

# KiloNeuS: Implicit Neural Representations with Real-Time Global Illumination

Stefano Esposito

[stefano.esposito@h-brs.de](mailto:stefano.esposito@h-brs.de)

Hochschule Bonn-Rhein-Sieg

Sankt Augustin, Germany

Daniele Baieri

[baieri@di.uniroma1.it](mailto:baieri@di.uniroma1.it)

La Sapienza Università di Roma

Rome, Italy

Stefan Zellmann

[stefan.zellmann@h-brs.de](mailto:stefan.zellmann@h-brs.de)

Hochschule Bonn-Rhein-Sieg

Sankt Augustin, Germany

André Hinkenjann

[andre.hinkenjann@h-brs.de](mailto:andre.hinkenjann@h-brs.de)

Hochschule Bonn-Rhein-Sieg

Sankt Augustin, Germany

Emanuele Rodolà

[rodola@di.uniroma1.it](mailto:rodola@di.uniroma1.it)

La Sapienza Università di Roma

Rome, Italy

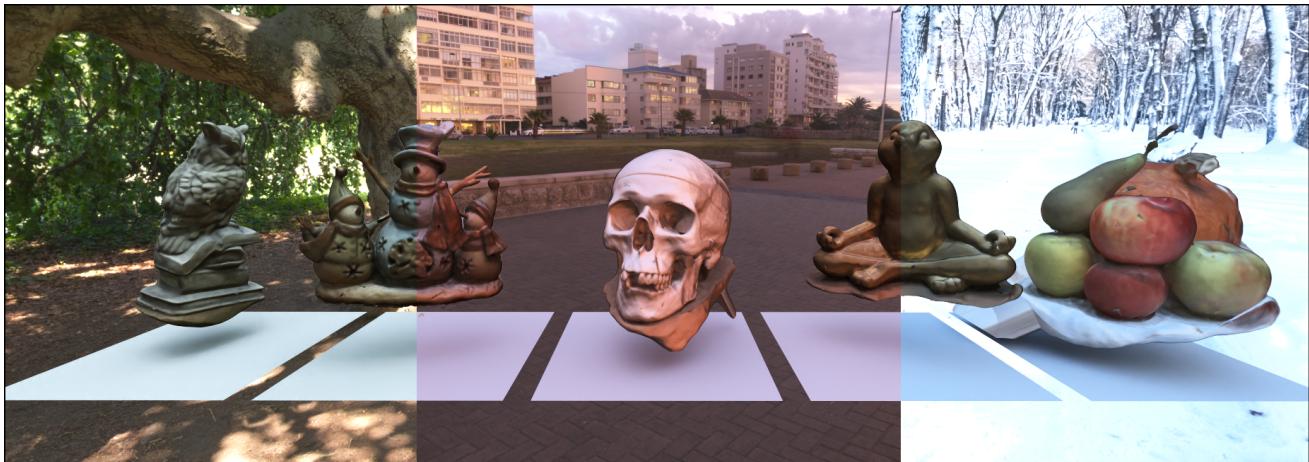


Figure 1: A showcase of KiloNeuS 3D objects under different environment illumination conditions. Our neural representation continuously encodes geometry and appearance of the shapes in thousands of small neural networks, and can be rendered in real-time under global illumination with our cuNPT path-tracer.

## ABSTRACT

The latest trends in inverse rendering techniques for reconstruction use neural networks to learn 3D representations as neural fields. NeRF-based techniques fit multi-layer perceptrons (MLPs) to a set of training images to estimate a radiance field which can then be rendered from any virtual camera by means of volume rendering algorithms. Major drawbacks of these representations are the lack of well-defined surfaces and non-interactive rendering times, as wide and deep MLPs must be queried millions of times per single frame. These limitations have recently been singularly overcome, but managing to accomplish this simultaneously opens up new use cases. We present KiloNeuS, a new neural object representation that can be rendered in path-traced scenes at interactive frame rates. KiloNeuS enables the simulation of realistic light interactions between neural and classic primitives in shared scenes, and it demonstrably performs in real-time with plenty of room for future optimizations and extensions.

