

DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks

T. Neff¹ , P. Stadlbauer¹ , M. Parger¹ , A. Kurz¹ , J. H. Mueller¹ , C. R. A. Chaitanya², A. Kaplanyan² , and M. Steinberger¹

¹Graz University of Technology, Austria

²Facebook Reality Labs, USA

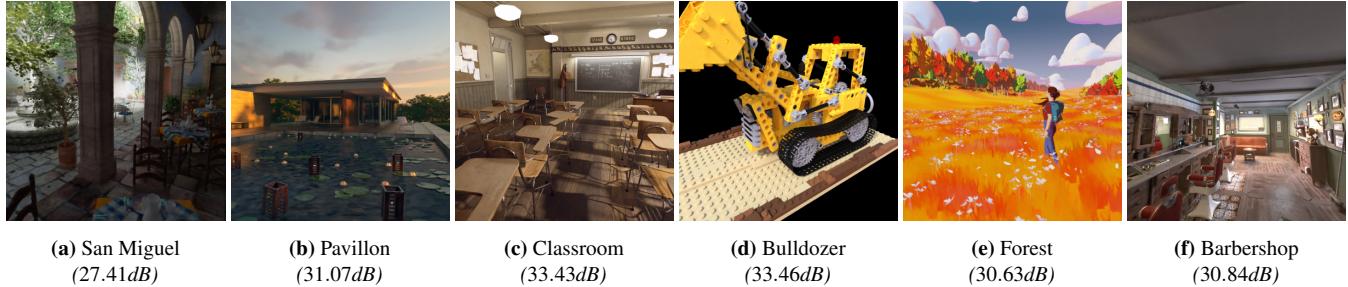


Figure 1: Example renderings with DONeRF at 400×400 pixels for our tested scenes (PSNR is in brackets). All shown results are rendered in real-time at 22 ms per frame on a single GPU and require approximately 4.35 MFLOP per pixel to compute. DONeRF requires only 4 samples per pixel thanks to a depth oracle network to guide sample placement, while NeRF uses 256 samples per pixel in total. We reduce the execution and training time by up to 48 \times and achieve better quality (NeRF average PSNR at 30.52 dB vs. our 31.14 dB).

Abstract

The recent research explosion around implicit neural representations, such as NeRF, shows that there is immense potential for implicitly storing high-quality scene and lighting information in compact neural networks. However, one major limitation preventing the use of NeRF in real-time rendering applications is the prohibitive computational cost of excessive network evaluations along each view ray, requiring dozens of petaFLOPS. In this work, we bring compact neural representations closer to practical rendering of synthetic content in real-time applications, such as games and virtual reality. We show that the number of samples required for each view ray can be significantly reduced when samples are placed around surfaces in the scene without compromising image quality. To this end, we propose a depth oracle network that predicts ray sample locations for each view ray with a single network evaluation. We show that using a classification network around logarithmically discretized and spherically warped depth values is essential to encode surface locations rather than directly estimating depth. The combination of these techniques leads to DONeRF, our compact dual network design with a depth oracle network as its first step and a locally sampled shading network for ray accumulation. With DONeRF, we reduce the inference costs by up to 48 \times compared to NeRF when conditioning on available ground truth depth information. Compared to concurrent acceleration methods for raymarching-based neural representations, DONeRF does not require additional memory for explicit caching or acceleration structures, and can render interactively (20 frames per second) on a single GPU.

CCS Concepts

- Computing methodologies → Rendering;

1. Introduction

Real-time rendering of photorealistic scenes with complex lighting is still an overly demanding problem. However, today, consumer machine learning accelerators are widespread from desktop GPUs to mobile phones and virtual reality (VR) headsets, making evaluation

of neural networks fast and power-efficient. Recent advances in implicit neural scene representations [SZW19; SMB*20; MST*20] impressively show that machine learning can be used for compact encoding and high-quality rendering of 3D scenes. Neural radiance fields (NeRFs) [MST*20] use only 1 000 000 parameters divided

among two multilayer perceptron (MLP) networks to encode scene structure alongside lighting effects. For image generation, NeRF uses traditional volume rendering drawing 256 samples for each view ray, where each sample requires a full network evaluation.

Although NeRF-like methods show significant potential for compact high-quality object and scene representations, they are too expensive to evaluate in real-time. Real-time rendering of such a representation onto a VR headset at 1440×1600 pixel per eye with 90 Hz would require 37 petaFLOPS (256 network evaluations each with $256^2 \cdot 7$ multiply add operations). Clearly, this is not possible on current GPU hardware and evaluation cost is a major limiting factor for neural representations to be used for real-time rendering. Additionally, NeRF only works well for small scale content, requiring splitting larger scenes into multiple NeRFs [ZRSK20; RJY*21], multiplying both the memory and evaluation cost.

In this work, we make neural representations practical for interactive and real-time rendering, while sticking to a tight memory budget. Particularly, our goal is to enable large scale *synthetic content* in movie quality in real-time rendering. We make the following contributions with depth oracle neural radiance fields (DONeRFs):

- We propose a compact dual network design to reduce evaluation costs for neural rendering. An *oracle network* predicts sample locations along view rays and a *shading network* places a small number of samples guided by the oracle to deliver the final color.
- We present a robust *depth oracle network* design and training scheme to efficiently provide sample locations for the shading network. The oracle uses filtered, discretized target depth values, which are readily available in synthetic content, and it learns to solve a classification task rather than to directly estimate depth.
- We introduce a non-linear transformation to handle large, open scenes and show that sampling of the shading network should happen in a warped space, to better capture different frequencies in the fore- and background, capturing content in a single network beyond the capability of previous work.
- Combining our efforts, we demonstrate high-quality real-time neural rendering of large synthetic scenes. At the same tight memory budget used by the original NeRF, we show equal quality for small scenes and significant improvements for large scenes, while reducing the computational cost by $24\text{--}98\times$.

With DONeRF, we are the first to render large-scale computer graphics scenes from a compact neural representation in real time. Additionally, DONeRF is significantly faster to train. We focus on static synthetic scenes and consider dynamic scenes and animations orthogonal to our work. Still, DONeRF can directly be used as a compact backdrop for distant parts of a game scene, or in VR and augmented reality (AR), where an environment map does not offer the required parallax for a stereo stimulus. Our source code and datasets are available at <https://dephoracnernf.github.io/>.

2. Related work

Image-based novel view synthesis Recently, image-based rendering techniques using multi-plane images (MPIs) [ZTF*18; FBD*19] managed to achieve impressive results by blending image layers. By blending between multiple MPIs [MSO*19] and using spherical image layers [BFO*20] the potential field of view can be increased

at the cost of memory efficiency. Further extending MPIs, neural basis functions can be learned to enable real-time view synthesis [WPYS21]. Alternatively, an implicit mapping between view, time or illumination conditions can be learned [BMSR20]. Although explicit image-based representations can be efficiently rendered, they typically only allow for small viewing regions [STB*19] in addition to requiring densely sampled input images, which substantially increases memory requirements for larger viewing regions.

Implicit neural scene representations Although explicit neural representations based on voxels [STH*19], MPIs [ZTF*18; FBD*19] or proxy geometry [HPP*18] enable fast novel view generation, they are fundamentally limited by the internal resolution of their representation. To circumvent this issue, implicit neural scene representations [PFS*19; SZW19] directly infer outputs from a continuous input space, such as ray origins and directions. Scene representation networks (SRNs) [SZW19] directly map 3D world coordinates to a feature representation and use a learned raymarcher to accumulate rays for the final RGB output. Similarly, neural volumes [LSS*19] use raymarching to accumulate rays in a learned, warped, volumetric representation. The quality of scene representations can be improved with periodic activation functions [SMB*20].

Neural Radiance Fields Opening a whole new subdomain of research, Mildenhall et al. [MST*20] introduced NeRF, which replaced the learned raymarching from SRN with a fixed, differentiable ray marcher. In NeRF, all ray samples are transformed into a high dimensional sine-cosine or Fourier space [TSM*20], and fed into an MLP, followed by an accumulation step to generate the final RGB output. The simplicity and impressive results inspired many adaptations to the original NeRF, sometimes being referred to as the *NeRF explosion* [DY21]: NeRFs can capture dynamic free-viewpoint video [PSB*20; LNSW21; XHKK20; PCPM21; DZY*20], generate photorealistic avatars [GTZN21; GSL*20; LSS*21], perform relighting on captured scenes [MRS*21; BXS*20; SDZ*21; BBJ*20], conditionally encode shape and appearance via latent codes [SLNG20; CMK*21; YYTK21; TY20] and compose scenes of multiple objects [OMT*21; YLSL21; NG21].

Although these NeRF variants show impressive quality, the large number of samples per ray typically makes NeRFs unsuitable for real-time applications. As a result, several recent publications incrementally improve run-time efficiency. To enable empty space skipping, neural sparse voxel fields (NSVFs) [LGZ*20] uses a self-pruning sparse voxel octree structure, where each ray sample includes information from a tri-linearly interpolated embedding of voxel vertices. Alternatively, to reduce the number of evaluations along a ray, partial integrals can be learned [LMW21]. Decomposed radiance fields (DeRFs) [RJY*20] decompose the scene with a Voronoi decomposition to train multiple NeRFs for each cell.

Baking of Neural Radiance Fields Recent research has focused on baking components of NeRF to achieve performance gains at the cost of extensive memory consumption [YLT*21; HSM*21; RPLG21; GKJ*21]. While baking could also be applied to our work, it departs from the beauty of a compact neural representation, potentially requiring hundreds of MBs up to GBs for a scene that can be represented by 4 MB in NeRF or our approach.

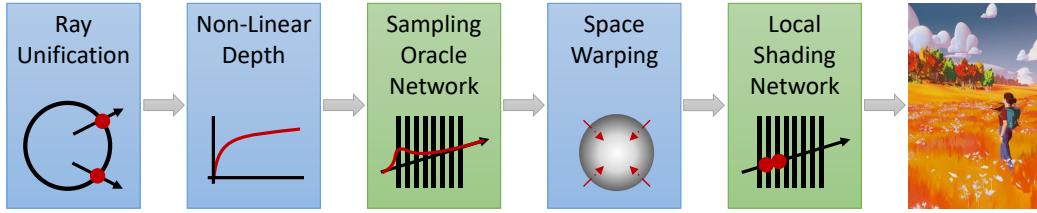


Figure 2: To enable efficient rendering of large-scale neural representations, DOnERF uses a five stage pipeline: (1) ray descriptions are unified within a view cell, (2) depth is considered in a non-linear space, (3) an oracle network estimates the importance of samples, (4) sample positions are warped towards the view cell, and (5) radiance is generated from the shading MLP with only a few samples along each ray.

