	0.026	*0.003*	0.097	*0.031*	**0.020**	*0.044*	*0.022*	*0.035*
Ref-NeRF	*0.022*	0.004	*0.095*	0.041	0.059	0.078	0.075	0.053
NDE (ours)	**0.017**	**0.002**	**0.039**	**0.024**	*0.022*	**0.033**	**0.014**	**0.022**

Table 1. **Quantitative comparison on synthetic scenes** showing our encoding (NDE) is either the **best** or *second best* compared to other methods for view synthesis of specular objects.

Results. Overall, our method gives the best rendering quality on synthetic scenes with quantitative results either better or comparable with the baselines (Tab. 1). This is because our NDE gives the most detailed modeling of both far-field reflections and interreflections, which also helps improve the geometry reconstruction (Fig. 7 bottom). While ENVIDR's SSIM is slightly better than ours in several scenes, we not only achieve much better PSNRs (surpassing 2dB), but also higher LPIPS scores. The PSNR on the Materials (Mat.) scene is worse than Ref-NeRF's because the SDF is inefficient at modeling the concave geometry of the sphere base. However, our directional MLP is much smaller (Sec. 5.2), and we still achieve perceptually better appearance as shown in the insets of Fig. 7. The qualitative comparison in Fig. 8 shows that NDE extends well to real scenes, producing clearer specular reflections of the complex real-world environments compared to NeRO.

Editability. The near- and far-field features provide a natural separation of different reflections, allowing us to render these effects separately by excluding $\mathbf{H}_f$ or $\mathbf{H}_n$ during inference (Fig. 9). Because interreflections are spatially encoded in the near-field feature grid, an object and its first-bounce reflections can be removed by masking out both σ and σ_n from the corresponding regions (Fig. 10). This does not work for multi-bounce reflections which are not encoded on the deleted object.

5.2. Performance comparison

We compare the evaluation frames per second (FPS) on an 800×800 resolution of the color network and its MLP size (#Params.) with all baselines in Sec. 5.1 on synthetic scenes. The color MLPs include the decoder of $\sigma_n, \mathbf{h}_n, \mathbf{c}_s$ for our model (Fig. 6), lighting MLPs for NeRO [31] and