## CCS CONCEPTS

- Computing methodologies → Ray tracing; Shape representations; Appearance and texture representations.

## KEYWORDS

Neural representations, path tracing, real-time

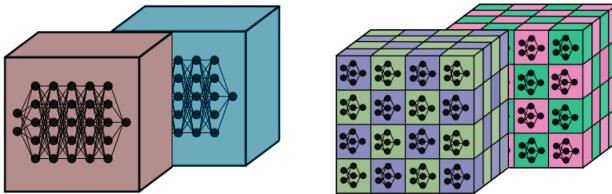
## 1 INTRODUCTION

For a human being, the task of understanding the geometry of a 3D object shown in an image is usually easy. Unfortunately, the same cannot be said for a computer. The loss of one dimension during the projection step makes the problem ill-posed, since infinitely many different geometries could produce the same image. For this reason, estimating the true 3D underlying geometry remains a challenging problem. In recent years there has been rapid progress in inverse rendering techniques reconstructing a scene description from observations using differentiable rendering optimization. The most advanced methods in the field use neural networks to learn 3D representations as neural fields, encoding them in their weights and biases. NeRF [Mildenhall et al. 2020] is a neural object representation that paved the way for a whole range of new techniques indirectly addressing the problem of geometry reconstruction by trying to render scenes from given viewpoints during the optimization process. NeRF-based techniques fit multi-layer perceptrons (MLPs) to a set of training images estimating volumetric fields such

as view-invariant density and view-dependent color (also called *radiance*). These volumes can be sampled and rendered from any virtual camera using differentiable volumetric integration with volumetric rendering algorithms, and are trained by minimizing the difference between rendered and training images.

Simulating light interactions of a shape in a path-traced environment requires having access to both its normals and its surface color. The former, in particular, are used to bounce rays off its surface, continuing their way in the environment. As previously shown in [Oechsle et al. 2021], a vanilla NeRF model defines objects as volume clouds; while this allows photo-realistic synthesis of novel views, it lacks an implicit surface representation enforcing constraints on the learned density field, not guaranteeing smooth, well-defined surface reconstruction and raising a problem in this sense. NeuS [Wang et al. 2021a] solved this problem by decoupling geometry from appearance and deriving the density field from a learnable signed distance function (SDF) that implicitly defines surfaces in the scene, and whose normals can be analytically computed. Another consequential benefit of representing neural objects as SDFs is that rendering can be accelerated as we can resort to sphere tracing [Hart et al. 1993] instead of volumetric ray marching.

Neural object representations do not provide direct access to surface geometry, which is why they are rendered by means of naïve ray marching-derived approaches with dense ray sampling, leading to a steep increase in rendering times. Visualization of these representations can thus be very computationally intensive. An extremely large number of feedforward passes through a wide and deep neural network must be performed for each ray cast. Depending on the target resolution and the number of ray samples to take, a rendering of a vanilla NeRF model, for example, could take minutes to complete, which is well beyond the requirements of real-time rendering. The authors of KiloNeRF [Reiser et al. 2021] demonstrated how real-time rendering can be achieved by utilizing, instead of one single large MLP, thousands of small and lightweight MLPs that uniformly partition the scene into geometrically simpler same-sized blocks. These can be queried at high rates, resulting in a rendering speed-up of three orders of magnitude compared to vanilla NeRF.