In our work, we increase the inference speed of NeRF-like representations while staying in the realm of compact MLPs without additional data structures or increased storage requirements. At the memory requirement of two MLPs, we show the most significant performance improvements compared to NeRF [MST*20]. Additionally, we increase image quality and support large-scale scenes, where the original NeRF, image-based methods, and methods that require additional data structures, like NSVF [LGZ*20], struggle.

3. Efficient Neural Rendering using Depth Oracle Networks

To achieve real-time rendering of compact neural representations for generated content, we introduce DOnERF. DOnERF replaces the MLP-based raymarching scheme of NeRF [MST*20] with a compact local sampling strategy to only consider important samples around surfaces. DOnERF consists of two networks in a five-stage pipeline (Figure 2): A *sampling oracle network* predicts optimal sample locations along the view ray using classification and a *shading network* uses NeRF-like raymarching accumulation to deliver RGBA output. To remove input ambiguity, we transform rays to a *unified space* and use *non-linear sampling* to focus on close regions. Between the two networks, we warp the local samples to direct high frequency predictions of the shading network to the foreground.

View Cells For training, we use RGBD input images sampled from a *view cell*. A view cell is defined as a bounding box with a primary orientation and maximum viewing angle, *i.e.*, it captures all view rays that originate in the bounding box and stay within a certain rotation, see Figure 3. In a streaming setup, trained network weights for partially overlapping view cells can be swapped or interpolated to enable seamless transitions between larger sets of potential views. We define the view cell specifics to provide a clear way of splitting large scenes and defining the potential input to our approach. As smaller view cells reduce the visible content of a scene, smaller view cells may lead to higher quality (with the extreme being a single view). However, large view cells can work similarly well while being even more memory efficient—depending on the network capacity used to represent them. Note that this is true for any scene representation, and comes at a cost of larger memory requirements.

4. Efficient Neural Sampling

Inference performance of NeRF-like neural representations scales most significantly with the number of samples per ray. While there are methods that deploy multiple lower-capacity networks for a

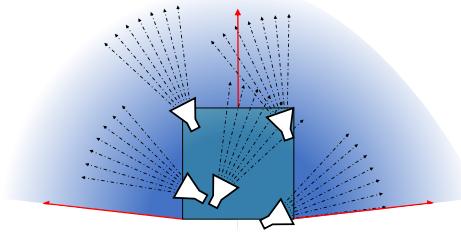


Figure 3: Top view of a view cell, defined by a bounding box, a forward direction (arrow) and a maximum viewing angle (blue with limiting arrows). Valid rays originate within the view cell and stay within the angle bounds (see example camera orientations).

moderate quality-speed tradeoff [RJY*21], the target output quality is fundamentally limited by the given network capacity. Thus, we consider network capacity optimizations orthogonal to our work. Instead, we consider different sample placement strategies to reduce the amount of samples for neural raymarching.

Uniform Sampling The default way of NeRF-style raymarching is to place samples uniformly between the near and far plane:

$$\mathbf{x}(d_i) = \mathbf{o} + d_i \cdot \mathbf{r} \quad (1)$$

$$d_i = \left(d_{min} + i \cdot \frac{(d_{max} - d_{min})}{N} \right), \quad i = [0, 1, 2, \dots, N],$$

where \mathbf{o} is the ray origin, \mathbf{r} is the ray direction, N is the number of placed samples, and d_{min} and d_{max} are the near and far plane distances from the camera pose. Sample locations are transformed to a view cell local coordinate system, divided by d_{max} , and positionally encoded to construct the feature vector \mathbf{f} , where \mathbf{c} is the view cell center:

$$\mathbf{f}(d_i) = \text{encode} \left(\frac{\mathbf{x}(d_i) - \mathbf{c}}{d_{max}} \right). \quad (2)$$

Non-linear Sampling While uniform sampling works well for individual objects and small scenes, large depth ranges are problematic. Focusing samples on objects closer to the camera is an intuitive first step to reduce network evaluations without losing quality. For non-linear sample placement, we use a logarithmic non-linearity:

$$\tilde{d}_i = d_{min} + \frac{\log(d_i - d_{min} + 1)}{\log(d_{max} - d_{min} + 1)} \cdot (d_{max} - d_{min}). \quad (3)$$

NDC Sampling NeRF [MST*20] suggests to transform rays into an average camera frame, and to uniformly sample within the projected normalized device coordinates (NDC), directly feeding NDC samples into positional encoding. This approach is only applicable if all samples lie strictly in the front hemisphere of the average camera frame, which is a limitation not shared by the other sampling strategies. Large deviations from the average camera frame lead to increasingly high non-linear perspective distortions, which may impact the learning process. We refer the reader to the appendix in the original NeRF paper for the detailed derivations; we use the same transformation with $d_{min} = 1$ and $d_{max} = \infty$.

Space Warping Although non-linear sampling focuses the samples on the foreground, positional encoding is applied equally for fore- and background. Early training for large scenes showed that the background often contains high frequencies that must be damped by the network while the foreground requires those to achieve sufficient detail. This is not surprising, as real cameras and graphics techniques such as mip-mapping average background details.

To remedy this issue, we propose a warping of the 3D space towards the view cell center. We warp the entire space using a radial distortion, bringing the background closer to the view cell for positional encoding. Initial experiments showed that using an inverse square root transform works well:

$$\tilde{\mathbf{f}} = \text{encode}((\mathbf{x}(\tilde{d}_i) - \mathbf{c}) \cdot W(\mathbf{x}(\tilde{d}_i) - \mathbf{c})) \quad (4)$$

$$W(\mathbf{p}) = \frac{1}{\sqrt{|\mathbf{p}| \cdot d_{max}}}. \quad (5)$$

While ray samples do not follow a straight line after warping, sample locations in space stay consistent, *i.e.*, samples from different view points landing at the same 3D location are evaluated equally. See Figure 4 for a visualization of the *uniform*, *logarithmic* and *log+warp* placement strategies. The NDC sampling strategy is linear along each ray in NDC space, but follows a linear sampling in *disparity* from the near plane to infinity in the original space, with a more aggressive $\frac{1}{x}$ sampling curve compared to logarithmic sampling. From the perspective of the network input, it therefore combines the uniform and logarithmic sampling approaches.

Local Sampling Even when focusing samples on the foreground, NeRFs with large sample counts spend many samples in empty space. Given a ground truth surface representation, *e.g.*, a depth texture, it is possible to take a fraction of the samples of a trained NeRF around the surface and still achieve mostly equal quality. This inspires the following question: *Given a ground truth depth texture to place samples during training, what is the best quality-speed tradeoff that can possibly be reached?*

Ablation Study To determine the effectiveness of the different sampling strategies, we run an ablation study based on the original NeRF. For all experiments, we assume static geometry and lighting and test on a single view cell. We vary the number of samples per ray N between $[2, 4, 8, 16, 32, 64, 128]$ and only use a single MLP (without the refinement network proposed in the original NeRF work which dynamically influences sample placement). To investigate local sampling, we perform *uniform*, *logarithmic*, *log+warp* and *NDC* sampling around the ground truth depth, keeping the step

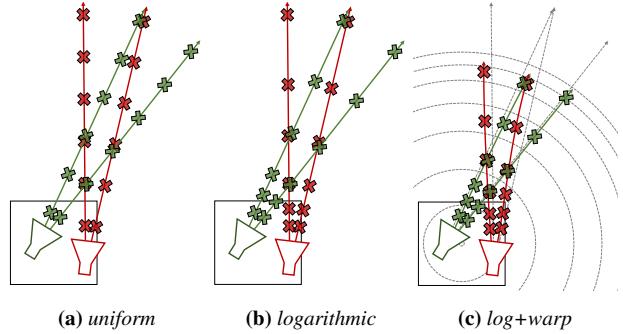


Figure 4: Different sampling approaches visualized: (a) uniform samples in equal steps between the near and far planes; (b) logarithmic sampling reduces the sample distance for close samples in favor of spacing out far samples; (c) log+warp pulls the space closer to the view cell center, making the scene appear smaller to the NeRF and bending rays (compare to the gray straight lines).

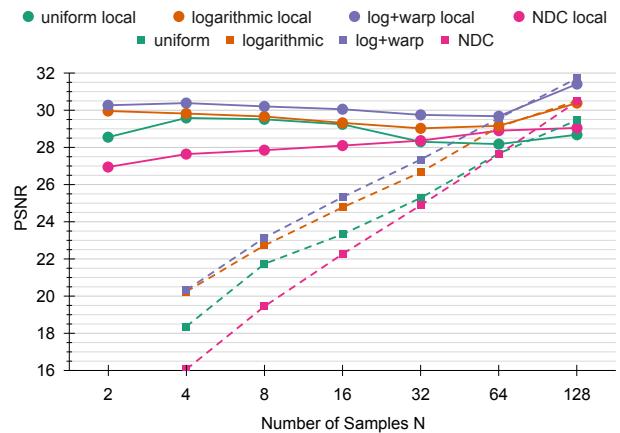


Figure 5: Average PSNR results for various sample reduction schemes over the sample count N . Uniform sampling roughly increases quality by 1.8 dB for each doubling of N . logarithmic increases PSNR by 1.3 dB and inverse square root warping (log+warp) adds another 0.6 dB on top. Sampling in NDC only manages to surpass uniform sampling at 128 samples. When sampling around the ground truth depth (local), the number of samples hardly affects quality. Still, PSNR roughly increases by 0.75 dB with logarithmic and another 0.6 dB with log+warp. Local NDC sampling performs worse than all other local sampling approaches until $N = 32$.

size identical to $N = 128$ for all sample counts. We conduct our experiments on four diverse scenes, *Bulldozer*, *Forest*, *Classroom* and *San Miguel*. For more details on the evaluation setup and the evaluated test scenes, please refer to Section 6 and Appendix E.

Averaged results are shown in Figure 5, while per-scene details can be found in Appendix A. First, for non-local sampling, the results show that *logarithmic* sampling increases quality over *uniform* while *log+warp* increases quality further. With *log+warp* the number of samples can be halved to still achieve equal quality to *uniform*.

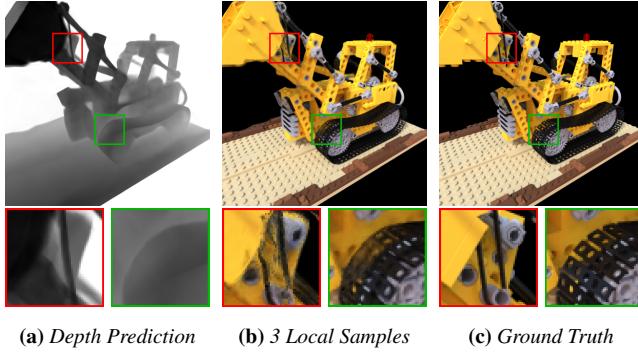


Figure 6: (a) A depth oracle network with a single depth output smooths depth around discontinuities. (b) As a result, local sampling mixes fore- and background and distorts features. (c) This becomes apparent when compared to the ground truth.