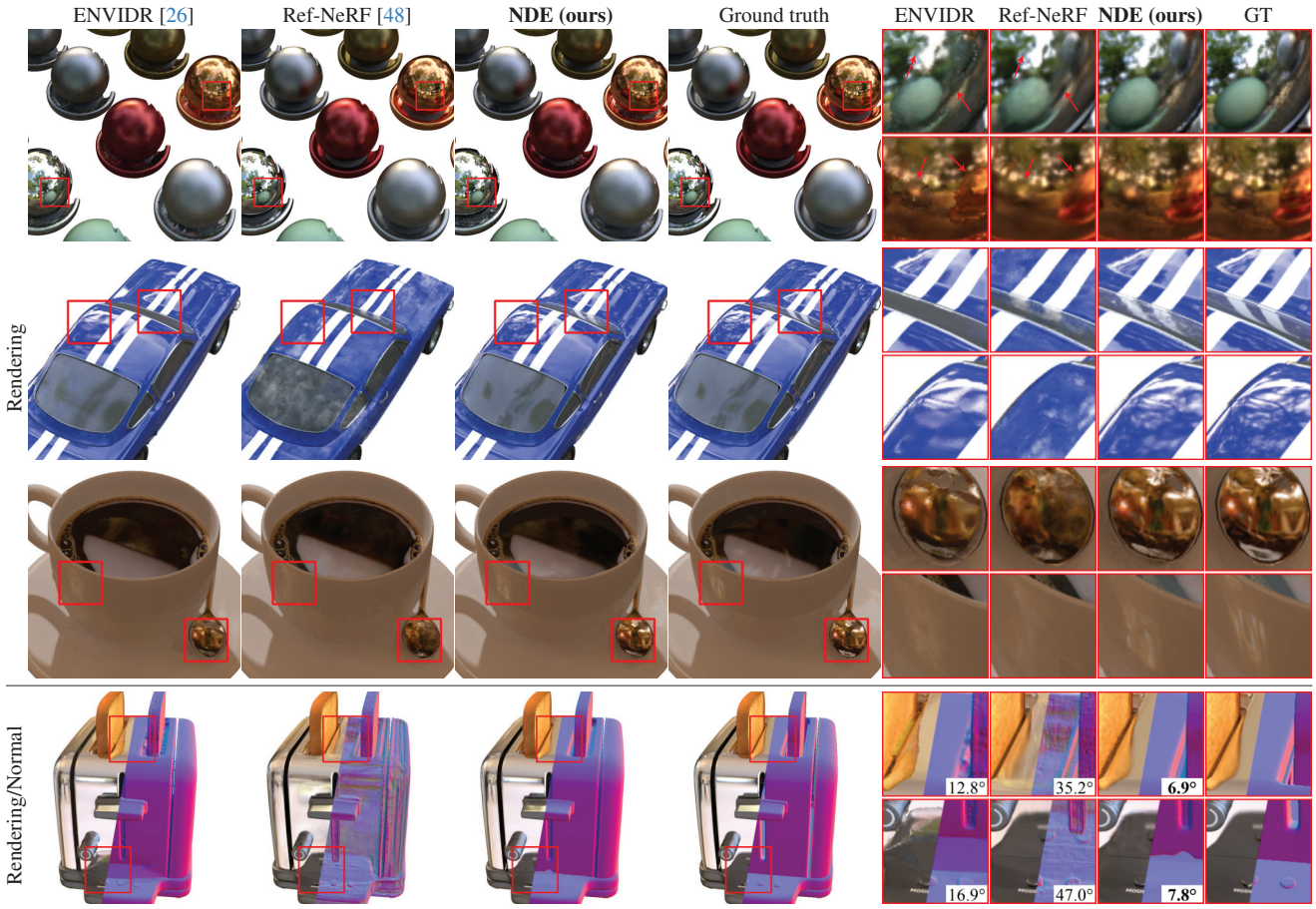

Figure 7. **Qualitative results for synthetic scenes** show our NDE successfully models the fine details of reflections from both environment lights (mirror sphere and car top) and other objects (glossy interreflections on spheres; zoom in to see the difference). Ref-NeRF tends to use wrong geometry to fake interreflections (2nd column on bottom). In contrast, our encoding has sufficient capacity to model interreflections, which enables more accurate normals (3rd column on bottom). Mean angular error of the normal is shown in the insets.

