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

THOMAS NEFF, Graz University of Technology, Austria
PASCAL STADLBAUER, Graz University of Technology, Austria
MATHIAS PARGER, Graz University of Technology, Austria
ANDREAS KURZ, Graz University of Technology, Austria
CHAKRAVARTY R. ALLA CHAITANYA, Facebook Reality Labs, USA
ANTON KAPLANYAN, Facebook Reality Labs, USA
MARKUS STEINBERGER, Graz University of Technology, Austria

(a) *San Miguel* (26.903dB) (b) *Pavillon* (31.032dB) (c) *Classroom* (32.244dB) (d) *Bulldozer* (31.50dB) (e) *Forest* (28.029dB)

Fig. 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 $24.3\,\mathrm{ms}$ 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 192 (64 + 128) samples per pixel. While reducing the execution and training time by up to $48\times$, we also achieve better quality across all scenes (NeRF achieves an average PSNR of 30.04 dB vs our 31.62 dB).

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 neural networks. However, one major limitation preventing the use of NeRF in interactive and real-time rendering applications is the prohibitive computational cost of excessive network evaluations along each view ray, requiring dozens of petaFLOPS when aiming for real-time rendering on consumer hardware.

In this work, we take a step towards bringing neural representations closer to practical rendering of synthetic content in interactive and 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 local samples are placed around surfaces in the scene. To this end, we propose a depth oracle network, which 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, a dual network design with a depth oracle network as a first step and a locally sampled shading network for ray accumulation. With our design, we reduce the inference costs by up to 48x compared to NeRF. Using an off-the-shelf inference API in combination with simple compute kernels, we are the first to render raymarching-based neural representations at interactive frame rates (15 frames per second at 800x800) on a single GPU. At the same time, since we focus on the important parts of the scene around surfaces, we achieve equal or better quality compared to NeRF to enable interactive high-quality rendering.

Author's addresses: Thomas Neff, thomas.neff@icg.tugraz.at; Pascal Stadlbauer, pascal.stadlbauer@icg.tugraz.at; Mathias Parger, mathias.parger@icg.tugraz.at; Andreas Kurz, andreas.kurz@icg.tugraz.at; Chakravarty R. Alla Chaitanya, chakravarty.alla@gmail.com; Anton Kaplanyan, kaplanyan@fb.com; Markus Steinberger, steinberger@icg.tugraz.at.

1 INTRODUCTION

Real-time rendering of photorealistic scenes with complex effects is a demanding problem in interactive applications, such as games. At the same time, consumer machine learning accelerators are widespread from desktop GPUs to mobile virtual reality (VR) headsets and mobile phones, making evaluation of neural networks fast and power-efficient. Recent advances in implicit neural scene representations [Sitzmann et al. 2020, 2019b], especially in the direction of neural radiance fields (NeRFs) [Mildenhall et al. 2020], impressively showed that machine learning can also be used for encoding and rendering of high-quality scenes and lighting effects. NeRFs combine a multilayer perceptron (MLP) network with traditional volume rendering methods to faithfully render high-quality content.

In this way, neural representations can reproduce high-resolution light fields containing detailed objects and complex illumination with practical storage requirements. In this work, we focus on making neural representations more practical for interactive and real-time rendering applications. Particularly, our ultimate goal is to enable *synthetic content* with movie-quality effects in interactive rendering. For example, it can be a compact and efficient backdrop representation for static distant parts of a game scene, or in VR

and augmented reality (AR) scenarios, where a simple environment map does not offer required parallax to create a high quality stereo stimulus.

While NeRF-like methods show significant potential for compact high-quality object and scene representations, they are too expensive to evaluate in real-time. The original NeRF method [Mildenhall et al. 2020] uses about 1 000 000 floating point parameters in two networks and requires 192 network evaluations for a single pixel. Real-time rendering of such a representation onto a VR headset at 1440×1600 pixel per eye with 90 Hz would require 37 petaFLOPS (192 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 image generation.

In this paper, we explore how a NeRF-like representation can be designed for efficient future real-time rendering. We make the following contributions:

- We propose a dual network design to reduce evaluation costs of neural rendering of synthetic content. The first oracle network estimates depth values along rays. The second shading network places a low number of samples guided by the output of the first network. This setup achieves equal quality to the original NeRFs method with only 2–8 network evaluations per pixel, effectively reducing the computational cost by 24–98×.
- We present a robust depth oracle network that utilizes the reference depth readily available in synthetic content and employs discretization of depth values along the ray and learns to solve a classification task rather than directly estimating depth values. This way, our depth oracle efficiently distributes shading samples, especially around depth discontinuities. We introduce spatial filtering of the target data and unify the input rays to improve the efficiency and robustness of depth estimates in complex scenes.
- We introduce a non-linear transformation of depth and sampling to efficiently handle large, open scenes. Moreover, we show that sampling of the shading network should happen in a warped space to better capture high frequencies in the foreground and reduce noise in the background.
- Combining our efforts, we demonstrate high-quality interactive neural rendering of large synthetic scenes for interactive applications.

Our work addresses an important practical problem towards adopting neural representations for real-time rendering. Besides improving the rendering efficiency, our design is also significantly faster to train, as fewer samples are needed per ray. Note that we focus on static synthetic scenes and consider dynamic scenes and animations orthogonal to our work.

2 RELATED WORK

Image-based novel view synthesis. Under the umbrella of image-based rendering (IBR) [Shum and Kang 2000; Zhang and Chen 2004], steady progress has been made over the years to further improve the realism of novel views generated only from few example images. Most recently, techniques using multi-plane images (MPIs) [Flynn et al. 2019; Zhou et al. 2018] managed to achieve impressive results by blending together image layers that are spaced along the camera viewing direction, using a neural network to generate the MPI light field representation. In this way, the generation of novel views requires only a simple alpha blending step over all layers. While this is efficient, the main drawback of MPIs is that they typically only allow for small viewing regions [Srinivasan et al. 2019], in addition to requiring densely sampled input images to assemble a high-quality MPI. By introducing systematic input image locations, blending between multiple generated MPIs [Mildenhall et al. 2019], and using spherical image layers [Broxton et al. 2020], the potential field of view can be increased. Alternatively, given enough images captured across different view, time or illumination conditions, it is possible to train an implicit mapping between these domains onto their input images, allowing for fast and efficient interpolation [Bemana et al. 2020].

Implicit neural scene representations. While explicit neural representations based on voxels [Sitzmann et al. 2019a], MPIs [Flynn et al. 2019; Zhou et al. 2018] or proxy geometry [Hedman et al. 2018] enable fast novel view generation, they are fundamentally limited by the internal resolution of their representation, which can result in blurred output for high-frequency content. Furthermore, the required memory for such representations scales poorly with spatial resolution, and is therefore not applicable for large-scale scenes. To circumvent this issue, recent work has explored the potential of implicit neural scene representations [Park et al. 2019; Sitzmann et al. 2019b] to directly infer outputs from a continuous input space, such as ray origins and directions. Scene representation networks (SRNs) [Sitzmann et al. 2019b] directly map 3D world coordinates to a feature representation of the scene, and use a learned raymarcher to accumulate rays for the final RGB output. Similarly, neural volumes [Lombardi et al. 2019] use raymarching to accumulate rays in a learned, warped, volumetric representation, making it possible to reconstruct high-quality images from a multiview capture. By using periodic activation functions [Sitzmann et al. 2020], it is even possible to accurately learn gradients and laplacians, as well as further improve the quality of the output compared to widely used ReLU activations.