Second, with local sampling, the number of samples can be reduced to two with nearly no decrease in quality. On average, *log+warp* adds about 1.3 dB in quality over *uniform* for local sampling. Sampling in *NDC* only reaches competitive results at more than 64 samples, falling behind *uniform* sampling in our evaluation. Since the network inputs in *NDC* are still uniform, the underlying non-linear transformation between different rays must be reconstructed by the network, which seems difficult at lower sample counts.

Our results indicate that—given an ideal sampling oracle—significant gains in quality-speed tradeoffs can be achieved by placing samples locally. However, in practice, ground truth depth is often not available during inference for neural rendering due to memory or computational constraints: large-scale scenes require a significant amount of storage to represent via geometry or individual depth maps, and reprojection would be necessary to generalize to novel views. Thus local sampling from ground truth depth during inference can be considered a niche scenario, and we therefore target an efficient and compact representation via an MLP-based sampling oracle that only uses ground truth depth during training.

5. Sampling Oracle Network

As mentioned before, sampling around a known ground truth surface representation can reduce the number of required samples by up to $64\times$. However, relying on explicit surface representations would defeat the purpose of having a compact neural representation. Therefore, we introduce an oracle network to predict ideal sample locations for the raymarched shading network. This oracle network takes a ray as input and provides information about sample locations along that ray. We found that using an MLP of the same size as the shading network generates sufficiently consistent depth estimates for the majority of rays in simple scenes. However, accurate estimates around depth discontinuities remain difficult to predict, leading to significant visual artifacts (Figure 6).

To mitigate this issue, we start with the following observation: In general, the exact geometric structure of the scene is faithfully reconstructed using neural raymarching, where samples are also placed in empty space. Thus, we can allow the oracle more freedom.

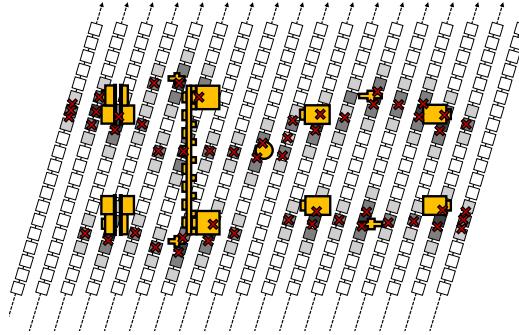


Figure 7: A horizontal slice through the Bulldozer dataset. Filtering the depth classification target (black = 1, white = 0) in both image dimensions and along depth smooths the classification target. An oracle producing such an output still results in high quality sample locations (red \times), with additional samples placed in free space. A smooth target is easier to learn as labels vary with lower frequency.

While it must predict sample locations around the surface, it may provide *additional* estimates. Consider neighboring rays around a depth discontinuity, either hitting the foreground or the background: Representing this discontinuity accurately in ray space is difficult, as small changes in ray origin or direction may alternate between fore- and background. However, if the oracle is allowed to return the same result for all rays around discontinuities, *i.e.*, sampling at the fore- and the background, the oracle’s task is easier to learn.

5.1. Classified Depth

Interestingly, a simultaneous prediction of multiple real-valued depth outputs did not improve results compared to a single depth estimate per pixel in our experiments. Alternatively, the oracle can output sample likelihoods along the ray—*i.e.*, a likelihood that a sample at a certain location will increase image quality. Unlike NeRF’s refinement network, we only want to evaluate the oracle network once and reduce the sample count of the shading network.

To this end, we propose that the oracle is trained via classification, where each class corresponds to a discretized segment along the ray. For every discrete ray segment, a high value indicates that it should receive (multiple) samples; a low value indicates that the segment can be skipped. The surfaces are represented accurately, *i.e.*,

$$C_{x,y}(z) = \begin{cases} 1, & \text{if } d_z \leq d_s < d_{z+1} \\ 0, & \text{otherwise,} \end{cases} \quad (6)$$

where C is the classification value, d_s corresponds to the depth value of the first surface along the ray and d_z and d_{z+1} describe the discretization boundaries for the ray segment z . This leads to a one-hot encoding as a target that can be trained using the common binary cross-entropy (BCE) loss.

To further aid the depth oracle in predicting consistent outputs at depth discontinuities, we provide a multi-class target that is filtered in image-space and along depth. We blur depth values from neighboring pixels in the ground truth target. To generate this filtered target, we use a radial (Euclidean distance) filter to include values

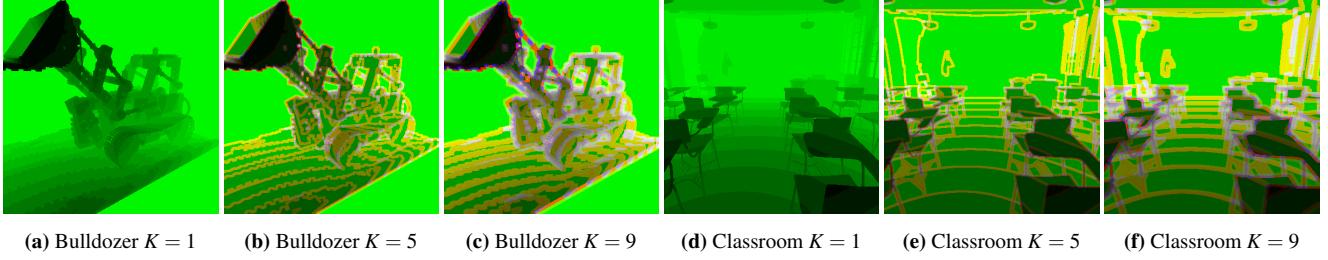


Figure 8: Visualization of the classified depth target for 16 classes along each ray and the largest classes mapped to G, R, B, i.e., a green value indicates a single class with the brightness encoding depth; a gray value means that all classes are at similar depth; and different colors in proximity indicate that classes vary. Small features are smoothed to neighboring pixels with increasing filter sizes (1-5-9).

of neighboring rays with a lower contribution:

$$\hat{C}_{x,y}(z) = \max_{i,j \in \pm [K/2]} \left(C_{x+i,y+j}(z) - \frac{\sqrt{i^2 + j^2}}{\sqrt{2} \cdot [K/2]} \right) \quad (7)$$

where K is the filter size. For example, using a 5×5 filter, rays at a distance of 3, 2, 1, 0 pixels contribute 0, ≈ 0.30 , ≈ 0.65 , 1 to the output, respectively. For multiple depth values with the same discretization result, we only take the maximum result (contribution from the closest neighboring ray) to ensure a valid classification target as well as that the largest value is placed at the actual surface.

Following the idea of label smoothing, we also filter along the depth axis, using a simple 1D triangle filter with kernel size Z :

$$\hat{C}(z) = \min \left(\sum_{i=-[Z/2]}^{[Z/2]} \hat{C}(z+i) \frac{[Z/2] + 1 - |i|}{[Z/2] + 1}, 1 \right). \quad (8)$$

From a classification point of view, filtering decreases false negatives (at the cost of false positives) and thus ensures that important regions are not missed. This becomes apparent when considering that rays exist in a continuous space, *i.e.*, the oracle does not need to learn hard boundaries at depth discontinuities. Translating a higher false positive rate to raymarching, we increase the likelihood for sampling regions that do not need any samples. Thus, overall, we need to place more samples to hit the *right* sample locations, while at the same time reducing the chance to miss surfaces.

Given that the oracle network serves a similar purpose as the coarse network in NeRF with the same capacity, while being evaluated only once instead of 64 times, allowing the network to move towards false positive classifications is essential. Additionally, false positives are handled by the local raymarching network, as empty space will still result in no image contribution, even at low sample counts. A missed surface on the other hand would clearly reduce image quality. Figure 7 shows an example slice for a fully filtered target and Figure 8 shows visualizations for different filter sizes. Note that the filtering only applies to the training targets for the depth oracle—no filtering is done during inference.

Finally, to translate the classification outputs to sample locations, we use the same approach as NeRF [MST*20] when going from the coarse to the fine network. Compared to NeRF, which builds a piecewise-constant probability density function (PDF) from the opacity outputs of the coarse shading network evaluated at multiple sample locations, we interpret our depth oracle output vector

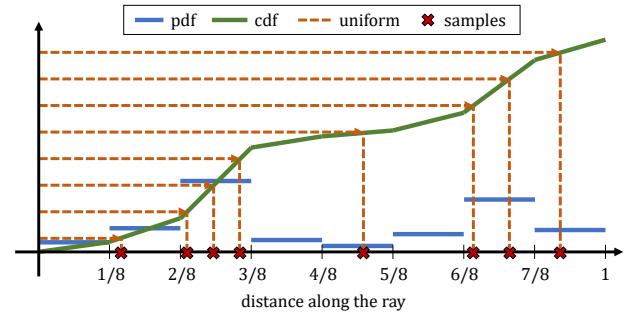


Figure 9: To place samples along the ray, we treat the classified depth output from the oracle as a piecewise-constant PDF (blue) and translate it into a CDF (green). Sampling the CDF at uniform steps (orange) concentrates samples (red \times) around regions with a high classification value.

(which comes from a single oracle network evaluation) directly as a piecewise-constant PDF and similarly sample along the inverse transform of the corresponding cumulative distribution function (CDF); see Figure 9.

5.2. Ray Unification and Oracle Input

To take the burden of disambiguating rays originating from different starting locations from the oracle, we map every ray starting location onto a sphere circumscribed around the view cell, see Figure 10. This unification works well for arbitrary views looking outside the view cell. For 360° object captures, such as in the original NeRF work, our ray unification scheme be applied in an inverse manner, where the circumscribed sphere is placed around the object instead.

To ease the task of the oracle network further, we supply 3D positions along the ray as input. We place those at the centers of the discretized depth ranges and provide them as additional inputs to the first network layer. We do not use positional or Fourier encoding for the depth oracle inputs, as this did not improve results in our experiments. To work in unison with the shading network, for which we place samples logarithmically (Section 4), we also perform the same logarithmic transformation on the classification steps.

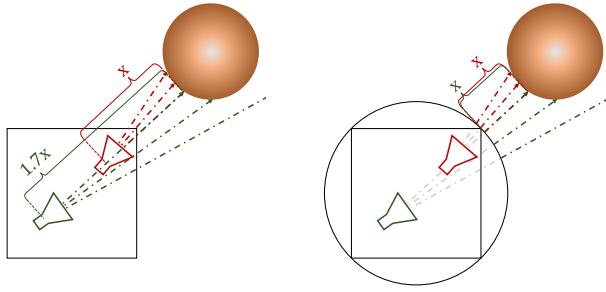


Figure 10: Ray unification maps ray starting points to a sphere surrounding the view cell. Without ray unification (left) the same ray is encoded differently and requires different depth values; after ray unification (right) identical rays have identical depth values.