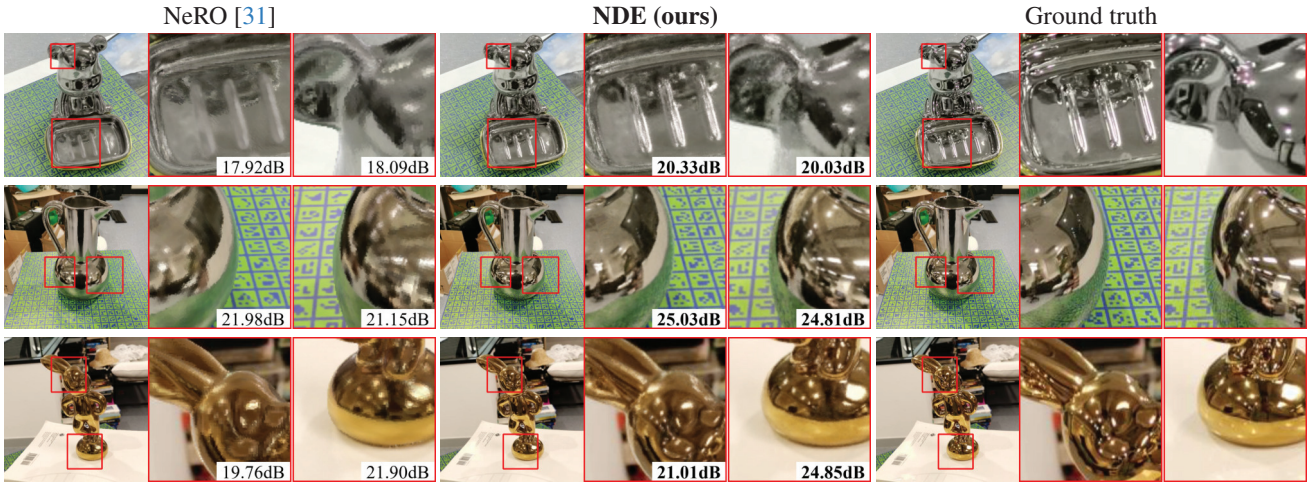

Figure 8. **Qualitative comparison on real scenes.** Our NDE gives better reconstruction of the interreflections (the bear's plate and bottom of the vase) and detailed highlights from the environment. Numbers in the insets are image PSNR values.

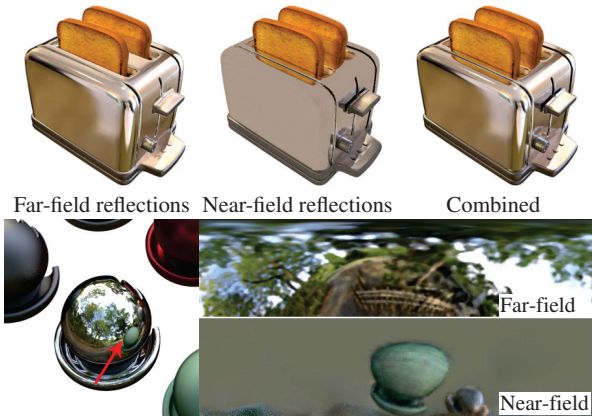

Figure 9. **Reflection separation.** We can visualize different reflection effects by feeding corresponding features into the network.

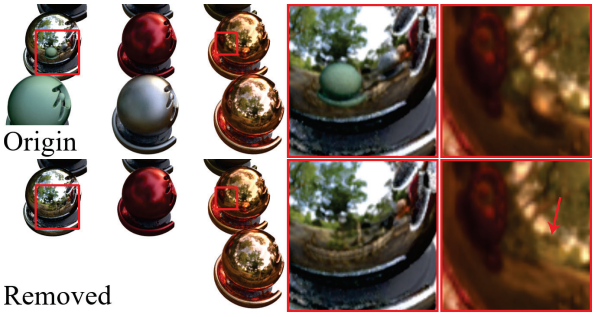

Figure 10. **Editability of our encoding.** Reflections from the deleted spheres can be removed by deleting the volume of their indirect features (bottom).

ENVIDR [26], and the directional MLP for Ref-NeRF [48]. The spatial-network evaluation is excluded to eliminate the difference caused by different geometry representations, network architectures, and sampling strategies. For each method, we choose the rendering batch size that maximizes its performance.

Results. As shown in the top half of Tab. 2, our NDE takes a fraction of a second to evaluate, because it requires substantially smaller MLPs to infer color without hurting the rendering. In contrast, other baselines need large MLPs to maintain rendering quality, which prevents them to be visualized in real-time.

Real-time application. It is possible to create a real-time version of our model by converting the SDF into a mesh through marching cubes [32] and baking $\mathbf{c}_d, \mathbf{k}_s, \rho, \mathbf{f}$ into mesh vertices. The pixel color then can be computed using the rasterized vertex attributes and $\mathbf{c}_s$ decoded from the NDE, which takes only a single cubemap lookup and cone tracing for each pixel. As a result, this process requires about the same budget as evaluating a real-time NeRF model [39, 45, 53]. We implement our real-time model (NDE-RT) in WebGL and report the full rendering frame rate (not just color evaluation) at the bottom of Tab. 2 with a real-time baseline 3DGS [19]. 3DGS is faster as it uses spherical harmonics for color without network evaluation,