Neural Radiance Fields. Opening a whole new subdomain of research, Mildenhall et al. [2020] introduced NeRF, which replaced the learned raymarching step from SRN with a fixed, differentiable ray marching step using a constant number of samples per ray. Compared to neural volumes, this ray marching step does not traverse through a fixed-size representation, but instead all ray samples are fed into an MLP, followed by an accumulation step to generate the RGB output. Even though this is a simpler pipeline overall, they achieve state-of-the-art results for novel view generation, and showcase additional features such as the ability to reconstruct depth maps. A key ingredient of NeRF being able to reconstruct high-frequency details is *positional encoding*, where each ray sample is transformed into a periodic, higher dimensional space using a sine-cosine basis. While the number of frequencies used for the periodic basis is an additional hyperparameter that needs to be tuned per scene, Tancik et al. [2020] have since shown that positional encoding can be generalized to randomized Fourier kernels.

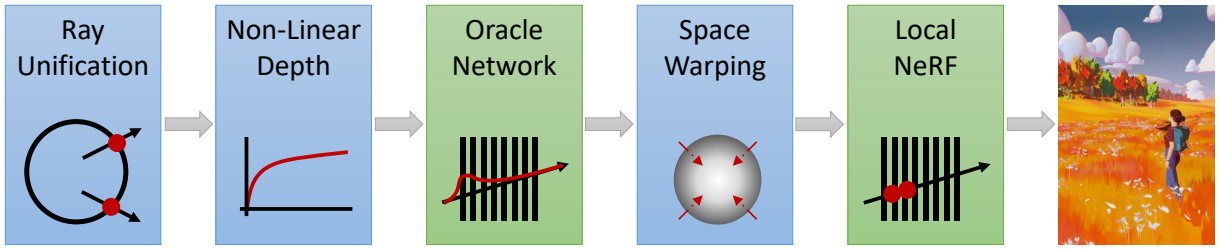


Fig. 2. To enable efficient rendering of large-scale neural representations, we propose a five stage pipeline: (1) ray descriptions are unified within a view cell, (2) depth is considered in a non-linear space, (3) an oracle network estimates the importance of sample positions along each view ray, (4) those sample positions are warped spherically towards the view cell, and (5) radiance information is generated using another MLP that accumulates only a few samples along each ray.

The combination of simplicity and impressive results inspired a large amount of modifications and adaptations to the original NeRF framework, sometimes being referred to as the *NeRF explosion* [Dellaert and Yen-Chen 2021]. Since its inception, NeRFs have been modified to capture dynamic free-viewpoint video [Du et al. 2020; Li et al. 2020; Park et al. 2020; Pumarola et al. 2020; Xian et al. 2020], generate photorealistic human avatars [Gafni et al. 2020; Gao et al. 2020], perform relighting on captured scenes [Bi et al. 2020; Boss et al. 2020; Martin-Brualla et al. 2020; Srinivasan et al. 2020], conditionally encode shape and appearance via latent codes [Chan et al. 2020; Schwarz et al. 2020; Trevithick and Yang 2020; Yu et al. 2020] and to compose scenes of multiple objects [Niemeyer and Geiger 2020; Ost et al. 2020; Yuan et al. 2021]. The sheer amount of recent publications underlines the importance and potential of NeRF-like representations.

While all of these NeRF variants show impressive qualitative results, the large number of required samples leads to long training times and makes NeRFs unsuitable for real-time applications. As a result, several recent publications incrementally improve the training scheme of NeRF, further increasing the quality and runtime efficiency. Decomposed radiance fields (DeRFs) [Rebain et al. 2020] decompose the scene with a learnable voronoi decomposition to train multiple NeRFs for each cell, therefore avoiding the diminishing returns of larger network sizes and increasing run-time efficiency due to more optimal cache locality. Combining the explicit voxel-based representations of previous work with NeRF-style raymarching, neural sparse voxel fields (NSVFs) [Liu et al. 2020] introduced a self-pruning sparse voxel octree structure, where each ray sample includes additional information from a tri-linearly interpolated embedding of voxel vertices. This not only improves quality, but can also reduce the number of samples per ray, as only non-empty voxels are considered for raymarching. However, for scenes with high depth complexity, the ratio of foreground to background can be significant, resulting in longer evaluation times if multiple voxels are intersected by a single ray. In comparison to our work, NSVF still requires more samples, an additional data structure, and may lead to variable evaluation times depending on the current view, which can be a disadvantage for real-time applications.

Recently, Lindell et al. [2020] showed that partial integrals along a ray can also be learned, which can reduce the number of evaluations along a ray when raymarching through a NeRF. However, their achieved image quality is lowered significantly at moderate performance gains.

In our work, we show the most significant performance gains compared to previous work, while also increasing image quality compared to NeRF [Mildenhall et al. 2020]. We do so in the context of complete neural representations, *i.e.*, by only using two neural networks. We show that our approach is also usable for large-scale scenes, where the original NeRF struggles, and techniques that require additional data structures, like NSVF [Liu et al. 2020], are difficult to establish as scene bounding boxes are overly large.

3 EFFICIENT NEURAL RENDERING USING DEPTH ORACLE NETWORKS

In order to achieve real-time movie-quality rendering of compact neural representations for generated content, we adapt the MLP-based neural raymarching scheme of NeRF [Mildenhall et al. 2020]. We introduce DONeRF, a compact local sampling strategy that enables raymarching-based neural representations to only consider important samples around surface regions.

DONeRF consists of two networks: A depth oracle MLP that predicts optimal sample locations along the view ray (*Sampling Oracle Network*, Section 5), and another shading MLP which predicts RGBA output and uses NeRF-like raymarching accumulation. The combination of these two networks enables efficient skipping of empty space, while inheriting the compactness and resolution independence of MLP-based neural representations such as NeRF.

DONeRF consists of a five-stage pipeline (Figure 2): We transform the input rays to a unified space, given information about a supported *view cell* (*Ray Unification*, Section 5.2). We use a classification network as our depth oracle, which outputs filtered depth value likelihoods for each discretized ray segment (*Classified Depth*, Section 5.1). The classification network is trained using modified ground truth depth images, which are typically available for generated content. All depth inputs and outputs are transformed to logarithmic space (*Non-linear Sampling*, Section 4.1), and we place a low amount of samples in the vicinity of the predicted depth (*Local Sampling*, Section 4.2). Finally, we transform the local samples with an inverse square root transform (*Space Warping*, Section 4.1), before applying positional encoding and feeding them into the shading network. By efficiently compressing the depth and radiance information into

two MLPs, we arrive at a compact representation that is suitable for streaming high-quality generated content for VR or games.