5.3. Optimal Filter Sizes and Depth Oracle Inputs

We conduct an ablation study to evaluate the impact of the filter sizes for our neighborhood filter K (Equ. 7), depth smoothing filter Z (Equ. 8) and the number of 3D input positions ($I = [1, 128]$). We use the classified depth oracle network as the input for a locally raymarched shading network, similar to the experiment in Section 4.

We vary the number of samples per ray $N = [2, 4, 8, 16]$. Both the depth oracle network and the shading network contain 8 hidden layers with 256 hidden units. We first train the depth oracle network for 300 000 iterations, before training the shading network for 300 000 iterations using the oracle’s predictions. To illustrate the importance of our depth classification, we compare against a depth oracle that only predicts a single depth value with (*SD unified*) and without (*SD*) ray unification. Our metric is the resulting quality of the RGB output of the shading network, which should improve when given better sample positions by the depth oracle.

The results in Table 1 show that (1) ray unification adds about 0.6 dB in PSNR, (2) using a classification network adds another 0.6 dB–1.5 dB, (3) providing multiple samples along the ray as input adds 1 dB, (4) the neighborhood filter adds 1.3 dB–1.8 dB, and the depth smoothing filter adds 0.1 dB. These improvements come at no inference cost (filtering) or virtually no inference cost (ray unification, multi input). In total, our additions improve the PSNR by 3.3 dB–4.7 dB. Note that an even larger filter size may reduce overall quality, as many samples are placed in empty space rather than on the surface.

6. Evaluation

To evaluate DOnERF, we focus on three competing requirements of (neural) scene representations: *quality* of the generated images, *efficiency* of the image generation, and *compactness* of the representation. Clearly tradeoffs between them are possible, but an ideal representation should generate high-quality outputs in real-time, while being compact and extensible, *e.g.*, for streaming dynamic scenes. We compare against NeRF [MST*20], NSVF [LGZ*20], Local Light Field Fusion (LLFF) [MSO*19] and Neural Basis Expansion (NeX) [WPYS21] to evaluate methods that choose different tradeoffs among our three goals.

Table 1: PSNR and FLIP results averaged over all scenes, evaluating the neighborhood filter size ($K-X$), the depth smoothing filter size ($Z-X$) and the number of inputs for the depth oracle ($I-X$) over the number of samples per ray N used for raymarching. We also compare against an oracle that only predicts a single depth value (SD), as well as a single depth value with unified input (SD unified).

Method \ N	PSNR \uparrow				FLIP \downarrow			
	2	4	8	16	2	4	8	16
SD	26.686	27.401	28.220	29.085	0.092	0.084	0.078	0.073
SD unified	27.423	28.052	28.825	29.554	0.085	0.079	0.074	0.071
K-1 Z-1 I-1	27.325	28.697	30.068	31.145	0.082	0.073	0.065	0.061
K-5 Z-1 I-1	28.685	30.521	31.988	32.982	0.075	0.066	0.059	0.055
K-5 Z-1 I-128	29.956	31.746	32.951	33.760	0.067	0.061	0.056	0.053
K-5 Z-5 I-128	30.071	31.842	33.027	33.836	0.067	0.061	0.056	0.053
K-9 Z-1 I-1	28.831	30.881	32.617	33.495	0.075	0.065	0.057	0.055
K-9 Z-1 I-128	29.299	31.645	33.125	33.994	0.071	0.061	0.056	0.053
K-9 Z-9 I-128	28.737	30.847	32.261	33.302	0.076	0.066	0.060	0.056

These methods capture a mix between being strictly MLP-based (NeRF), using explicit structures and MLPs (NSVF, NeX) and using a mostly image-based representation (LLFF). For NeX, we include an additional variant that does not bake radiance coefficients and neural basis functions into an MPI, but recomputes those via MLP inference during test time (NeX-MLP). For NSVF, we run a grid search and evaluate on three representative variants (*NSVF-small*, *NSVF-medium* and *NSVF-large*) that capture the lowest memory footprint, best quality-speed tradeoff, and best quality respectively. Furthermore, we include a variant of NeRF that uses our *log+warp* sampling to show the effect of the sampling strategy in isolation (NeRF (*log+warp*)). See Appendix E for details about the methods.

We analyze the ability to extract novel high-quality views for generated content where reference depth maps are available during training. As an additional proof-of-concept, we extract estimated depth maps from a densely sampled NeRF for each scene, and use these depth maps to train our depth oracles, showcasing a solution for scenes without available ground truth depth. We evaluate quality by computing PSNR and FLIP [ANA*20] against ground truth renderings, efficiency as FLOP per pixel and compactness by total storage cost for the representation. For all methods, images are downsampled to a resolution of 400×400 to speed up training.

6.1. Training and Real-time Implementation

We use K-5 Z-5 I-128 for our depth oracle network (see Section 5.3) and use various numbers of samples per ray, named DOnERF- X , where X denotes the number of samples per ray. We transform each sample logarithmically and warp it towards the view cell, as described in Section 4. We train each network for 300 000 iterations and use the checkpoint with the lowest validation loss for testing. We use Adam [KB15] with a learning rate of 0.0005 and a batch size of 4096 samples per iteration.

For the RGB output of the shading network we use standard MSE loss, while the depth classification uses BCE loss. Furthermore, during initial experiments, we observed that shading networks with low sample counts tend to “cheat” in their outputs, relying on the opacity outputs to mix the background color into the accumulation,

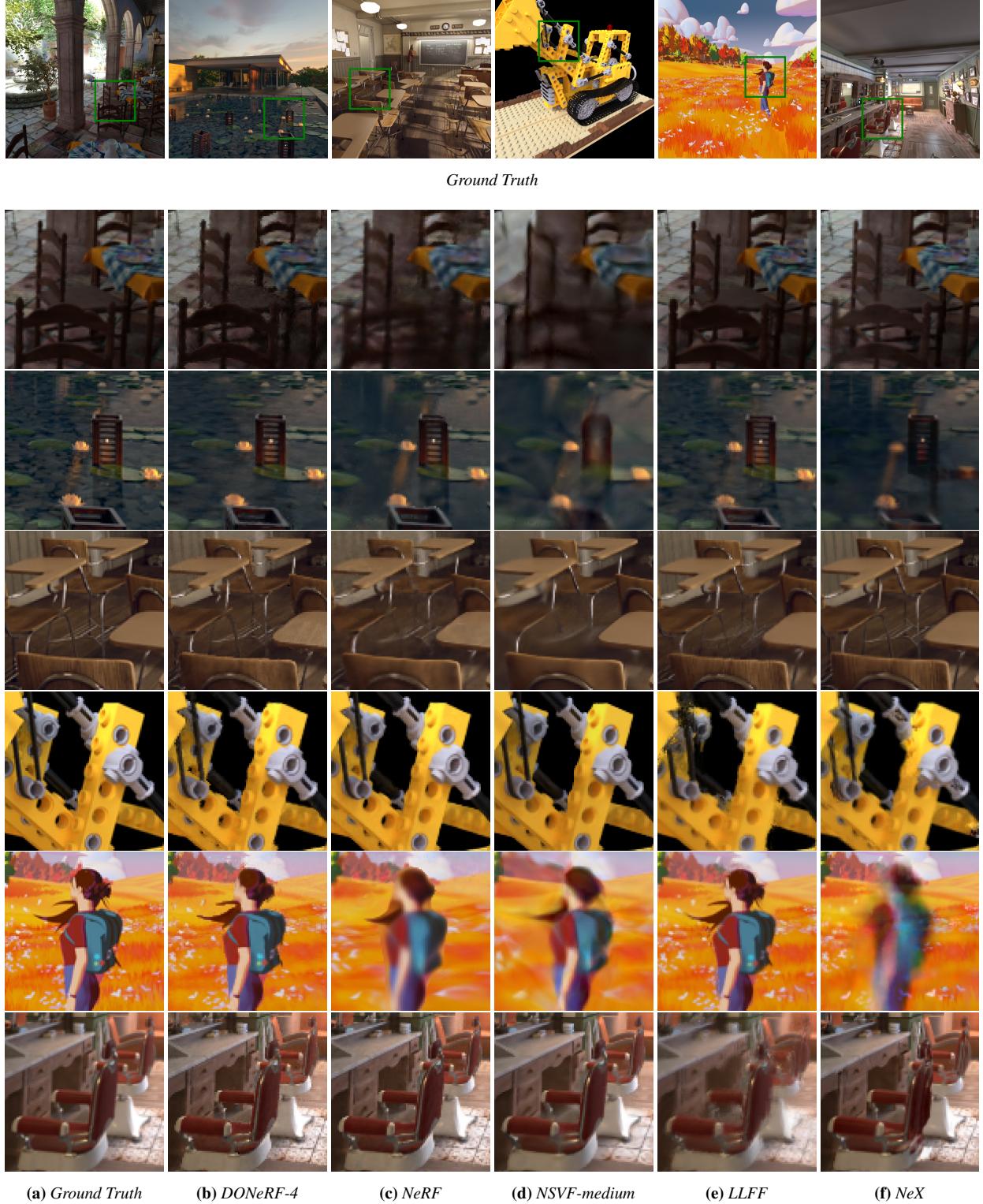


Figure 11: Even on our challenging dataset, DOnErF achieves higher quality on average than all other methods at only 4 samples per ray. NeRF manages to faithfully reconstruct smaller scenes such as Bulldozer and Barbershop, but struggles with large-scale scenes such as Forest or highly complex geometry such as in Classroom. NSVF shows worse quality on average compared to DOnErF-4 at increased memory requirements and much worse performance. Although LLFF requires significantly more memory, it still struggles to represent fine details accurately, even at a cropped field of view. NeX achieves good quality for San Miguel, but larger rotations and offsets from its reference pose cause significant artifacts due to its explicit MPI representation.

Table 2: Across all our test scenes, DOnRF shows a significant improvement in quality-speed tradeoff, beating NeRF in most scenes with only 2 – 4 samples per ray, resulting in $48\times$ to $78\times$ fewer FLOP per pixel at equal or better quality. Where NSVF struggles to achieve a consistent quality or performance for our tested scenes, especially for larger scenes such as Forest, DOnRF performs consistently well across all scales. LLFF is the cheapest to compute, but fails to replicate the quality of the other methods, in addition to requiring storage that scales unfavorably with the amount of training images. Although NeX is also very efficient at rendering, it suffers from artifacts related to the MPI representation when poses differ too much from the reference pose, and requires roughly $24\times$ the amount of storage to achieve worse quality than DOnRF across the board. NeX-MLP is able to remedy some of the artifacts at a cost of $9\times$ worse performance compared to DOnRF-4. Finally, using depth maps extracted from a dense NeRF without depending on available ground truth depth, DOnRF-noGT still achieves the best tradeoff between performance, quality and storage overall. Top results in each column are color coded as **Top 1**, **Top 2** and **Top 3**.