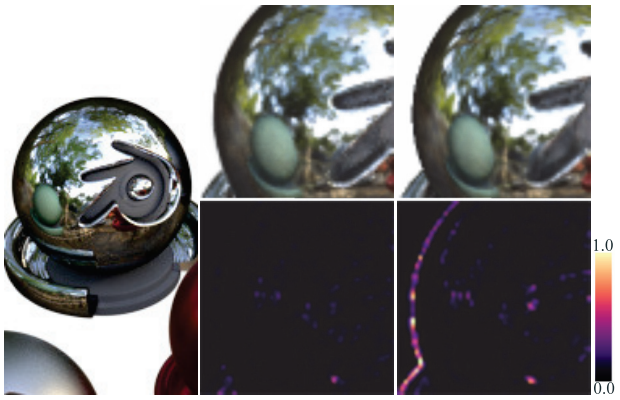

Figure 11. **Error near object boundaries in our real-time model** is caused by the marching-cube extraction of a triangle mesh and its subsequent rasterization (squared error maps at the bottom). This error does not lead to significant qualitative differences (top).

Method	FPS↑	#Params↓	PSNR↑	SSIM↑	LPIPS↓
NeRO	0.11	454k	30.61	0.936	0.109
ENVIDR	0.55	*206k*	34.95	**0.982**	0.035
Ref-NeRF	0.08	521k	*35.88*	0.969	0.053
NDE (ours)	3.03	**75k**	**37.19**	**0.982**	**0.022**
3DGS	**235**	-	30.30	0.949	0.076
NDE-RT (ours)	*66*	**75k**	35.48	*0.976*	*0.027*

Table 2. **Performance comparison.** Our NDE achieves high rendering quality, and its use of small MLPs enables fast color evaluation and real-time rendering. We report only the evaluation time and parameter counts of color MLPs except for 3DGS (no color MLPs) and our NDE-RT, for which we report the total rendering time. All metrics are averaged over the synthetic scenes in Tab. 1.

which leads to poor specular appearance reconstruction. Instead, our NDE-RT shows rendering quality comparable to other baselines while achieving frame rates above 60. The loss in PSNR is mainly due to error around object edges which is cause by the marching-cube mesh extraction and subsequent rasterization (Fig. 11). This error does not significantly affect the visual quality and can be resolved by fine-tuning the mesh [8, 40].

5.3. Ablation study

Different directional encodings. In Fig. 12 we compare different directional encodings on the Materials scene. IDE [48] (analytical) with our tiny MLP yields blurry reflections. Interreflections cannot be reconstructed using only the far-field feature, and if we volume-render rather than cone-trace the near-field feature, mirror interreflections can be recovered but reflections on rough surfaces look too sharp. It is therefore necessary to use both the cubemap-based far-field feature and the cone-traced near-field feature to get the best specular appearance (Tab. 3).

Network architecture. Table 4 shows the performance trade-off between different network architectures of our model on synthetic scenes. Using a smaller MLP width for

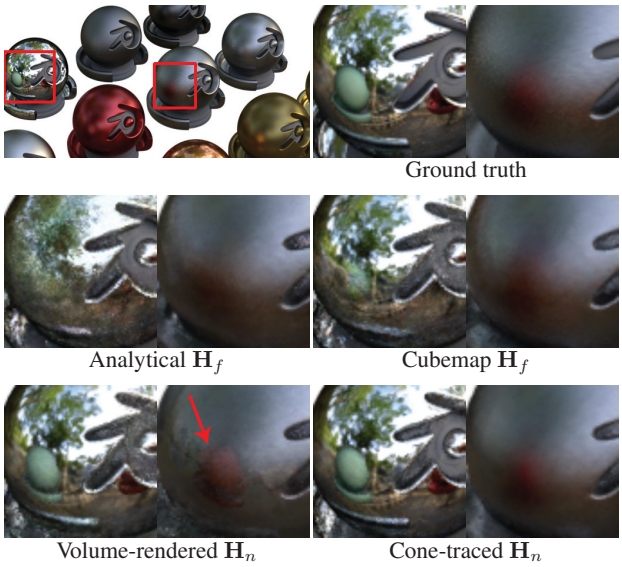

Figure 12. **Qualitative ablation of NDE components.** Details from the environment light fail to be reconstructed with an analytical encoding (mirror sphere on 2nd row). It is also necessary to use the cone-traced near-field feature, otherwise rough surfaces are rendered incorrectly (grey sphere on 3rd row).