For training, we use RGBD input images sampled from a *view cell*. A view cell is defined as an axis aligned bounding box with a center and size in all three dimensions, and defines the maximum allowed range for novel view generation. 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.

4 EFFICIENT NEURAL SAMPLING

In terms of inference performance, NeRF-like neural representations scale most significantly with the number of samples per ray. While there are methods that deploy multiple lower-capacity networks for a moderate quality-speed trade-off [Rebain et al. 2020], 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, following two scenarios:

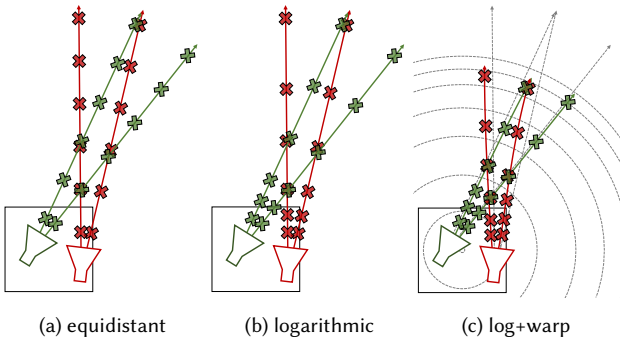
(1) Without any knowledge of the scene, can the sample count be reduced by an alternative sample placement structure without sacrificing quality?
(2) Given information about surfaces in the scene, how can samples be placed selectively to increase performance while keeping high visual quality?

To answer these questions, we run six ablation studies. For all experiments, we assume static geometry and lighting, and split each scene into slightly overlapping view cells. We vary the number of samples per ray N between $[2, 4, 8, 16, 32, 64, 128]$ and only use a single MLP as our representation (without the refinement network proposed in the original NeRF work which dynamically influences sample placement). We conduct our experiments on four versatile scenes, *Bulldozer*, *Forest*, *Classroom* and *San Miguel* (Section 6.1). For *Bulldozer*, *Forest*, and *Classroom*, we keep the positional and directional encoding frequencies at 10 and 4, as suggested by Mildenhall et al. [2020]. For *San Miguel*, we observed a need for more high-frequency details, and use 14 positional frequencies. For more details on the evaluation setup and the evaluated test scenes, please refer to Section 6 (Evaluation). Qualitative example test images for all scenes in all ablation experiments can be found in the supplemental material.

4.1 General Sampling

We first assume that no additional information of a scene except for its size (and thus near and far plane) is available and samples are placed only based on the camera location.

Equidistant Sampling. Following the original NeRF framework, the default strategy is to place samples with equal distance between the near and far plane. We normalize sample positions to ±1 before positional encoding, placing the view cell center in the origin and dividing by d_{max} (the maximum sample distance from the view cell center).



(a) equidistant (b) logarithmic (c) log+warp

Fig. 3. Different sampling approaches visualized: (a) equidistant sampling samples in equal steps between the near and far planes; (b) logarithmic sampling reduces the sample distance for close samples in favor of increasingly spacing out far samples; (c) log+warp pulls the space closer to the view cell center and thus bends rays, making the scene appear smaller to the NeRF. Note that close samples for different rays in the non-warped space are still close in the warped space.

Non-linear Sampling. Since large depth ranges are unsuitable for equidistant sampling in world space, we also investigate non-linear sampling as initially proposed by Mildenhall et al. [2020]. However, instead of exploiting normalized device coordinates and sampling in inverse depth, we use a logarithmic non-linearity to place the samples

$$\tilde{d} = \frac{\log(d+1)}{\log(d_{max}+1)},$$

where d is the distance to the view cell center, and $\tilde{d}$ is the non-linearly transformed sample distance. As before, we normalize by d_{max} before positional encoding.

Space Warping. Early training results for non-linear sampling show that the background typically contains high frequencies that must be dampened by the network while the foreground requires those to achieve high detail. This is not surprising, as real cameras and graphics techniques such as mip-mapping and anti-aliasing average background details as well.

To remedy this issue, we propose a warping of the 3D space towards a view cell-centric space. Starting from the center c of the supported view cell, we warp the entire space using a radial distortion, bringing the background closer to the view cell:

$$\tilde{x} = (x - c) \cdot W(x - c).$$

We transform every ray sample, *i.e.*, we place samples in a view cell-centric space and transform them before positional encoding. While this transformation warps ray samples such that they do not follow a straight line, sample locations in space stay consistent, *i.e.*, samples from different view points landing at the same 3D location are evaluated equally.

Initial experiments showed that using an inverse square root transform achieves the best results:

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

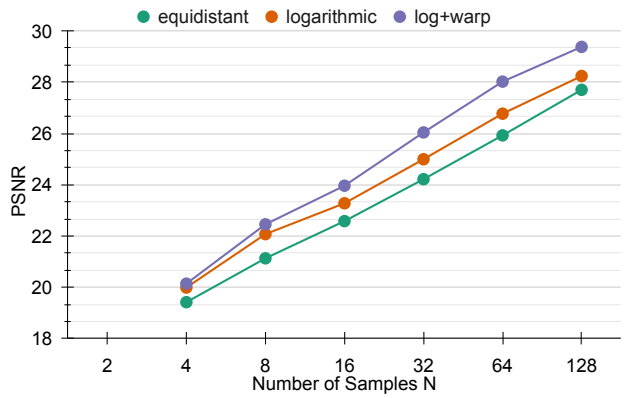


Fig. 4. PSNR results for various numbers of samples per ray N averaged over all scenes. The results show that quality of the the original NeRF setup roughly increases by 1.8 dB for each doubling of N. When using logarithmic sampling (*logarithmic*), our test scenes benefit from a sample distribution that allocates more samples towards the foreground, increasing the PSNR at the same sample count by roughly 0.75 dB. An additional inverse square root warping (*log+warp*) increases quality at the same sample count by 0.75 dB on top of logarithmic sampling.

(a) 4 samples (b) 16 samples (c) 128 samples

Fig. 5. Example views for *Classroom* using the different sampling methods and sample counts. While logarithmic and log+warp increase image quality, low sample counts are insufficient when placing samples in a scene agnostic way.

Note that the use of d_{max} already leads to the desired normalization before feature encoding. See Figure 3 for a visualization of all three placement strategies.

Quantitative results are shown in Figure 4 and example views for *Classroom* in Figure 5. Image quality and number of samples follow

(a) 2 samples (equidistant) (b) 2 samples (logarithmic) (c) 2 samples (log+warp)

Fig. 6. Example views of the test set of the *Classroom* scene using three methods for local sample placement, given a ground truth depth. Even at two samples high quality image results can be achieved. Logarithmic and log+warp still slightly increase quality for local sampling.

an approximately linear relationship for all three techniques (PSNR is in logarithmic space). In all three cases, reducing the number of samples also significantly reduces image quality, and only high sample counts achieve satisfactory quality. However, logarithmic sampling significantly improves the quality and achieves better quality-speed trade-offs for lower sample counts than equidistant sampling. Sampling in a warped space further increases quality. Still, even with log+warp we can only reduce the number of samples approximately by half to achieve identical quality to equidistant sampling.