**Figure 2: Figures adapted from [Reiser et al. 2021]. Left: NeuS SDF and Color deep MLPs. Right: KiloNeuS SDF and Color grids of lightweight MLPs.**

The combination of the capability of neural representations to capture 3D models from real world objects in an easy, automated fashion, with the possibility to visualize them in real-time path-traced environments allows for a new degree of freedom in accessing and manipulating 3D data. In order to achieve this goal, we propose the following contributions:

- We first define KiloNeuS, a new neural representation designed to be fast to render and easier to relight, overcoming both NeRF’s slow inference times – in a similar fashion to the KiloNeRF approach – and its lack of a well-defined implicit surface, following the NeuS scheme.
- We then present a method for real-time rendering of such representations in scenes with arbitrary global illumination, which may contain one or more bounded objects of both classic (e.g., polygonal mesh) or neural representations. Our GPU-based path-tracer allows for the simulation of realistic light interactions between objects of different types, as demonstrated in Figure 1.
- We conduct a comprehensive evaluation of both our model and path-tracer, in terms of representation quality, features and execution performance, with respect to the results of the latest proposals in the field.

## 2 RELATED WORK

Our work intersects several lines of research, most notably neural rendering, implicit representations (and in particular, neural radiance fields), and neural scene relighting. We review works relating to ours in each of them.

*Neural Rendering.* Inverse rendering refers to a set of techniques estimating intrinsic scene characteristics such as camera, geometry, material and light parameters, given a single image or a set of images using differentiable rendering optimization [Deschaintre et al. 2018, 2019; Henzler et al. 2019; Li et al. 2018; Marschner 1998]. Neural rendering is closely related to inverse rendering; it introduces components learned by deep neural networks into the rendering pipeline in place of predefined physical models or data structures used in classical rendering. A typical neural rendering approach takes as input scene properties and builds a neural scene representation from them, which can then be rendered with different properties to synthesize novel images. The authors of NeuS [Wang et al. 2021a] proposed to decouple geometry from appearance by jointly learning a) a signed distance function (SDF) analytically defining the density field and the surface normals, and b) a color field defining the surface appearance.

*Implicit Representations.* Representing 3D surfaces as continuous functions over space is a recent trend in geometric deep learning. Manipulating continuous surfaces offers new perspectives in geometry processing, and led to significant advances in various complex tasks. Occupancy fields [Mescheder et al. 2019] and signed distance fields [Park et al. 2019] found major applications in 3D shape reconstruction [Atzmon and Lipman 2020; Gropp et al. 2020; Peng et al. 2020; Sitzmann et al. 2020] and 4D shape reconstruction [Chen et al. 2021; Lei and Daniilidis 2022; Niemeyer et al. 2019; Tiwari et al. 2021]. In addition, signed distance functions allow to render 3D scenes by sphere tracing [Hart et al. 1993], a ray-marching heuristic allowing for real-time rendering.

*Neural Radiance Fields.* Neural Radiance fields were proposed in [Mildenhall et al. 2020] and gained much attention from the community because of their unprecedented results on novel view synthesis of 3D scenes, the problem of generating novel camera perspectives of a scene given a fixed set of images. Follow-up works

attempted to improve and extend the original proposal: [Meng et al. 2021; Wang et al. 2021b] proposed unsupervised (i.e. without camera parameters) formulations for NeRF training, while [Park et al. 2021; Pumarola et al. 2020; Tretschk et al. 2020] generalized the setting to non-rigidly deforming scenes.

*Speeding-up Neural Rendering.* Several approaches have been proposed to speed-up the rendering of NeRF-based representations. Some of these focus on dramatically reducing the number of volumetric rendering ray-samples without sacrificing visual quality, using techniques such as empty space skipping and early ray termination. NSVF [Liu et al. 2020] progressively learns an underlying structure of voxel-bounded implicit fields organized in a sparse voxel octree to model local properties in each cell. While claiming to be 10x faster than NeRF, they are still far from meeting real-time requirements and while their method can also be applied for scene editing and composition, they do not allow for light transport simulation. Using a different approach, the authors of AutoInt [Lindell et al. 2021] proposed a technique learning closed-form solutions to integrals by fitting a network representing the antiderivative. While this can be used in a variety of contexts, the authors claim it can speed-up NeRF numerical approximation of the volume rendering integral of an order of magnitude by significantly reducing the number of queries needed per pixel. DONeRF [Neff et al. 2021], on the other hand, jointly trains a depth oracle network to perform only sample points placed locally around surfaces in the scene, reducing the inference costs by up to  $\sim 48x$  compared to NeRF reaching interactive framerates ( $\sim 20$  FPS) and visual quality. On a different note, DeRF [Rebain et al. 2021] proposed a spatial decomposition of a scene based on Voronoi cells, where each part is rendered independently and the final image is composed via Painter’s Algorithm, providing up to  $\sim 3x$  more efficient inference than NeRF. Other approaches like FastNeRF [Garbin et al. 2021] and PlenOctree [Yu et al. 2021] speed-up the volume rendering process by making use of smart *caching* techniques to entirely skip the MLP query and reach impressive framerates ( $> 200$  FPS), at the cost of losing the continuous representation and of a larger memory footprint that grows cubically with the targeted quality.