Far-field feature	Near-field feature	PSNR↑	SSIM↑	LPIPS↓
Analytical	-	28.54	0.944	0.029
Cubemap	-	*30.27*	*0.962*	*0.022*
Cubemap	Volume-rendered	29.31	0.951	0.034
Cubemap	Cone-traced	**31.53**	**0.972**	**0.017**

Table 3. **Ablation on directional encodings** shows each component of NDE is needed for the best rendering quality. The comparison is made on the Materials scene.

Model	MLP width	PSNR↑	SSIM↑	LPIPS↓	FPS↑
Our offline	64	**37.19**	**0.982**	**0.022**	<1
	32	*36.69*	*0.979*	*0.026*	<1
	16	36.23	0.977	0.028	<1
Our real-time	64	35.48	0.976	0.027	66
	32	33.97	0.971	0.034	*211*
	16	33.71	0.969	0.036	**331**

Table 4. **Ablation on our network architecture.** Using a smaller MLP width introduces a minor loss in rendering fidelity but a noticeable real-time performance boost.

the decoder of $\sigma_n, \mathbf{h}_n, \mathbf{c}_s$ has only a slight negative impact on the rendering quality but significantly improves real-time performance. The rendering quality reduction of the real-time model is mainly caused by the error near object edges as discussed in Sec. 5.2.

Spatial mip-mapping strategies. Besides mip-mapped tri-plane [5, 16], our architecture can also work with a mip-mapped hash grid [39] for the near-field feature encoding. Similar to [2, 25], the hash-grid mip-mapping is implemented by gradually masking out fine-resolution features as the mip level increases. This results in limited model capacity for rough surfaces where most of the features are masked

	Mat.	Teapot	Toaster	Car	Ball	Coffee	Helmet	Mean
PSNR ↑								
Hash grid	30.89	49.00	29.46	30.16	43.48	34.98	37.67	36.52
Tri-plane	**31.53**	**49.12**	**30.32**	**30.39**	**44.66**	**36.57**	**37.77**	**37.19**
SSIM ↑								
Hash grid	0.968	**0.999**	0.953	0.967	0.990	0.974	**0.990**	0.977
Tri-plane	**0.972**	**0.999**	**0.968**	**0.968**	**0.995**	**0.979**	**0.990**	**0.982**
LPIPS ↓								
Hash grid	0.019	**0.002**	0.058	0.025	0.031	0.043	**0.014**	0.027
Tri-plane	**0.017**	**0.002**	**0.039**	**0.024**	**0.022**	**0.033**	**0.014**	**0.022**

Table 5. **Ablation on mip-mapping strategies** suggests that the mip-mapped tri-plane represents averaged near-field features and density better than the mip-mapped hash grid.

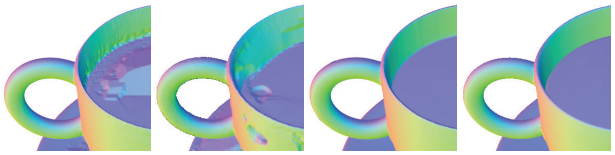

Figure 13. **Unstable geometry optimization of specular objects** prevents us from encoding the SDF using a hash grid [39] as it gives incorrect surface normals (middle left). This is also the case for other hash-grid-based methods (left).

out, such that a mip-mapped hash grid produces slightly worse rendering than the tri-plane encoding (Tab. 5).