4.2 Local Sampling

It can be observed that fully-trained NeRFs with large sample counts spend most of their samples in empty space. Given a ground truth for a surface representation, *e.g.*, a depth texture for each view describing the distance of the first opaque surface hit by each ray, it is possible to take a fraction of the outputs of a trained NeRF around the surface depth and still get a result of mostly equal quality.

This inspires the following question: *Given such a ground truth depth texture to place samples, what is the best quality-speed trade-off that can possibly be reached?* To investigate this, we test all three previous methods of sample placement (*equidistant*, *logarithmic*, *log+warp*) locally around the ground truth depth, keeping the step size identical for all sample counts. This should enable the network to only focus on the non-empty regions, without spending network capacity on empty space. The results of this study are shown in Figure 6 and 7.

These results show that, given an ideal sampling oracle, significant gains in quality-speed trade-offs can be achieved when rendering and training raymarching-based neural representations. When comparing the *equidistant* local sampling to the *logarithmic* and *log+warp* sampling, our results suggest that non-linear sampling with additional non-linear space warping towards a view cell achieves the best results overall.

In any case, relying on localized sampling guided by an oracle has multiple advantages: First, raymarching can be sped up significantly, as few samples are sufficient to generate high quality images, see Figure 1. Second, supervised learning for localized samples reduces training time significantly, as the network does not need to learn about empty space. Third, the network can spend its capacity on

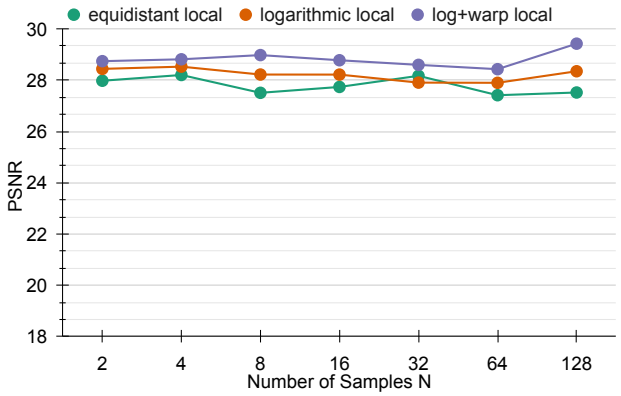


Fig. 7. PSNR results for various numbers of samples per ray N using sample placement around a known ground truth depth averaged over all scenes. The results show that the number of samples does not affect the resulting quality significantly—the networks can even learn well for only $N = 2$ samples per ray. However, we still observe an increase in PSNR of roughly 0.75 dB when using *logarithmic* sampling vs. *equidistant* sampling, and a further increase of roughly 0.9 dB when adding additional space warping (*log+warp*). This suggests that even with a perfect sampling oracle, sample placement is still important.

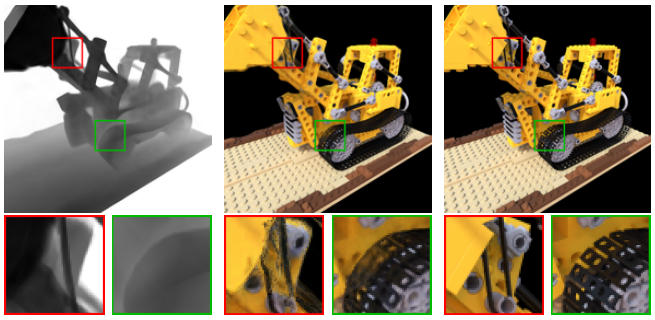
important samples that contribute to the final image, and does not need to waste capacity for empty space.

However, in practice, the 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. Thus local sampling from ground truth depth can be considered a niche scenario, and we therefore target an efficiently compressed sampling representation via an MLP-based sampling oracle that only uses ground truth depth during training.

5 SAMPLING ORACLE NETWORK

In Section 4, we showcased potential improvements to the distribution of ray samples during training and rendering. Locally sampling around a known ground truth surface representation is the most promising. However, surface depth for novel views is generally unknown during inference, unless all potential views in a given view cell are densely captured or a surface representation is available for rendering—both defeating the purpose of having a compact neural representation.

Therefore, we introduce an oracle network to predict ideal sampling locations for a second raymarched shading network. This oracle network takes a ray as input and provides information about sample locations along that ray. Per-ray depth estimation in a view cell is non-trivial, if depth estimates must be consistent across views, especially around depth discontinuities. 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, as shown in Figure 8, leading to significant visual artifacts.



(a) Depth Prediction (b) 3 Local Samples (c) Ground Truth

Fig. 8. A depth oracle network with a single depth output is not able to faithfully predict depth around discontinuities, such as edges (a). As a result, local sampling mixes fore- and background and distorts features (b), which becomes apparent when comparing to the ground truth (c).

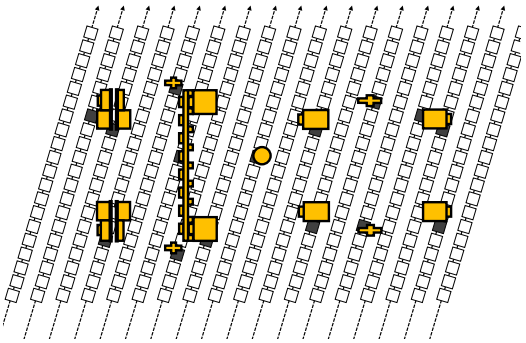


Fig. 9. A horizontal slice through the *Bulldozer* data set at about 2/3 of the height. A classification oracle should report the hit surface location (1 dark, 0 white) along each ray. Note that only the first visible surface intersection is encoded. The low number of active cells and high variability of target classification results between neighboring rays may reduce the quality of the prediction.

To mitigate this issue, we make 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 give the oracle network more freedom; while it must predict sample locations around the hit surface, it can also provide *additional* estimates. Consider two neighboring rays around a depth discontinuity, one hitting the foreground object and one the background object: 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 for both the background and the foreground rays the network is allowed to return the same result, *i.e.*, sampling at the foreground *and* the background depth, the oracle task is easier to learn.

5.1 Classified Depth

Interestingly, a straightforward 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 want to avoid raymarching the oracle network and instead want to evaluate it only once. Moreover, we also want to significantly reduce the number of samples of the shading network . To this end, 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.

Discretizing the ground truth depth value of each ray leads to a one-hot encoding as target that can be trained using the common binary cross-entropy (BCE) loss, see Figure 9.

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 across depth. To this end, 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 of neighboring rays with a lower contribution:

$$
\begin{aligned}
c_{x,y,z} &= \max\left(1 - \frac{\sqrt{(x+i)^2 + (y+j)^2}}{\sqrt{2} \cdot \lfloor K/2 \rfloor}, 0\right), \qquad (1)\\
z &= \left\lfloor \frac{d_{x+i,y+j}}{N_z} \right\rfloor \\
\forall x &= [-\lfloor K/2 \rfloor \cdots \lfloor K/2 \rfloor], \\
\forall y &= [-\lfloor K/2 \rfloor \cdots \lfloor K/2 \rfloor],
\end{aligned}
$$

where $c_{x,y,z}$ is the target for a ray going through the pixel x, y and quantization step z, N_z is the number of quantization steps along the ray, K is the filter size, and $d_{x,y}$ is the depth value at pixel x, y. For example, using a 5×5 filter, rays at a distance of $[3, 2, 1, 0]$ pixels contribute $[0, \approx 0.30, \approx 0.65, 1]$ to the output, respectively.