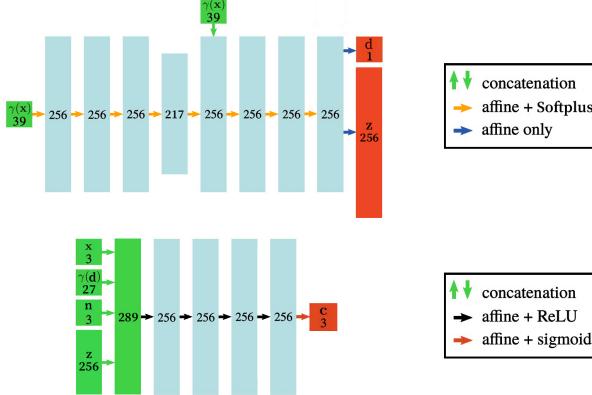


Figure 3: NeuS MLPs architectures. Top: SDF network, bottom: Color network.

On a different note, Light Field Networks [Sitzmann et al. 2021] represent a 3D scene with a 360-degree, four-dimensional light field. Such representation can render a ray with a single MLP evaluation; however the lack of dense depth information make it difficult to include in a path-traced scene. Lastly, the authors of KiloNeRF [Reiser et al. 2021] improved rendering times of vanilla NeRF by proposing to partition space by a uniform coarse grid of small MLPs which can be queried at higher rates than NeRF’s large MLP, achieving real-time performance without caching.

*NeRF relighting.* Although NeRF-based approaches can reproduce view-dependent appearance effects, the radiance of a point in any direction is “baked” into these networks, thus making them unsuitable for relighting. Consequently, others have focused on enabling the rendering of unseen views under novel illuminations, for which additional properties are required to be learned. NeRV [Srinivasan et al. 2021] takes a set of images of a scene illuminated by unconstrained but known lighting and models a reflectance function describing how particles reflect incoming light, to simulate how the light from external sources is attenuated and reflected by the scene. NeRD [Boss et al. 2021] and NeRFactor [Zhang et al. 2021] jointly optimize a model for shape, BRDF, and illumination by minimizing the photometric error over an input image collection of an object, captured under fixed or different lighting.

### 3 METHOD

We now discuss how we achieve real-time rendering of neural object representations under path-traced global illumination, outlining our contributions on top of NeuS [Wang et al. 2021a] and KiloNeRF [Reiser et al. 2021]. Our work can be outlined following two different but tightly linked paths: a) the definition of a lightweight and disentangled neural object representation, able to efficiently encode an implicit surface as a signed distance field, and b) the development of a real-time path-tracer supporting the aforementioned representation, along with classic computer graphics primitives in shared scenes, all in real-time.