		San Miguel	Pavillon	Classroom	Bulldozer	Forest	Barbershop	Average	
Method	Storage [MiB]	MFLOP per pixel	PSNR FLIP						
DONeRF-2	3.6	2.70	26.01 .094	30.50 .103	31.66 .061	30.15 .063	29.29 .082	29.41 .074	29.50 .079
DONeRF-2-noGT	3.6	2.70	25.33 .098	29.84 .103	30.11 .067	26.92 .077	28.36 .089	29.01 .075	28.26 .085
DONeRF-4	3.6	4.36	27.41 .080	31.07 .098	33.43 .058	33.46 .048	30.63 .077	30.84 .065	31.14 .071
DONeRF-4-noGT	3.6	4.36	26.19 .090	30.69 .096	31.44 .061	29.78 .060	29.31 .086	30.42 .067	29.64 .077
DONeRF-8	3.6	7.66	28.65 .071	31.46 .096	35.23 .048	35.88 .039	32.09 .070	31.72 .060	32.50 .064
DONeRF-8-noGT	3.6	7.66	26.88 .086	31.56 .091	33.19 .055	32.96 .047	29.98 .084	31.73 .062	31.05 .071
DONeRF-16	3.6	14.29	29.67 .065	31.79 .094	36.27 .045	36.98 .036	31.32 .074	32.15 .059	33.03 .062
DONeRF-16-noGT	3.6	14.29	27.70 .078	32.22 .088	34.63 .049	35.41 .040	30.74 .079	32.80 .057	32.25 .065
NeRF	3.2	211.42	25.19 .117	29.54 .115	34.02 .056	36.83 .038	23.90 .151	33.63 .052	30.52 .088
NeRF (log + warp)	3.2	211.42	28.98 .074	32.88 .089	35.19 .051	36.22 .040	28.97 .101	33.60 .055	32.64 .068
NSVF-small	2.3	74.66	24.00 .132	29.42 .110	31.00 .070	25.75 .167	23.79 .159	27.72 .094	26.95 .122
NSVF-medium	4.6	132.03	25.07 .110	29.81 .105	33.04 .055	26.51 .163	25.08 .135	29.62 .077	28.19 .108
NSVF-large	8.3	187.52	25.73 .097	30.48 .099	34.06 .051	33.14 .042	26.05 .119	30.61 .061	30.01 .078
LLFF	4130.6	.03	24.53 .106	27.50 .123	24.87 .114	24.76 .114	22.19 .148	24.13 .129	24.66 .122
NeX	88.8	.06	28.07 .094	26.28 .174	30.34 .085	29.20 .072	20.95 .220	22.98 .152	26.30 .133
NeX-MLP	89.0	42.71	30.68 .060	30.41 .102	34.10 .046	34.03 .046	24.65 .125	29.45 .075	30.55 .076

resulting in a darkening with black background color. At low sample counts, this tends to hurt generalization, especially when rays *should* accumulate to an opacity of at least 1 for opaque surfaces. As a result, dark pixel artifacts are visible in some test set views, which we remedy by adding an additional *opacity loss* term that forces the accumulated ray opacity δ to be *at least* 1:

$$\text{loss}_O = \begin{cases} 0, & \text{if } \sum_{i=1}^X \delta_i \geq 1 \\ \left(\left(\sum_{i=1}^X \delta_i \right) - 1 \right)^2, & \text{otherwise,} \end{cases} \quad (9)$$

where X is again the number of samples per ray. Our final loss function is a weighted sum of both the MSE and the opacity loss

$$\text{loss} = \alpha \cdot \text{loss}_{\text{MSE}} + \beta \cdot \text{loss}_O. \quad (10)$$

For all DOnRF experiments, we use $\alpha = 1.0$ and $\beta = 10.0$ to conservatively remove all dark pixel artifacts. Lower values for β could be selectively applied to further increase quality in certain scenes. The network architecture of DOnRF is visualized in Appendix D. For DOnRF without ground truth depth (DOnRF-X-noGT), we train a dense NeRF with 12 layers of 512 hidden units each using 128 samples per ray, and extract depth maps for all poses.

Using these depth maps, we train DOnRF in exactly the same way as when using the reference depth maps.

For our prototype real-time implementation of DOnRF, we use a combination of TensorRT and CUDA. Ray unification, space warping, feature generation and raymarching run in custom CUDA kernels with one thread per ray. Both networks are evaluated using TensorRT in half floating point precision. All parts still offer significant optimization potential, by, *e.g.*, using multiple threads during input feature generation or reducing the bit length for the network evaluation further.

Nevertheless, we already achieve real-time rendering at medium resolutions on an NVIDIA RTX 3090. Ray unification and first feature generation (0.21 ms), the oracle network (12.64 ms), space warping and the second feature generation (3.46 ms), the shading network (34.9 ms), and final color generation (0.09 ms) take a total of 51.3 ms for a 800×800 image and 2 samples per ray, *i.e.*, about 20 frames per second. Note that the shading network still is the major cost in our setup and that computation times increase nearly linearly with the number of samples: 34.9 ms, 65.4 ms, 136.4 ms and 270.4 ms for 2, 4, 8 and 16 samples, respectively.

6.2. Dataset

We collect a set of scenes that showcase both fine, high-frequency details as well as large depth ranges to validate that DOnERF is applicable to a wide variety of scenes. All datasets are rendered using Blender, using their Cycles path tracer to render 300 images for each scene (taking approximately 20 minutes per image on a 64 core CPU), which we split into train/validation/test sets at a 70%/10%/20% ratio. For each scene, we define a view cell that showcases a wide range of movement and rotation to reveal disocclusions and challenging geometry, while not intersecting geometry within the scene. Poses are randomly sampled within the view cell, limiting the rotation to 30 degrees in pitch and 20 degrees in yaw relative to the initial camera direction. We limit the rotation to be comparable to plane-based representations such as NeX and LLFF—DOnERF would be able to handle unrestricted rotation angles. Please refer to Appendix C for more details about the scenes.

6.3. Comparisons

The results of our main evaluation are summarized in Table 2 and qualitative example views are shown in Figure 11. In the following we individually compare the competing approaches to DOnERF in terms of *quality*, *efficiency*, and *compactness*.

NeRF Compared to NeRF, the advantages of both our sampling strategy and oracle network are immediately visible. On average, DOnERF achieves equal or better quality with only 2–4 samples per ray compared to NeRF’s 256. At 16 samples, DOnERF achieves up to 7 dB higher PSNR, and outperforms NeRF on all scenes except for *Barbershop*. At the same time, DOnERF is 15–78× faster to evaluate than NeRF. As both only use two MLPs, they are among the most compact methods—the small increase in memory of DOnERF is due to the increased number of inputs and outputs of our oracle network compared to the coarse NeRF network. Even when applying our improved sampling strategy for NeRF (log+warp), DOnERF achieves equal quality with just 8 samples. Overall, DOnERF is superior compared to NeRF—especially for large open scenes—both when considering *quality*, *efficiency* and *compactness* as a whole, and when considering them individually.

NSVF The comparison to NSVF is interesting, as NSVF is able to adjust the tradeoff between quality, efficiency and compactness by changing the voxel size of its representation. Even though NSVF-small is 2.5 dB lower in quality compared to DOnERF-2, it takes 27× longer to evaluate. NSVF-medium and NSVF-large are similar in quality to DOnERF-2 and DOnERF-4 respectively, but take 43× times longer to evaluate. As NSVF only uses a single MLP, using fewer voxels can be advantageous if a very low memory footprint is required at the cost of quality, and NSVF-small is the smallest scene representation in our comparison. However, to achieve competitive quality, NSVF requires more than 2× the amount of memory of DOnERF. Additionally, NSVF takes about 6× longer to train than DOnERF and also required ground truth depth for initialization for our challenging scenes. Thus, for every configuration, a DOnERF exists that provides identical or better *quality* at significantly better *compactness* that can be evaluated much more *efficiently*.

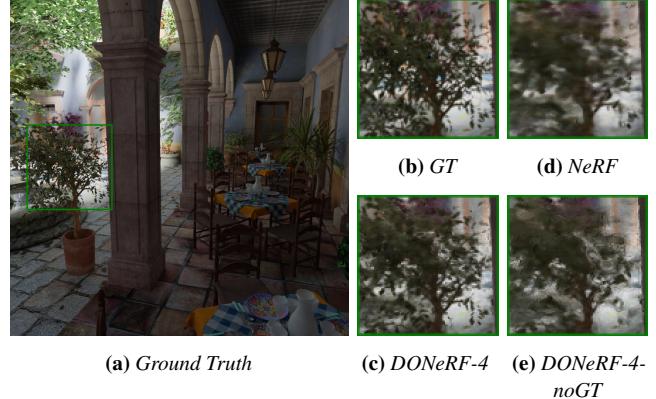


Figure 12: (a) Example ground truth view of the test set of San Miguel with the corresponding inset (b). (c) DOnERF manages to preserve high-frequency detail around the leaves of the tree at only 4 samples per ray. (d) Although NeRF also preserves the tree faithfully, the edges are blurrier. (e) Even when trained with extracted NeRF depth maps (no GT depth), DOnERF produces slightly sharper results than NeRF.

LLFF Compared to purely image-based methods like LLFF, the advantages of *neural* scene representations become apparent. Even when using the entire training set as the basis for image-based rendering (and thus requiring more than 1000× the memory of DOnERF) and cropping the image to remove border artifacts, LLFF achieves the lowest quality among all tested methods. Only for the highly challenging views of *Forest* and *San Miguel*, PSNR values are close to NeRF but still 1 dB–7 dB from DOnERF-2. However, due to its simplicity of only selecting few, small light fields for each pose at test time and blending between them, LLFF is the fastest method for novel view generation. Thus, if memory is no concern, LLFF may be the preferred option for low-power rendering.

NeX NeX shows similar artifacts to LLFF, in that it suffers from its explicit MPI representation when generating views that differ too much from its reference view. Although it is very fast to evaluate (45× faster than DOnERF) and achieves competitive quality for *San Miguel*, overall it requires 25× more memory compared to DOnERF, and its quality is 3 dB lower than DOnERF-2 on average. Looking at the improved quality of NeX-MLP, we can confirm that transferring its neural scene representation into an MPI comes at a significant loss in quality. Compared to NeX-MLP, DOnERF achieves better quality using only 4–8 samples for most scenes, at a 10× speedup and 25× more compactness. For simple scenes, limited fields of view, and if compactness is no concern, precomputing an MPI from a neural scene representation enables high-resolution real-time performance. However, when targeting higher quality or streaming, a complete neural scene representation, like DOnERF, is the better choice. Finally, NeRF-like representations have also been shown to support dynamic scenes and relighting, which may tilt the scales further towards an approach like DOnERF, where the memory consumption of MPI videos becomes even more prohibitive.