For multiple depth values with the same discretization result, we only take the maximum result (contribution from the closest ray) to ensure a valid classification target as well as that the highest value is placed at the actual surface.

From a classification point of view, filtering increases the number of false positives compared to the target surface depth. However, for the subsequent raymarching step, it effectively decreases the overall number of false negatives by ensuring that important regions of interest around the surface depth are not missed. This becomes apparent when considering that rays exist in a continuous space, *i.e.*, the network does not need to learn hard boundaries between rays at depth discontinuities. Translating a higher false positive rate to raymarching, we increase the likelihoods for regions that do not need any samples. Thus, overall, we need to place more samples along rays to hit the "right" sample location, 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 wrong 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.

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

$$
\begin{aligned}
D_F[k] = &\sum_{i=1}^{\lfloor Z/2 \rfloor + 1} D_I[k + i - \lfloor Z/2 \rfloor - 1] \frac{i}{\lfloor Z/2 \rfloor + 1} \\
&+ \sum_{i=\lfloor Z/2 \rfloor}^{1} D_I[k + i - \lfloor Z/2 \rfloor + 1] \frac{i}{\lfloor Z/2 \rfloor + 1} \\
D_O[k] = &\operatorname{clamp}\left(D_F[k], 0, 1\right), \qquad (2)
\end{aligned}
$$

where $D_I[k]$ describes the discretized input depth segment at position k after filtering in image-space, $D_F[k]$ is the filtered result along the depth dimension and $D_O[k]$ is the filtered depth after clamping it between 0 and 1.

Figure 11 shows an example output for a fully filtered target. This filter increases the positively classified cells further, enabling the oracle network to more conservatively learn the depth of surfaces.

According to our observations, for scenes with fine structures and large view cells, both smoothing steps are essential. See Figure 10 for the depth oracle target output for different filtering techniques.

Finally, to translate the classification values to sample locations, we use the same approach as NeRF when going from the coarse to the fine network by sampling the inverse transform of the piecewise-constant probability density function (PDF) [Mildenhall et al. 2020]; see Figure 12.

5.2 Ray Unification and Oracle Input

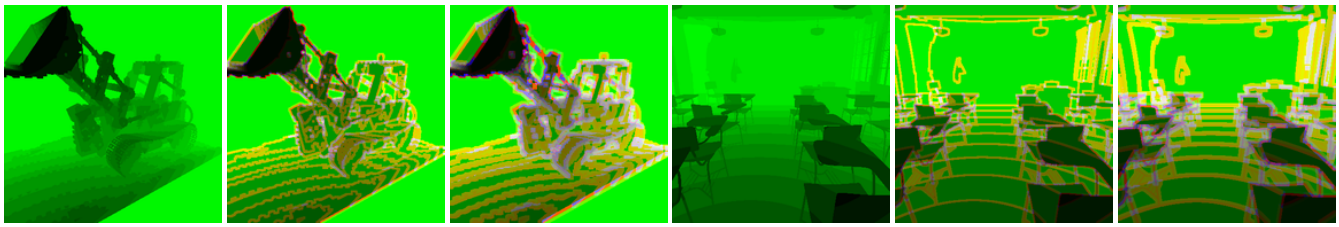
To take the burden of disambiguating between rays originating from different starting locations from the oracle network, we map every ray starting location onto a sphere circumscribed around the view cell, see Figure 13.

Following the intuition of asking the oracle network to provide multiple (discretized) outputs along the ray, we additionally supply the same number of 3D positions regularly sampled along the ray—the centers of the discretized depth ranges for depth classification. These 3D positions are provided at once without increasing the network capacity, therefore keeping the evaluation cost similar. We experimented with higher dimensional positional and Fourier encoding for the input parameters, but did not reliably increase output quality. This may be due to sheer number of input parameters, which makes it more difficult for the network to learn.

As a logarithmic transformation on the sample positions along the ray increases quality for raymarching-based neural representations (Section 4.1), we use the same transformation on the classification steps, using smaller steps close to the camera and larger steps in the background.

5.3 Optimal Filter Sizes and Depth Oracle Inputs

To evaluate the optimal filter sizes for our neighborhood pixel filter K (Equation 1), our depth label smoothing filter Z (Equation 2) and the number of 3D input positions along the ray ($I = [1, 128]$), we conduct an additional ablation study across a subset of those three parameters, using the classified depth oracle network as the input for a locally raymarched shading network, similar to the experiment in Section 4.2.



(a) *Bulldozer* $K = 1$ (b) *Bulldozer* $K = 5$ (c) *Bulldozer* $K = 9$ (d) *Classroom* $K = 1$ (e) *Classroom* $K = 5$ (f) *Classroom* $K = 9$

Fig. 10. Visualization of the classified depth target output for 16 classes along each ray and the largest classes mapped to G, R, B and encoded with their depth value, *i.e.*, a pure green value indicates there is only a single class with the brightness indicating the depth. A gray value indicates that the three highest classes are at similar depth, and different colors in proximity indicate that the largest classes are moved in depth. Small features are smoothed to neighboring pixels with increasing filter sizes (1-5-9).

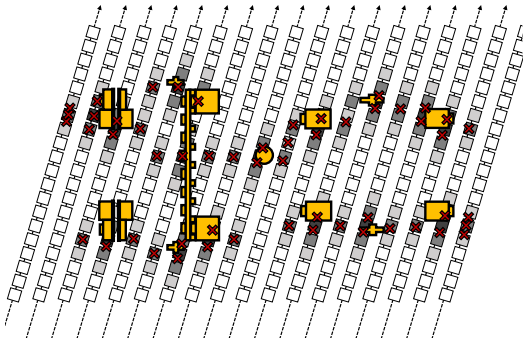


Fig. 11. Filtering the target depth classification in both image dimensions as well as along depth smooths the classification target. An oracle producing such an output still results in high quality sample locations (red), with additional samples placed in free space. A smooth target is easier to learn as labels vary with lower frequency.

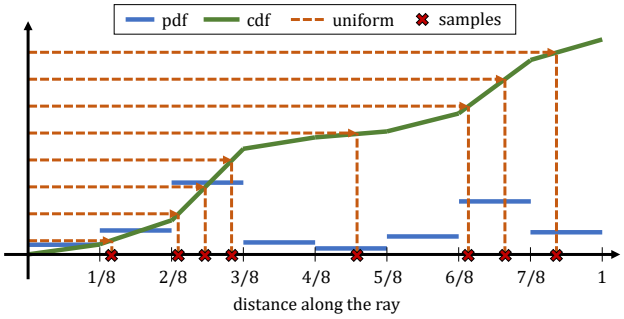


Fig. 12. To place samples along the ray (red ×), we treat the classified depth output as a piecewise-constant PDF (blue) and translate it into a cumulative distribution function (CDF, green). Sampling the resulting CDF at equidistant steps (orange), concentrates the samples (red ×) around the ray regions with a large classified depth output.