#### 3.1 KiloNeuS

This section illustrates the KiloNeuS neural object representation, which combines the reconstructive capabilities of NeuS [Wang et al. 2021a] and KiloNeRF’s ability to render complex models in real-time [Reiser et al. 2021]. KiloNeuS uses thousands of small MLPs organized in two regular grids, which we refer to as KiloSDF and KiloColor, replacing the respective large and wide SDF and Color networks of the NeuS representation, as visualized in Figure 2. Each

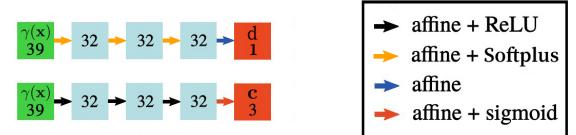
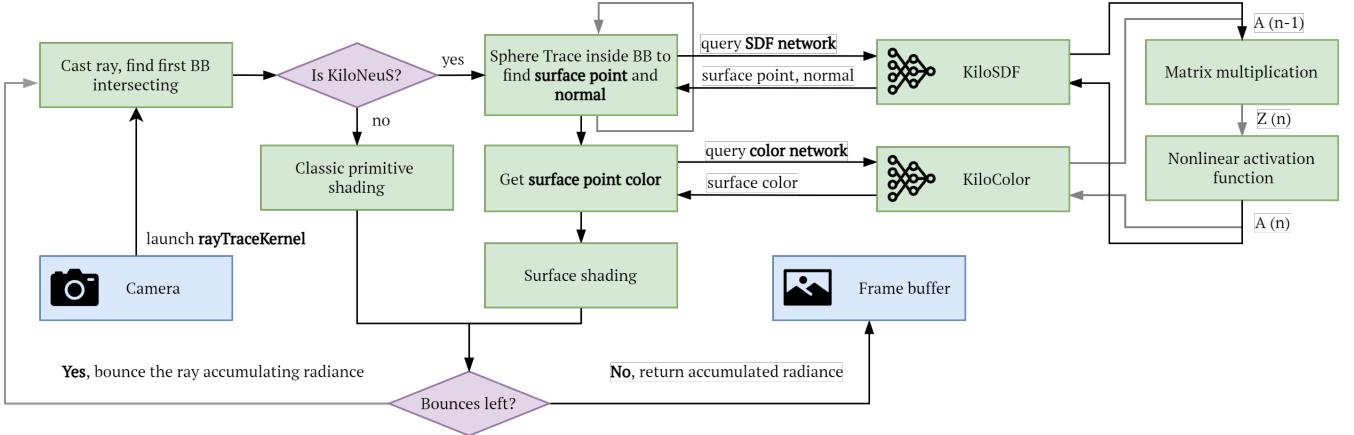


Figure 4: KiloNeuS single SDF and Color MLPs architectures, designed to be a significant simplification of the respective NeuS architectures.



**Figure 5: High-level diagram of the cuNPT GPU thread execution flow focusing on the rendering a KiloNeuS object.**

of those MLPs only represents a small – and therefore simple – fraction of the respective field, allowing the use of few parameters to represent them, thus drastically reducing inference times. Similarly to how KiloNeRF’s MLP architecture is a downscaled version of NeRF’s architecture, KiloSDF and KiloColor (Figure 4) are designed to be a downscaled version of the respective NeuS model architectures (Figure 3). Both networks take as input a 6-frequencies ( $L = 6$ ) positional encoding of the spatial coordinates  $\gamma(\mathbf{x})$ , where  $L$  is a hyperparameter controlling the frequencies used by a fixed set of basis functions. Each scalar component  $p$  of position vector  $\mathbf{x}$  (normalized to lie in  $[-1, 1]$ ) is separately mapped by  $\gamma$  from  $\mathbb{R}$  to a higher dimensional space  $\mathbb{R}^{2L}$ , and results are concatenated to the original coordinates in a new vector in  $\mathbb{R}^{3+6L}$ . The encoding function  $\gamma$ , also known as *Fourier feature mapping* [Tancik et al. 2020], is defined as:

$$\gamma(\mathbf{x}) = (\hat{\gamma}(x_0), \hat{\gamma}(x_1), \dots, \hat{\gamma}(x_n)), \quad (1)$$

where

$$\hat{\gamma}(p) = \left( \sin(2^0 \pi p), \cos(2^0 \pi p), \dots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p) \right) \quad (2)$$