Figure 13: (a) Example image and ground truth depth from the Barbershop test set with the corresponding inset (b). (c) The reflective mirrors cannot be accurately represented by just sampling around the mirror’s depth in DOnRF. (d) NeRF places samples behind the mirror, constructing a virtual reflected room. (e) As depth maps extracted from NeRF include the virtual room, DOnRF-4-noGT learns to represent the mirrors with much fewer samples than NeRF.

6.4. DOnRF without Ground Truth Depth

Our proof-of-concept DOnRF that is trained without available ground truth depth (DOnRF-X-noGT) loses about 1 dB on average compared to DOnRF (see Table 2). The losses are largest for *Bulldozer*; for *Pavillon* and *Barbershop* DOnRF-X-noGT actually outperforms DOnRF for higher sample counts. For *Bulldozer*, NeRF outputs high frequency depth values across the entire background, which impedes the task of the depth oracle, only to create samples with zero contribution during shading—adding background detection to remove those samples during oracle training would likely resolve this issue. Looking at *San Miguel* (Figure 12), DOnRF is capable of reconstructing fine details for foliage. DOnRF-X-noGT interestingly produces sharper results than NeRF although we use a single NeRF depth output as ground truth depth. We attribute DOnRF’s ability to recover such high quality to filtering the depth targets and being able to place samples all around the foliage.

Even more surprising is the fact that DOnRF-X-noGT can outperform DOnRF, as in *Pavillon* and *Barbershop* (see Figure 13). Transparent surfaces and fully reflective mirrors pose an issue for DOnRF, as the low sample count combined with only a single depth estimate is not sufficient for the network to perfectly replicate

these challenging view-dependent phenomena. However, for these scenes, NeRF essentially learns to place samples at multiple surfaces (for the refractive water) and in a virtual mirrored room (for the reflective mirror), and thus DOnRF-X-noGT places samples better than DOnRF for these parts and can reach higher quality for increased sample counts.

Overall, these results show that even without ground truth depth to train its depth oracle network, DOnRF is able to achieve better quality compared to NeRF at much lower sample counts, and provides the best tradeoff in terms of quality, performance and storage requirements. Thus, a perfectly accurate depth estimate is not necessary to benefit from using DOnRF.

7. Conclusion, Limitations and Future Work

Starting from an evaluation of the sampling strategies of NeRF, we showed that sampling non-linearly and warped towards a view cell is beneficial in a variety of scenes. Using ground truth depth for optimal sample placement, we showed that *local sampling* achieves equal quality with as few as 2 samples per ray compared to a fully sampled network with 128 samples. From this insight, we proposed our *classified depth oracle network* which discretizes the space along each ray, and spatially filters the target across *x*, *y* and *depth* to further improve the sample placement for challenging geometric scenarios. Using our oracle network to guide sample placement for a raymarched shading network, our compact DOnRF approach achieves equal or better quality compared to NeRF with 256 samples across most scenes, while using as few as 2 samples per ray at the memory requirement of only two MLPs. Compared to other scene representations and light field methods, DOnRF compares favorably in terms of storage requirements by a large margin and outperforms all other methods in quality. Only image-based methods can be rendered faster than DOnRF. Nevertheless, DOnRF makes a big step towards rendering directly from *neural* scene representations with all their advantages in real-time.

Partially transparent objects and mirror-like surfaces can pose an issue when using standard depth maps for training, as the first surface depth value does not represent the necessary sample locations along the ray. Fortunately, due to being a classification network, it would be straightforward to extend the training target of the depth oracle network with multiple ground truth surface points. The proof-of-concept DOnRF that is trained by using depth maps extracted from a NeRF already shows promising results for these surfaces. However, both, using multiple depth values for ground truth depth training, and eliminating the round trip through a full NeRF when there is no ground truth depth, are obvious next steps for DOnRF.

Furthermore, we only focused on static content in our evaluation. For dynamic content, related research has already shown that depth-aware losses can be introduced to achieve more consistency [XHKK20; LNSW21]. Our classified depth sampling strategy could be adapted as a variation of these ideas, allowing for more consistency across dynamic content while staying within a compact neural representation. Another partially orthogonal approach to ours is caching and baking NeRFs to further increase evaluation speeds. Integrating an oracle, especially in combination with dynamic caching, may allow for further increases in rendering efficiency without compromising compactness for streaming.

To our knowledge, DOneRF is the first reliable method to render from a raymarched neural scene representation at interactive frame rates without exhaustive caching, and opens the door for compact high-quality dynamic rendering in real-time. We are confident that such a local sampling strategy will be essential for real-time neural rendering going forward.

References

- [ANA*20] ANDERSSON, PONTUS, NILSSON, JIM, AKENINE-MÖLLER, TOMAS, et al. “FLIP: A Difference Evaluator for Alternating Images”. *Proc. ACM Comput. Graph. Interact. Tech.* 3.2 (Aug. 2020) 7.
- [BBJ*20] BOSS, MARK, BRAUN, RAPHAEL, JAMPANI, VARUN, et al. “NeRD: Neural Reflectance Decomposition from Image Collections”. (2020). arXiv: 2012.03918 [cs.CV] 2.
- [BFO*20] BROXTON, MICHAEL, FLYNN, JOHN, OVERBECK, RYAN, et al. “Immersive Light Field Video with a Layered Mesh Representation”. *ACM Trans. Graph.* 39.4 (July 2020) 2.
- [BMSR20] BEMANA, MOJTABA, MYSZKOWSKI, KAROL, SEIDEL, HANS-PETER, and RITSCHEL, TOBIAS. “X-Fields: Implicit Neural View-, Light- and Time-Image Interpolation”. *ACM Trans. Graph.* 39.6 (Nov. 2020) 2.
- [BXS*20] BI, SAI, XU, ZEXIANG, SRINIVASAN, PRATUL, et al. “Neural Reflectance Fields for Appearance Acquisition”. (2020). arXiv: 2008.03824 [cs.CV] 2.
- [CMK*21] CHAN, ERIC, MONTEIRO, MARCO, KELLNHOFER, PETR, et al. “pi-GAN: Periodic Implicit Generative Adversarial Networks for 3D-Aware Image Synthesis”. *CVPR*. 2021 2.
- [DY21] DELLAERT, FRANK and YEN-CHEN, LIN. *Neural Volume Rendering: NeRF And Beyond*. 2021. arXiv: 2101.05204 [cs.CV] 2.
- [DZY*20] DU, YILUN, ZHANG, YINAN, YU, HONG-XING, et al. “Neural Radiance Flow for 4D View Synthesis and Video Processing”. *arXiv preprint arXiv:2012.09790* (2020) 2.
- [FBD*19] FLYNN, J., BROXTON, M., DEBEVEC, P., et al. “DeepView: View Synthesis With Learned Gradient Descent”. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, 2362–2371 2.
- [GKJ*21] GARBIN, STEPHAN J., KOWALSKI, MAREK, JOHNSON, MATTHEW, et al. “FastNeRF: High-Fidelity Neural Rendering at 200FPS”. (2021). arXiv: 2103.10380 [cs.CV] 2.
- [GSL*20] GAO, CHEN, SHIH, YICHANG, LAI, WEI-SHENG, et al. “Portrait Neural Radiance Fields from a Single Image”. *arXiv preprint arXiv:2012.05903* (2020) 2.
- [GTZN21] GAFNI, GUY, THIES, JUSTUS, ZOLLMÖFER, MICHAEL, and NIESSNER, MATTHIAS. “Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction”. (June 2021), 8649–8658 2.
- [HPP*18] HEDMAN, PETER, PHILIP, JULIEN, PRICE, TRUE, et al. “Deep Blending for Free-Viewpoint Image-Based Rendering”. *ACM Trans. Graph.* 37.6 (Dec. 2018) 2.
- [HSM*21] HEDMAN, PETER, SRINIVASAN, PRATUL P., MILDENHALL, BEN, et al. “Baking Neural Radiance Fields for Real-Time View Synthesis”. *arXiv* (2021) 2.
- [KB15] KINGMA, Diederik P. and BA, JIMMY. “Adam: A Method for Stochastic Optimization”. *ICLR (Poster)*. 2015 7.
- [LGZ*20] LIU, LINGJIE, GU, JIATAO, ZAW LIN, KYAW, et al. “Neural sparse voxel fields”. *Advances in Neural Information Processing Systems* 33 (2020) 2, 3, 7, 14.
- [LMW21] LINDELL, DAVID B., MARTEL, JULIEN N.P., and WETZSTEIN, GORDON. “AutoInt: Automatic Integration for Fast Neural Volume Rendering”. (2021) 2.
- [LNSW21] LI, ZHENGQI, NIKLAUS, SIMON, SNAVELY, NOAH, and WANG, OLIVER. “Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes”. (June 2021), 6498–6508 2, 11.
- [LSS*19] LOMBARDI, STEPHEN, SIMON, TOMAS, SARAGIH, JASON, et al. “Neural Volumes: Learning Dynamic Renderable Volumes from Images”. *ACM Trans. Graph.* 38.4 (July 2019), 65:1–65:14 2.
- [LSS*21] LOMBARDI, STEPHEN, SIMON, TOMAS, SCHWARTZ, GABRIEL, et al. *Mixture of Volumetric Primitives for Efficient Neural Rendering*. 2021. arXiv: 2103.01954 [cs.GR] 2.
- [MRS*21] MARTIN-BRULLA, RICARDO, RADWAN, NOHA, SAJJADI, MEHDI S. M., et al. “NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, 7210–7219 2.
- [MSO*19] MILDENHALL, BEN, SRINIVASAN, PRATUL P., ORTIZ-CAYON, RODRIGO, et al. “Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines”. *ACM Transactions on Graphics (TOG)* (2019) 2, 7, 14.
- [MST*20] MILDENHALL, BEN, SRINIVASAN, PRATUL P., TANCIK, MATTHEW, et al. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”. *ECCV*. 2020 1–4, 6, 7, 13, 14.
- [NG21] NIEMEYER, MICHAEL and GEIGER, ANDREAS. “GIRAFFE: Representing Scenes As Compositional Generative Neural Feature Fields”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, 11453–11464 2.
- [OMT*21] OST, JULIAN, MANNAN, FAHIM, THUEREY, NILS, et al. “Neural Scene Graphs for Dynamic Scenes”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, 2856–2865 2.
- [PCPM21] PUMAROLA, ALBERT, CORONA, ENRIC, PONS-MOLL, GERARD, and MORENO-NOGUER, FRANCESC. “D-NeRF: Neural Radiance Fields for Dynamic Scenes”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, 10318–10327 2.
- [PFS*19] PARK, J. J., FLORENCE, P., STRAUB, J., et al. “DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation”. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, 165–174 2.
- [PSB*20] PARK, KEUNHONG, SINHA, UTKARSH, BARRON, JONATHAN T., et al. “Deformable Neural Radiance Fields”. *arXiv preprint arXiv:2011.12948* (2020) 2.
- [RJY*20] REBAIN, DANIEL, JIANG, WEI, YAZDANI, SOROOSH, et al. “DeRF: Decomposed Radiance Fields”. *arXiv e-prints*, arXiv:2011.12490 (Nov. 2020), arXiv:2011.12490. arXiv: 2011.12490 [cs.CV] 2.
- [RJY*21] REBAIN, DANIEL, JIANG, WEI, YAZDANI, SOROOSH, et al. “DeRF: Decomposed Radiance Fields”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, 14153–14161 2, 3.
- [RPLG21] REISER, CHRISTIAN, PENG, SONGYOU, LIAO, YIYI, and GEIGER, ANDREAS. “KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs”. (2021). arXiv: 2103.13744 [cs.CV] 2.
- [SDZ*21] SRINIVASAN, PRATUL P., DENG, BOYANG, ZHANG, XIUMING, et al. “NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, 7495–7504 2.
- [SLNG20] SCHWARZ, KATJA, LIAO, YIYI, NIEMEYER, MICHAEL, and GEIGER, ANDREAS. “GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis”. *Advances in Neural Information Processing Systems (NeurIPS)*. 2020 2.
- [SMB*20] SITZMANN, VINCENT, MARTEL, JULIEN N.P., BERGMAN, ALEXANDER W., et al. “Implicit Neural Representations with Periodic Activation Functions”. *Proc. NeurIPS*. 2020 1, 2.
- [STB*19] SRINIVASAN, P. P., TUCKER, R., BARRON, J. T., et al. “Pushing the Boundaries of View Extrapolation With Multiplane Images”. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, 175–184 2.