We vary the number of samples per ray N between $[2, 4, 8, 16]$ samples and test on the *Bulldozer*, *Forest*, *Classroom* and *San Miguel* scenes (Section 6.1), evaluating the overall quality on the test sets. Both the depth oracle network and the shading network contain 8

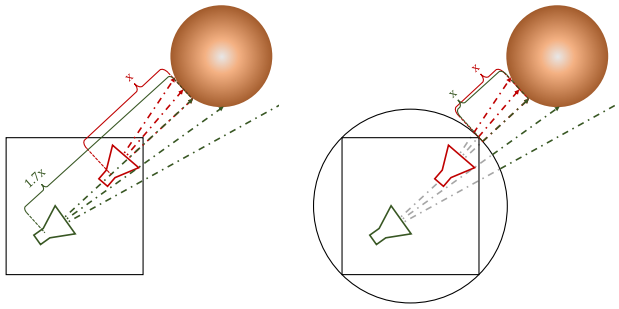


Fig. 13. Ray unification maps ray starting points to a sphere surrounding the view cell. Without ray unification (a) the same ray is encoded differently and the oracle needs to predict different depth values; after ray unification (b) identical rays have identical depth values.

hidden layers with 256 hidden units each. We first train the depth oracle network for 300 000 iterations, before training the shading network for 300 000 iterations using the predictions from the depth oracle. Additionally, to illustrate the importance of our depth classification, we compare against a combination of a simpler depth oracle which only predicts a single depth value to guide the shading network with (*SD unified*) and without (*SD*) ray unification. Our final metric is the resulting quality of the RGB output of the shading network, which should improve when given better sampling positions by the depth oracle.

The results shown in Table 1 indicate that filtering both along the image dimensions and along depth leads to improvements in PSNR, without additional inference cost, even for very low sample counts. Furthermore, including additional sample positions along the ray seems to improve quality even further. For additional per-scene results, please refer to the supplemental material.

6 EVALUATION

We compare DONeRF against the baseline of NeRF [Mildenhall et al. 2020]. We mainly analyze the ability to extract novel high-quality views from neural representations of generated content, given challenging scenarios such as high-frequency geometry and large depth ranges. We evaluate this by computing the PSNR and FLIP [Andersson et al. 2020] values against ground truth renderings.

(a) *SD unified* (8 samples) ≈ 7.4 MFLOP per pixel

(b) *log+warp* NeRF (64 + 128 samples) ≈ 211.4 MFLOP per pixel

(c) DONeRF (2 samples) ≈ 2.7 MFLOP per pixel

(d) DONeRF (4 samples) ≈ 4.4 MFLOP per pixel

(e) DONeRF (8 samples) ≈ 7.7 MFLOP per pixel

Fig. 14. Example views comparing DONeRF against a variant that uses only a single estimated depth value for sample placement, given unified ray inputs (*SD unified*), and the original NeRF coarse + fine network architecture. Even at only 2 samples (a 78× improvement in MFLOP per pixel), DONeRF achieves equal or better quality. Furthermore, while the single-depth oracle network also demonstrates a noticeable improvement in quality at lower sample counts, it is not able to reconstruct thin, difficult geometry, such as the chairs' legs in *Classroom*, the details in the grass in *Forest* or the leaves of the trees in *San Miguel*.

Table 1. PSNR and FLIP results averaged over all scenes, evaluating the neighborhood pixel filter size (K-X), the depth label smoothing filter size (Z-X) and the number of inputs for the depth oracle network (I-X) over the number of samples per ray N that are used for raymarching. Additionally, we also compare against a network that only predicts a single depth value (*SD*), as well as a single depth value with unified input (*SD unified*), which both show significantly lower quality compared to the filtered depth classification oracle.

	PSNR ↑				FLIP ↓			
Method	N = 2	N = 4	N = 8	N = 16	N = 2	N = 4	N = 8	N = 16
SD	23.970	24.449	25.260	26.182	0.135	0.126	0.114	0.104
SD unified	25.437	25.910	26.638	27.326	0.113	0.107	0.101	0.094
K-1 Z-1 I-1	26.525	27.685	28.578	29.722	0.101	0.092	0.086	0.081
K-5 Z-1 I-1	27.086	28.470	29.462	30.771	0.098	0.087	0.082	0.075
K-5 Z-1 I-128	27.893	29.247	30.631	31.303	0.089	0.084	0.077	0.073
K-5 Z-5 I-128	**28.198**	**29.670**	30.957	**31.589**	**0.087**	**0.080**	0.074	0.072
K-9 Z-1 I-1	27.198	28.698	29.992	31.105	0.096	0.084	0.085	0.075
K-9 Z-1 I-128	27.747	29.382	**30.937**	31.574	0.091	0.083	**0.074**	0.072
K-9 Z-9 I-128	27.501	29.118	30.363	31.510	0.095	0.084	0.077	**0.072**

Please refer to Section 4 and Section 5 for ablation studies of the main components of DONeRF, and please see the supplemental material for more detailed results per scene.

For our prototype real-time implementation of DONeRF we use a combination of TensorRT and CUDA. Ray unification, space warping, feature generation and raymarching runs in custom CUDA kernels with one thread per ray. Both networks are evaluated using TensorRT in half floating point precision. Thus, 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 interactive frame rates at medium resolutions on an NVIDIA RTX 3090. Ray unification and first feature generation, the oracle network, space warping and the second feature generation, the shading network, and final color generation take $1.43\,\text{ms}$, $12.51\,\text{ms}$, $9.52\,\text{ms}$, $37.60\,\text{ms}$, and $0.18\,\text{ms}$ respectively for a 800×800 image and 2 samples per ray, *i.e.*, about 15 fps. 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: $37.60\,\text{ms}$, $74.37\,\text{ms}$, and $140.49\,\text{ms}$ for 2, 4, and 8 samples, respectively.

6.1 Datasets

We collected a set of scenes that showcase both fine, high-frequency details as well as large depth ranges to validate that DONeRF is applicable for a wide variety of scenes. All datasets are rendered with Blender [1], using their Cycles path tracing engine to render 300 images for each scene (taking approximately 20 minutes per image), which we split into train/validation/test sets at a $70\%/10\%/20\%$ ratio. Examples of the ground truth images using the same views as the teaser (Figure 1) are shown in the supplemental material. For each scene, a view cell has been defined that showcases a wide range of movement and rotation to uncover potential issues with disocclusions and challenging geometry, while not intersecting geometry within the scene.

Bulldozer[2] [view cell size: $x = 1, y = 1, z = 1$] shows a close view of a toy bulldozer made of building bricks with lots of fine, high-frequency details and mostly diffuse shading.

Forest[3] [view cell size: $x = 2, y = 2, z = 2$] shows a vast field of cel-shaded foliage and trees, with a person in the foreground looking into the distance. This dataset enforces a good foreground and background representation, and the foliage provides challenging high-frequency geometric content.

Classroom[4] [view cell size: $x = 0.7, y = 0.7, z = 0.2$] provides a challenging indoors scenario, with thin geometric detail such as the chairs' legs, as well as very detailed texture work.