Limitations. Like previous works [26, 31, 48], NDE is sensitive to the quality of the surface normal. This prevents us from using more efficient geometry representations such as a hash grid, which tends to produce corrupted geometry (Fig. 13). As a result, we use positional-encoded MLPs to model the SDF, which leads to long training times and is difficult for modeling transparent objects. Meanwhile, the editibility of our method is limited.

6. Conclusion

We have adapted feature-based NeRF encodings to the directional domain and introduced a novel spatio-spatial parameterization of view-dependent appearance. These improvements allow for efficient modeling of complex reflections for novel-view synthesis and could benefit other applications that model spatially varying directional signals, such as neural materials [13, 23, 58] and radiance caching [38].

Acknowledgements. This work was supported in part by NSF grants 2110409, 2100237, 2120019, ONR grant N00014-23-1-2526, gifts from Adobe, Google, Qualcomm, Rembrand, a Sony Research Award, as well as the Ronald L. Graham Chair and the UC San Diego Center for Visual Computing. Additionally, we thank Jingshen Zhu for insightful discussions.

References

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

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *ICCV*, 2023. 2, 8

[3] Yash Belhe, Bing Xu, Sai Praveen Bangaru, Ravi Ramamoorthi, and Tzu-Mao Li. Importance sampling brdf derivatives. In *ACM TOG*, 2024. 2

[4] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. Pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *CVPR*, 2021. 2

[5] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*, 2022. 1, 2, 4, 8

[6] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, 2021. 2

[7] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022. 1, 2, 3

[8] Zhiqin Chen, Thomas Funkhouser, Peter Hedman, and Andrea Tagliasacchi. Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In *CVPR*, 2023. 2, 7

[9] Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. Interactive indirect illumination using voxel cone tracing. In *Computer Graphics Forum*, 2011. 4

[10] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *CVPR*, 2019. 2

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

[12] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *ICCV*, 2021. 2

[13] Alban Gauthier, Robin Faury, Jérémy Levallois, Théo Thonat, Jean-Marc Thiery, and Tamy Boubekeur. Mipnet: Neural normal-to-anisotropic-roughness mip mapping. In *ACM TOG*, 2022. 8

[14] Jiatao Gu, Alex Trevithick, Kai-En Lin, Joshua M Susskind, Christian Theobalt, Lingjie Liu, and Ravi Ramamoorthi. Nerfdiff: Single-image view synthesis with nerf-guided distillation from 3d-aware diffusion. In *ICML*, 2023. 2

[15] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *ICCV*, 2021. 2, 5

[16] Wenbo Hu, Yuling Wang, Lin Ma, Bangbang Yang, Lin Gao, Xiao Liu, and Yuewen Ma. Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields. In *ICCV*, 2023. 2, 4, 8

[17] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. In *ACM TOG*, 2016. 2

[18] Brian Karis. Real shading in unreal engine 4. In *SIGGRAPH 2013 Course: Physically Based Shading in Theory and Practice*, 2013. 3

[19] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. In *ACM TOG*, 2023. 2, 7

[20] Leonid Keselman and Martial Hebert. Flexible techniques for differentiable rendering with 3d gaussians. *arXiv preprint arXiv:2308.14737*, 2023. 2

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[22] Jonas Kulhanek and Torsten Sattler. Tetra-nerf: Representing neural radiance fields using tetrahedra. In *ICCV*, 2023. 2

[23] Alexandr Kuznetsov, Krishna Mullia, Zexiang Xu, Miloš Hašan, and Ravi Ramamoorthi. Neumip: Multi-resolution neural materials. In *ACM TOG*, 2021. 3, 8

[24] Ruilong Li, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: A general nerf acceleration toolbox. *arXiv preprint arXiv:2210.04847*, 2022. 3, 4, 5

[25] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *CVPR*, 2023. 2, 8

[26] Ruofan Liang, Hui-Hsia Chen, Chunlin Li, Fan Chen, Selvakumar Panneer, and Nandita Vijaykumar. Envidr: Implicit differentiable renderer with neural environment lighting. In *ICCV*, 2023. 1, 2, 5, 6, 7, 8