- [STH*19] SITZMANN, VINCENT, THIES, JUSTUS, HEIDE, FELIX, et al. “Deepvoxels: Learning persistent 3d feature embeddings”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, 2437–2446 2.
- [SZW19] SITZMANN, VINCENT, ZOLLMÖFER, MICHAEL, and WETZSTEIN, GORDON. “Scene representation networks: Continuous 3d-structure-aware neural scene representations”. *Advances in Neural Information Processing Systems*. 2019, 1121–1132 1, 2.
- [TSM*20] TANCIK, MATTHEW, SRINIVASAN, PRATUL P., MILDENHALL, BEN, et al. “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains”. *NeurIPS* (2020) 2.
- [TY20] TREVITHICK, ALEX and YANG, BO. “GRF: Learning a General Radiance Field for 3D Scene Representation and Rendering”. *arXiv:2010.04595*. 2020 2.
- [WPYS21] WIZADWONGSA, SUTTISAK, PHONGTHAWEE, PAKKAPON, YENPHRAPHAI, JIRAPHON, and SUWAJANAKORN, SUPASORN. “NeX: Real-time View Synthesis with Neural Basis Expansion”. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021 2, 7, 14.
- [XHKK20] XIAN, WENQI, HUANG, JIA-BIN, KOPF, JOHANNES, and KIM, CHANGIL. “Space-time Neural Irradiance Fields for Free-Viewpoint Video”. *arXiv preprint arXiv:2011.12950* (2020) 2, 11.
- [YLSL21] YUAN, WENTAO, LV, ZHAOYANG, SCHMIDT, TANNER, and LOVEGROVE, STEVEN. “STaR: Self-Supervised Tracking and Reconstruction of Rigid Objects in Motion With Neural Rendering”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, 13144–13152 2.
- [YLT*21] YU, ALEX, LI, RUILONG, TANCIK, MATTHEW, et al. “PlenOctrees for Real-time Rendering of Neural Radiance Fields”. *arXiv*. 2021 2.
- [YYTK21] YU, ALEX, YE, VICKIE, TANCIK, MATTHEW, and KANAZAWA, ANGJOO. “pixelNeRF: Neural Radiance Fields From One or Few Images”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, 4578–4587 2.
- [ZRSK20] ZHANG, KAI, RIEGLER, GERNOT, SNAVELY, NOAH, and KOLTUN, VLADLEN. “NeRF++: Analyzing and Improving Neural Radiance Fields”. (2020). *arXiv: 2010.07492 [cs.CV]* 2.
- [ZTF*18] ZHOU, TINGHUI, TUCKER, RICHARD, FLYNN, JOHN, et al. “Stereo Magnification: Learning View Synthesis Using Multiplane Images”. *ACM Trans. Graph.* 37.4 (July 2018) 2.

Appendix A: Additional Results: Efficient Neural Sampling

Tables 7, 8, 9 and 10 show per-scene results for varying numbers of samples per ray, using the sampling methods described in Section 4.

Appendix B: Additional Results: Sampling Oracle Network

Tables 3, 4, 5 and 6 show per-scene results for varying numbers of samples per ray and various sampling oracle configurations. Section 5 details the various depth oracle configurations.

Appendix C: Additional Evaluation Setup: Datasets

Bulldozer (by ”Heinzelnisse“ <https://www.blendswap.comblend/11490>) [view cell size: $x = 1, y = 1, z = 1$] shows a close view of a toy bulldozer made of building bricks with fine, high-frequency details and mostly diffuse shading. This scene differs from the dataset used in the original NeRF [MST*20], as we re-rendered it to better fit our view cell methodology.

Forest (by Robin Tran <https://cloud.blender.org/p/gallery/5fbdb186ec57d586577c57417>) [$x = 2, y = 2, z = 2$]

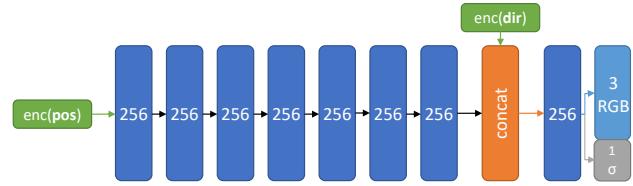


Figure 14: For evaluation, we use the same MLP architecture for both NeRF and DONeRF: 8 layers with 256 hidden units each, with a single skip connection to forward the encoded directions to the last layer. For the depth oracle network, we do not use a skip connection. The inputs and outputs are described in Section 5.1.

shows a vast field of cel-shaded high-frequency foliage and trees, with a person in the foreground, which enforces a good foreground and background representation.

Classroom (by Christophe Seux <https://download.blender.org/demo/test/classroom.zip>) [$x = 0.7, y = 0.7, z = 0.2$] provides a challenging indoors scenario, with thin geometric detail such as the chairs’ legs, as well as very detailed texture work.

San Miguel (by Guillermo M. Leal Llaguno <https://casual-effects.com/g3d/data10/index.html>)

[$x = 1, y = 1, z = 0.4$] provides a test on a proven computer graphics test scene with difficult high-frequency transparent materials (the leaves of the trees) as well as complex disocclusions.

Pavillon (by Hamza Cheggour / ”eMirage“ https://download.blender.org/demo/test/pabellon_barcelona_v1.scene_.zip) [$x = 1, y = 1, z = 1$] contains complex view-dependent effects such as reflection and refraction in the pool as well as completely transparent objects.

Barbershop (by ”Blender Animation Studio“ https://svn.blender.org/svnroot/bf-blender/trunk/lib/benchmarks/cycles/barbershop_interior/)

[$x = 1.5, y = 1.5, z = 0.4$] is a small indoors scene containing realistic global illumination and challenging mirrors.

Appendix D: Additional Evaluation Setup: DONeRF

We use a slightly adapted version of the network architecture of NeRF for both networks of DONeRF and NeRF in our evaluation (Figure 14): slightly less capacity with 8 layers at 256 hidden units and only a single skip connection for the encoded directions. The depth oracle network neither uses skip connections nor encoding for the positions and directions, but still uses ray unification and non-linear sampling (see Section 5).

Appendix E: Additional Evaluation Setup: Baselines

For each baseline method, we converted our datasets to use their required format, *i.e.*, we provide ground truth poses and undistorted ground truth images for each method. We count the storage by summing all required network weights, checkpoints or images that are necessary to render novel views. FLOP per pixel are counted either by profiling with NVIDIA Nsight Compute or directly from the neural network evaluation.

NeRF [MST*20] We use an open source PyTorch port of NeRF (<https://github.com/yenchenlin/nerf-pytorch>) in our evaluation setup. For consistency, we use the same network architecture as used for evaluating DOneRF (Figure 14) and use 64 coarse and 128 additional fine samples (resulting in $64 + 64 + 128 = 256$ total network evaluations) as in the original NeRF [MST*20]. We train NeRF at 300000 iterations total, taking 2048 samples from 2 images per iteration.

NSVF [LGZ*20] For NSVF, we use the authors’ open source code (<https://github.com/facebookresearch/NSVF>) and perform a grid search over the initial voxel size. As NSVF varies greatly in terms of quality and performance based on the initial voxel size, we choose 3 representative variants by choosing the best version within three scenarios: *NSVF-large* is selected by choosing the best quality, ignoring performance and memory constraints, *NSVF-medium* is aimed at the best quality-performance tradeoff, and *NSVF-small* aims at the lowest memory requirements. We train each NSVF for 150000 iterations using 4096 samples per iteration.

LLFF [MSO*19] For LLFF, we use the authors’ available code (<https://github.com/Fyusion/LLFF>), and evaluate the quality metrics by first removing a border of 10 % from each side of the image, as suggested by the authors.

NeX [WPYS21] For NeX, we use the authors’ open source code (<https://github.com/nex-mpi/nex-code/>) and their suggested parameters that fit into the memory budget of 11 GB GPU RAM, *i.e.*, 256 hidden units for their main MLP, 6 layers and 12 sublayers for their MPI. This also provides a more competitive comparison in terms of storage requirements. We use the authors’ real-time web viewer to render images for the evaluation, and train for 300000 iterations with 4096 samples per iteration. Furthermore, we include an additional baseline for NeX, where we do not precompute the radiance and neural basis functions into an MPI, and instead query them from the MLPs directly during inference, to show the potential upper limit in quality. Note that this NeX-MLP variant has slightly higher storage requirements than NeX, as it contains the base color MPI in 32-bit floating point format.

Table 3: Ablation results for various depth oracle configurations for the Bulldozer scene. Please refer to Section 5 of the main paper for a detailed explanation on these depth oracle configurations.