San Miguel[5] [view cell size: $x = 1, y = 1, z = 0.4$] showcases both difficult high-frequency transparent materials (the leaves of the trees) as well as complex disocclusions caused by the large pillars, and provides a test on a proven computer graphics test scene.

Pavillon[6] [view cell size: $x = 1, y = 1, z = 1$] contains complex view-dependent effects such as reflection and refraction in the pool as well as completely transparent objects, testing the ability of each neural representation to correctly reproduce those effects.

6.2 Evaluation Setup

For each scene, we compare our best performing DONeRF, using a neighborhood filter size of 5, a depth smoothing filter size of 5 and 128 unified inputs to the depth oracle network (Section 5.3), against NeRF [Mildenhall et al. 2020], using their coarse+fine sampling approach at $64 + 128$ samples. As discussed in Section 4, we transform each sample logarithmically, and warp it towards the view cell. We train each network for 300 000 iterations and use the checkpoint with the lowest validation loss for testing. As our optimizer, we use Adam [Kingma and Ba 2015] with a learning rate of 0.0001. We use a varying number of samples per ray depending on available GPU memory for each network, typically around 1024 samples for 2 images per batch. For RGB outputs, we use standard MSE loss, while DONeRF's depth classification uses the binary cross entropy loss. DONeRF's depth oracle network is pretrained for 300 000 iterations on the ground truth depth, after which the weights are locked and the shading network is trained for 300 000 iterations. All images are downsampled to a resolution of 400×400 to speed up training. All MLPs contain 8 layers with 256 hidden units each, and all inputs to the shading network are encoded using positional encoding with 10 positional and 4 directional frequencies, except for *San Miguel* and *Pavillon*, where we use 14 positional and 4 directional encoding frequencies.

6.3 Results

The results of our evaluation are summarized in Table 2, and qualitative example views are shown in Figure 14. Compared to the NeRF

[1] https://www.blender.org/

[2] by "Heinzelnisse" https://www.blendswap.com/blend/11490

[3] by Robin Tran https://cloud.blender.org/p/gallery/5fbd186ec57d586577c57417

[4] by Christophe Seux https://download.blender.org/demo/test/classroom.zip

[5] by Guillermo M. Leal Llaguno https://casual-effects.com/g3d/data10/index.html

[6] by "Hamza Cheggour / eMirage" https://download.blender.org/demo/test/pabellon_barcelona_v1.scene_.zip

Table 2. Across all our test scenes, DONeRF shows a significant improvement in quality-speed trade-off, 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.

		Bulldozer		*Forest*		*Classroom*		*San Miguel*		*Pavillon*	
Method	Average MFLOP per pixel	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP	PSNR	FLIP
DONeRF (2 samples)	2.703	29.161	0.075	26.946	0.108	30.749	0.070	25.936	0.097	30.541	0.103
DONeRF (4 samples)	4.355	31.502	0.062	28.029	0.103	32.244	0.067	26.903	0.088	31.032	0.100
DONeRF (8 samples)	7.659	33.046	0.057	29.667	0.094	33.482	0.060	27.632	0.086	31.370	0.098
DONeRF (16 samples)	14.285	**33.960**	**0.055**	**29.897**	**0.093**	**34.348**	**0.058**	**28.150**	**0.084**	31.574	**0.097**
log+warp NeRF (64+128)	211.422	33.341	0.059	26.026	0.135	32.036	0.073	27.041	0.099	**31.754**	0.099

baseline, DONeRF achieves better quality at significantly higher performance across all our test scenes. While DONeRF can still benefit from additional samples along each ray, the local sampling strategy reaches diminishing returns much earlier, and we demonstrate identical quality compared to NeRF already at 2 or 4 samples along the ray, using $78\times/48\times$ fewer FLOP per pixel, respectively.

We also show a qualitative comparison between NeRF, DONeRF and a simplified version of DONeRF that only uses a single depth value for sample placement in Figure 15, Figure 16 and Figure 17.

Especially for *Classroom* with its small geometric features around all chairs and high frequency shadow details the results show that our depth oracle can also handle complex geometric relations. *San Miguel* forms another challenging use case where our depth oracle and local sampling faithfully reconstruct fine geometric structures. Furthermore, especially the regions around masked materials and half transparent foliage can be difficult when significantly reducing sample counts, as both the foreground and background need to be sampled. However, we can see potential for improvement when looking at *Pavillon*—the transparent surfaces of the water and glass windows pose an issue for our depth estimation, as the ground truth depth only contains a single depth value for large surface regions.

(a) DONeRF (4 samples) (b) log+warp NeRF (64 + 128 samples) (c) *SD unified* (4 samples)

Fig. 15. Example views of the test set of *Classroom*. DONeRF manages to accurately represent the fine leg structures and cast shadows on the chairs at only 4 samples per ray (a). NeRF is not able to reconstruct the detailed geometry at (64 + 128) samples (b). Predicting only a single depth for DONeRF's sample placement also fails to guide samples accurately enough (c).

7 CONCLUSION

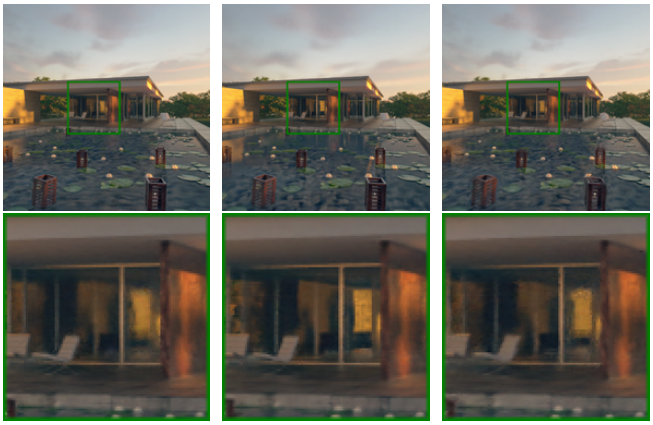
Starting from an evaluation of the sampling strategies of NeRF, we showed that sampling should be done both non-linearly and warped towards a view cell. Using a given ground truth depth map for optimal sample placement, we showed that *local sampling* achieves equal quality with as few as 2 samples per 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 difficult geometric scenarios. By using this classified depth oracle network to guide sample placement for a raymarched shading network, our DONeRF approach achieves equal or better quality compared to NeRF across all scenes, while using as little as 2–8 samples per pixel (instead of 64 + 128 samples). This shows that we can reliably predict good sample positions even for complex and large-scale scenes, where NeRF struggles.

7.1 Limitations and Future Work

Due to the scope, we only focused on static synthetic scenes in our evaluation. Additionally, partially transparent objects in scenes like *San Miguel* and *Pavillon*, such as the water or the glass windows, pose an issue, as only a single ground truth depth value was used for training the sampling oracle network. This leads to samples only being placed around the initial depth value. 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 to correctly sample partially transparent objects. Furthermore, related research has already shown that depth-aware losses can be introduced to achieve more consistency in dynamic videos [Li et al. 2020; Xian et al. 2020] as well as to initialize a voxel representation to constrain the space of potentially visible voxels [Liu et al. 2020]. Our classified depth sampling strategy could be adapted as a variation of these ideas, allowing for more consistency across dynamic content.