[27] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *CVPR*, 2023. 2

[28] Kai-En Lin, Yen-Chen Lin, Wei-Sheng Lai, Tsung-Yi Lin, Yi-Chang Shih, and Ravi Ramamoorthi. Vision transformer for nerf-based view synthesis from a single input image. In *WACV*, 2023. 2

[29] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020. 1, 2, 3

[30] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Zexiang Xu, Hao Su, et al. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. In *NeurIPS*, 2023. 2

[31] Yuan Liu, Peng Wang, Cheng Lin, Xiaoxiao Long, Jiepeng Wang, Lingjie Liu, Taku Komura, and Wenping Wang. Nero: Neural geometry and brdf reconstruction of reflective objects from multiview images. In *ACM TOG*, 2023. 2, 5, 6, 8

[32] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, 1987. 7

[33] Alexander Mai, Dor Verbin, Falko Kuester, and Sara Fridovich-Keil. Neural microfacet fields for inverse rendering. In *ICCV*, 2023. 2

[34] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, 2021. 2

[35] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995. 2

[36] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. In *ACM TOG*, 2019. 2

[37] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2, 3, 5

[38] Thomas Müller, Fabrice Rousselle, Jan Nov'ak, and Alexander Keller. Real-time neural radiance caching for path tracing. In *ACM TOG*, 2021. 8

[39] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. In *SIGGRAPH*, 2022. 1, 2, 3, 5, 7, 8

[40] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *CVPR*, 2022. 2, 4, 7

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 5

[42] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023. 2

[43] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *ICCV*, 2021. 2

[44] Radu Alexandru Rosu and Sven Behnke. Permutosdf: Fast multi-view reconstruction with implicit surfaces using permutohedral lattices. In *CVPR*, 2023. 2

[45] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 1, 2, 3, 7

[46] Alex Trevithick and Bo Yang. Grf: Learning a general radiance field for 3d representation and rendering. In *ICCV*, 2021. 2

[47] Alex Trevithick, Matthew Chan, Michael Stengel, Eric Chan, Chao Liu, Zhiding Yu, Sameh Khamis, Manmohan Chandraker, Ravi Ramamoorthi, and Koki Nagano. Real-time radiance fields for single-image portrait view synthesis. In *ACM TOG*, 2023. 2

[48] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *CVPR*, 2022. 1, 2, 3, 4, 5, 6, 7, 8

[49] Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. Microfacet models for refraction through rough surfaces. In *EGSR*, 2007. 3

[50] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021. 2

[51] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021. 2

[52] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. In *IEEE transactions on image processing*, 2004. 5

[53] Liwen Wu, Jae Yong Lee, Anand Bhattad, Yu-Xiong Wang, and David Forsyth. Diver: Real-time and accurate neural radiance fields with deterministic integration for volume rendering. In *CVPR*, 2022. 1, 2, 3, 7

[54] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *CVPR*, 2022. 2

[55] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *NeuRIPS*, 2021. 2, 4

[56] Lior Yariv, Peter Hedman, Christian Reiser, Dor Verbin, Pratul P. Srinivasan, Richard Szeliski, Jonathan T. Barron, and Ben Mildenhall. Bakedsdf: Meshing neural sdfs for real-time view synthesis. In *SIGGRAPH*, 2023. 2

[57] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 2

[58] Tizian Zeltner, Fabrice Rousselle, Andrea Weidlich, Petrik Clarberg, Jan Novák, Benedikt Bitterli, Alex Evans, Tomáš Davidovič, Simon Kallweit, and Aaron Lefohn. Real-time neural appearance models. *arXiv preprint arXiv:2305.02678*, 2023. 3, 8

[59] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2

[60] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *CVPR*, 2021. 2

[61] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5

[62] Xiaoshuai Zhang, Abhijit Kundu, Thomas Funkhouser, Leonidas Guibas, Hao Su, and Kyle Genova. Nerflets: Local radiance fields for efficient structure-aware 3d scene representation from 2d supervision. In *CVPR*, 2023. 2