Bulldozer	PSNR \uparrow	FLIP \downarrow
Method	N = 2 N = 4 N = 8 N = 16 N = 2 N = 4 N = 8 N = 16	
<i>SD</i>	25.187 25.977 26.684 27.347 0.096 0.083 0.074 0.067	
<i>SD unified</i>	26.714 27.242 27.736 28.196 0.079 0.071 0.066 0.061	
<i>K-1 Z-1 I-1</i>	26.936 28.776 30.476 31.926 0.074 0.061 0.052 0.047	
<i>K-5 Z-1 I-1</i>	30.435 33.241 34.893 36.019 0.059 0.046 0.041 0.038	
<i>K-5 Z-1 I-128</i>	31.442 33.840 35.248 36.203 0.053 0.044 0.040 0.037	
<i>K-5 Z-5 I-128</i>	31.351 34.253 36.071 37.119 0.054 0.043 0.038 0.036	
<i>K-9 Z-1 I-1</i>	30.511 33.506 35.441 36.561 0.058 0.046 0.040 0.037	
<i>K-9 Z-1 I-128</i>	30.837 33.931 35.645 36.661 0.056 0.044 0.039 0.037	
<i>K-9 Z-9 I-128</i>	29.497 32.654 35.156 37.132 0.064 0.050 0.042 0.036	

Table 4: Ablation results for various depth oracle configurations for the Forest scene. Please refer to Section 5 of the main paper for a detailed explanation on these depth oracle configurations.

Forest	PSNR \uparrow	FLIP \downarrow
Method	N = 2 N = 4 N = 8 N = 16 N = 2 N = 4 N = 8 N = 16	
<i>SD</i>	29.884 30.640 31.832 33.212 0.067 0.062 0.057 0.053	
<i>SD unified</i>	30.768 31.608 32.822 33.876 0.063 0.058 0.054 0.050	
<i>K-1 Z-1 I-1</i>	29.261 30.933 32.425 33.501 0.069 0.061 0.056 0.052	
<i>K-5 Z-1 I-1</i>	30.262 32.535 34.057 34.978 0.064 0.056 0.051 0.050	
<i>K-5 Z-1 I-128</i>	32.188 34.327 35.119 35.591 0.056 0.051 0.049 0.048	
<i>K-5 Z-5 I-128</i>	32.395 34.177 35.385 36.308 0.056 0.051 0.047 0.045	
<i>K-9 Z-1 I-1</i>	30.617 33.203 34.920 35.648 0.063 0.054 0.049 0.048	
<i>K-9 Z-1 I-128</i>	31.136 33.861 35.298 35.869 0.061 0.052 0.049 0.047	
<i>K-9 Z-9 I-128</i>	30.558 32.941 34.452 35.830 0.064 0.054 0.050 0.046	

Table 5: Ablation results for various depth oracle configurations for the Classroom scene. Please refer to Section 5 of the main paper for a detailed explanation on these depth oracle configurations.

Classroom	PSNR \uparrow	FLIP \downarrow
Method	N = 2 N = 4 N = 8 N = 16 N = 2 N = 4 N = 8 N = 16	
<i>SD</i>	26.832 27.403 28.205 29.601 0.103 0.095 0.086 0.076	
<i>SD unified</i>	27.732 28.315 29.233 30.351 0.091 0.086 0.078 0.071	
<i>K-1 Z-1 I-1</i>	27.449 28.882 29.587 31.733 0.092 0.081 0.079 0.068	
<i>K-5 Z-1 I-1</i>	28.697 30.506 30.823 32.998 0.083 0.072 0.074 0.063	
<i>K-5 Z-1 I-128</i>	29.965 31.882 33.236 33.581 0.073 0.069 0.061 0.061	
<i>K-5 Z-5 I-128</i>	30.749 32.244 33.482 34.348 0.070 0.067 0.060 0.058	
<i>K-9 Z-1 I-1</i>	29.387 31.640 32.681 33.836 0.078 0.067 0.066 0.059	
<i>K-9 Z-1 I-128</i>	30.006 32.200 33.619 33.984 0.073 0.068 0.058 0.059	
<i>K-9 Z-9 I-128</i>	29.698 31.574 32.656 34.029 0.081 0.069 0.062 0.059	

Table 6: Ablation results for various depth oracle configurations for the San Miguel scene. Please refer to Section 5 of the main paper for a detailed explanation on these depth oracle configurations.

San Miguel	PSNR \uparrow	FLIP \downarrow
Method	N = 2 N = 4 N = 8 N = 16 N = 2 N = 4 N = 8 N = 16	
<i>SD</i>	25.314 25.699 26.283 27.163 0.097 0.092 0.086 0.081	
<i>SD unified</i>	25.142 25.487 26.078 26.925 0.099 0.094 0.088 0.082	
<i>K-1 Z-1 I-1</i>	26.080 26.977 27.970 28.746 0.092 0.082 0.075 0.070	
<i>K-5 Z-1 I-1</i>	26.406 27.580 28.547 29.364 0.086 0.078 0.071 0.066	
<i>K-5 Z-1 I-128</i>	27.125 27.986 28.992 29.739 0.079 0.073 0.068 0.064	
<i>K-5 Z-5 I-128</i>	27.105 27.980 28.906 29.781 0.079 0.074 0.068 0.064	
<i>K-9 Z-1 I-1</i>	26.505 27.808 29.056 29.815 0.086 0.076 0.068 0.065	
<i>K-9 Z-1 I-128</i>	26.875 28.200 29.257 29.893 0.082 0.073 0.066 0.064	
<i>K-9 Z-9 I-128</i>	26.330 27.571 28.484 29.431 0.088 0.078 0.071 0.066	

Table 7: Ablation results for various sampling methods for the Bulldozer scene. Please refer to Section 4 of the main paper for a detailed explanation on these sampling strategies.

Bulldozer	uniform		logarithmic		log+warp		NDC		uniform local		logarithmic local		log+warp local		NDC local		
	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	
Number of samples N	2								27.930	0.062	27.774	0.063	27.755	0.064	25.158	0.099	
	4	16.662	0.276	16.562	0.276	16.834	0.275	11.929	0.408	28.071	0.061	28.038	0.060	28.034	0.060	26.147	0.086
	8	19.720	0.204	19.458	0.209	19.637	0.206	14.814	0.314	28.115	0.061	28.093	0.059	28.129	0.059	26.358	0.083
	16	22.790	0.146	22.683	0.147	22.769	0.145	18.384	0.229	28.152	0.061	28.169	0.059	28.192	0.059	26.483	0.084
	32	26.551	0.096	26.233	0.098	26.458	0.095	22.027	0.156	28.208	0.060	28.223	0.058	28.269	0.059	26.891	0.081
	64	30.660	0.064	30.472	0.064	30.817	0.062	25.381	0.106	28.366	0.060	28.939	0.055	28.975	0.056	29.424	0.068
	128	34.362	0.047	34.261	0.047	34.651	0.045	29.265	0.075	30.169	0.052	32.551	0.047	32.640	0.047	29.172	0.071

Table 8: Ablation results for various sampling methods for the Forest scene. Please refer to Section 4 of the main paper for a detailed explanation on these sampling strategies.

Forest	uniform		logarithmic		log+warp		NDC		uniform local		logarithmic local		log+warp local		NDC local		
	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	
Number of samples N	2								24.126	0.156	29.666	0.079	30.867	0.071	22.108	0.182	
	4	14.794	0.525	21.062	0.227	21.185	0.220	15.431	0.474	27.384	0.102	28.513	0.089	30.760	0.075	22.683	0.170
	8	20.172	0.242	22.494	0.171	23.129	0.160	18.651	0.305	27.164	0.104	27.909	0.100	29.799	0.087	22.801	0.169
	16	20.282	0.239	24.430	0.144	25.363	0.131	21.185	0.221	26.724	0.109	26.762	0.117	29.070	0.091	23.417	0.164
	32	20.824	0.229	25.294	0.139	25.995	0.129	23.403	0.166	22.786	0.179	25.492	0.133	27.754	0.104	23.581	0.164
	64	22.498	0.178	27.357	0.113	26.951	0.118	25.962	0.122	22.137	0.194	24.996	0.150	26.242	0.129	22.961	0.170
	128	23.082	0.167	26.536	0.125	28.809	0.100	29.291	0.097	22.416	0.184	27.067	0.123	29.552	0.097	23.696	0.164

Table 9: Ablation results for various sampling methods for the Classroom scene. Please refer to Section 4 of the main paper for a detailed explanation on these sampling strategies.

Classroom	uniform		logarithmic		log+warp		NDC		uniform local		logarithmic local		log+warp local		NDC local		
	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	
Number of samples N	2								33.389	0.054	33.668	0.054	33.458	0.055	32.157	0.060	
	4	21.135	0.194	21.804	0.204	21.497	0.211	18.920	0.275	33.994	0.052	33.964	0.053	33.815	0.054	33.033	0.057
	8	24.357	0.141	25.263	0.130	25.689	0.120	22.690	0.180	33.889	0.053	33.949	0.054	33.948	0.054	33.259	0.056
	16	26.519	0.111	27.313	0.105	27.838	0.094	25.689	0.123	33.346	0.055	33.688	0.054	33.952	0.053	33.325	0.056
	32	28.882	0.085	29.678	0.082	30.435	0.072	28.342	0.088	33.495	0.056	33.786	0.055	33.883	0.054	33.587	0.057
	64	31.322	0.068	32.252	0.064	32.822	0.060	31.307	0.065	33.660	0.057	34.182	0.055	34.452	0.053	33.586	0.058
	128	33.113	0.059	33.781	0.057	34.527	0.053	33.823	0.055	33.574	0.058	33.727	0.057	34.510	0.055	33.674	0.058

Table 10: Ablation results for various sampling methods for the San Miguel scene. Please refer to Section 4 of the main paper for a detailed explanation on these sampling strategies.

San Miguel	uniform		logarithmic		log+warp		NDC		uniform local		logarithmic local		log+warp local		NDC local		
	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	
Number of samples N	2								28.689	0.072	28.658	0.071	28.919	0.068	28.286	0.073	
	4	20.763	0.213	21.442	0.197	21.662	0.190	17.843	0.291	28.824	0.071	28.705	0.071	28.861	0.070	28.627	0.072
	8	22.607	0.171	23.610	0.146	24.031	0.137	21.549	0.190	28.810	0.072	28.632	0.072	28.879	0.072	28.913	0.071
	16	23.634	0.149	24.588	0.128	25.250	0.114	23.711	0.138	28.675	0.075	28.595	0.074	28.935	0.072	29.096	0.071
	32	24.795	0.126	25.369	0.114	26.429	0.098	25.702	0.103	28.654	0.076	28.533	0.077	29.027	0.072	29.330	0.071
	64	26.137	0.105	26.283	0.102	27.607	0.086	27.783	0.080	28.476	0.079	28.460	0.079	28.976	0.074	29.566	0.070
	128	27.227	0.092	27.391	0.089	28.825	0.075	29.453	0.069	28.498	0.079	28.132	0.079	28.869	0.075	29.582	0.069