(a) DONeRF (4 samples) (b) log+warp NeRF (64 + 128 samples) (c) *SD unified* (4 samples)

Fig. 16. Example views of the test set of *San Miguel*. DONeRF manages to preserve high-frequency detail around the leaves of the tree at only 4 samples per ray (a). While NeRF at (64 + 128) samples also preserves the tree faithfully, the edges are more blurry compared to DONeRF (b). Predicting only a single depth for DONeRF's sample placement completely distorts the tree (c).



(a) DONeRF (4 samples) (b) log+warp NeRF (64 + 128 samples) (c) *SD unified* (4 samples)

Fig. 17. Example views of the test set of *Pavillon*. The transparent surfaces such as water and the glass window lead to DONeRF generating blurrier results (a). NeRF at (64 + 128) handles these materials better due to allocating more samples behind those transparent surfaces (b). Similarly to DONeRF, the single depth prediction blurs the content behind the window (c).

Another open point for investigation is the accuracy of sample placement when ground truth depth is not available, *e.g.*, for real-world content. Our filtered depth discretization should already provide a good starting point for depth estimates that are not fully accurate, as long as the estimated 3D surface positions are consistent across views. To our knowledge, DONeRF is the first reliable method to render from a raymarched neural scene representation at interactive frame rates, and opens the door for high-quality dynamic renderings in real-time. We are confident that such a local sampling strategy will be essential for real-time neural rendering going forward.

REFERENCES

Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. 2020. FLIP: A Difference Evaluator for Alternating Images. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 2, Article 15 (Aug. 2020), 23 pages. https://doi.org/10.1145/3406183

Mojtaba Bemana, Karol Myszkowski, Hans-Peter Seidel, and Tobias Ritschel. 2020. X-Fields: Implicit Neural View-, Light-and Time-Image Interpolation. *arXiv preprint arXiv:2010.00450* (2020).

Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. 2020. Neural Reflectance Fields for Appearance Acquisition. (2020). arXiv:cs.CV/2008.03824

Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P. A. Lensch. 2020. NeRD: Neural Reflectance Decomposition from Image Collections. (2020). arXiv:cs.CV/2012.03918

Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew DuVall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. 2020. Immersive Light Field Video with a Layered Mesh Representation. 39, 4 (2020), 86:1–86:15.

Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. 2020. pi-GAN: Periodic Implicit Generative Adversarial Networks for 3D-Aware Image Synthesis. In *arXiv*.

Frank Dellaert and Lin Yen-Chen. 2021. Neural Volume Rendering: NeRF And Beyond. arXiv:cs.CV/2101.05204

Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. 2020. Neural Radiance Flow for 4D View Synthesis and Video Processing. *arXiv preprint arXiv:2012.09790* (2020).

J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker. 2019. DeepView: View Synthesis With Learned Gradient Descent. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2362–2371. https://doi.org/10.1109/CVPR.2019.00247

Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2020. Dynamic Neural Radiance Fields for Monocular 4D Facial Avatar Reconstruction. (2020). arXiv:cs.CV/2012.03065

Chen Gao, Yichang Shih, Wei-Sheng Lai, Chia-Kai Liang, and Jia-Bin Huang. 2020. Portrait Neural Radiance Fields from a Single Image. *arXiv preprint arXiv:2012.05903* (2020).

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep Blending for Free-viewpoint Image-based Rendering. 37, 6 (2018), 257:1–257:15.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*. http://arxiv.org/abs/1412.6980

Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. 2020. Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. *arXiv preprint arXiv:2011.13084* (2020).

David B Lindell, Julien NP Martel, and Gordon Wetzstein. 2020. AutoInt: Automatic Integration for Fast Neural Volume Rendering. *arXiv preprint arXiv:2012.01714* (2020).

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020. Neural sparse voxel fields. *Advances in Neural Information Processing Systems* 33 (2020).

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. 2019. Neural Volumes: Learning Dynamic Renderable Volumes from Images. *ACM Trans. Graph.* 38, 4, Article 65 (July 2019), 14 pages.

Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. 2020. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. (2020).

Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. 2019. Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines. *ACM Transactions on Graphics (TOG)* (2019).

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

Michael Niemeyer and Andreas Geiger. 2020. GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields. arXiv:cs.CV/2011.12100

Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. 2020. Neural Scene Graphs for Dynamic Scenes. arXiv:cs.CV/2011.10379

J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 165–174.

https://doi.org/10.1109/CVPR.2019.00025

Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. 2020. Deformable Neural Radiance Fields. *arXiv preprint arXiv:2011.12948* (2020).

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2020. D-NeRF: Neural Radiance Fields for Dynamic Scenes. *arXiv preprint arXiv:2011.13961* (2020).

Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. 2020. DeRF: Decomposed Radiance Fields. *arXiv e-prints*, Article arXiv:2011.12490 (Nov. 2020), arXiv:2011.12490 pages. arXiv:cs.CV/2011.12490

Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. 2020. DeRF: Decomposed Radiance Fields. arXiv:cs.CV/2011.12490

Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. 2020. GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Harry Shum and Sing Bing Kang. 2000. Review of image-based rendering techniques. In *Visual Communications and Image Processing 2000*, King N. Ngan, Thomas Sikora, and Ming-Ting Sun (Eds.), Vol. 4067. International Society for Optics and Photonics, SPIE, 2 – 13. https://doi.org/10.1117/12.386541

Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. 2020. Implicit Neural Representations with Periodic Activation Functions. In *Proc. NeurIPS*.

Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. 2019a. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2437–2446.

Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. 2019b. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*. 1121–1132.

Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. 2020. NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis. (2020).

P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely. 2019. Pushing the Boundaries of View Extrapolation With Multiplane Images. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 175–184. https://doi.org/10.1109/CVPR.2019.00026

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *NeurIPS* (2020).

Alex Trevithick and Bo Yang. 2020. GRF: Learning a General Radiance Field for 3D Scene Representation and Rendering. In *arXiv:2010.04595*.

Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. 2020. Space-time Neural Irradiance Fields for Free-Viewpoint Video. *arXiv preprint arXiv:2011.12950* (2020).

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. 2020. pixelNeRF: Neural Radiance Fields from One or Few Images. arXiv:cs.CV/2012.02190

Wentao Yuan, Zhaoyang Lv, Tanner Schmidt, and Steven Lovegrove. 2021. STaR: Self-supervised Tracking and Reconstruction of Rigid Objects in Motion with Neural Rendering. *arXiv preprint arXiv:2101.01602* (2021).

Cha Zhang and Tsuhan Chen. 2004. A survey on image-based rendering—representation, sampling and compression. *Signal Processing: Image Communication* 19, 1 (2004), 1 – 28. https://doi.org/10.1016/j.image.2003.07.001

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. 2018. Stereo Magnification: Learning View Synthesis Using Multiplane Images. *ACM Trans. Graph.* 37, 4, Article 65 (July 2018), 12 pages. https://doi.org/10.1145/3197517.3201323