The main differences of the networks lie in the activation functions between layers, Softplus for SDF and ReLU for Color, and in the activation function applied to the output, linear for the former and sigmoid for the latter. As can be observed from Figure 4, KiloColor network discards the feature vector  $\mathbf{z}$ , the vector normal to the surface  $\mathbf{n}$  and view direction  $\mathbf{d}$  of its input. The first two are mainly useful when training the NeuS model from scratch, and help avoiding ambiguities in the optimization process. Whereas for the third, we assume that the surface materials of KiloNeuS objects are Lambertian, to eliminate view-dependency in the lighting. Therefore, we fix it during training as the mirrored direction to the surface normal  $\mathbf{d} = -\mathbf{n}$ . By only learning the surface color viewed under this direction we reduce the influence of specular reflections on the surface of the target object. This is a necessary step in order to enable application inside realistic scenes: objects should react to the illumination of the environment, rather than the illumination

**ALGORITHM 1:** KiloNeuS distillation training procedure, transferring the knowledge from the large NeuS MLPs to the KiloNeuS grids of small MLPs. This is done by optimizing the latter to reconstruct the signal from the former. The small MLP to use is selected according to the spatial region of the query point.

---

**Input:** NeuS teacher  $(c^t, d^t)$ , KiloNeuS students  $(c^s, d^s)$ , epochs  $N$ , KiloNeuS resolution  $k, m = k^3$   
**Output:** KiloNeuS model parameters  $\{\theta_i, \eta_i\}_{i=1}^m$

```

for i = 0 → N do
     $\mathbf{x} \leftarrow \mathcal{U}([-1; 1]^3)$  // Uniform points sampling
     $\mathbf{s}, \mathbf{z} = d^t(\gamma(\mathbf{x}, L = 6))$  // Teacher SDF and feat. vec.
     $\mathbf{n} = \nabla_{\mathbf{x}} s / \|\nabla_{\mathbf{x}} s\|$  // Norm. SDF gradients = normals
     $\mathbf{c} = c^t(\mathbf{x}, \gamma(\mathbf{d}, L = 4), \mathbf{n}, \mathbf{z})$  // Teacher color
     $s' = d^s(\gamma(\mathbf{x}, L = 6); \{\eta_i\}_{i=1}^m)$  // Student SDF
     $c' = c^s(\gamma(\mathbf{x}, L = 6); \{\theta_i\}_{i=1}^m)$  // Student color
     $L = \|s - s'\|^2 + \|\mathbf{c} - \mathbf{c}'\|^2$  // MSE between pred and target
    optimize( $L, \{\theta_i, \eta_i\}_{i=1}^m$ )
end

```

---

used in the training data. The KiloColor network is only queried on surface points, behaving as a continuous texture.

**3.1.1 Distillation training.** The training procedure shall transfer the knowledge from the large MLP to the set of many smaller ones. We achieve this by using *distillation*: we first train a regular NeuS as teacher model, and then its KiloNeuS counterpart such that its network outputs match those of the teacher for any input. We outline the iterative training process of both the KiloSDF and KiloColor models in Algorithm 1.

**Sampling optimization.** We observed that the distillation training learns a coarse approximation of the target function within a very short amount of time, with most of the training time spent learning high-frequency details. From this observation and the fact that the surface and appearance of the object mainly depend on high frequency features near the zero level set of the SDF, we derived a refined sampling procedure in order to speed-up convergence. We



**Figure 6:** Left to right: target reconstructed view, Neu teacher model (view-dependent) and KiloNeuS student model (resolution  $k = 16$ ) rendered with sphere-tracing. Scenes: DTU scans 63, 110, 105 resp.

first train with uniform sampling for a small number of iterations (e.g. 200), then we voxelize the 3D space with resolution  $k$  and only sample from voxels containing SDF values within an absolute threshold (we use  $\tau = 1e-3$ ); occupation of voxels is verified by sampling  $n$  random points within each of them. Note that this procedure has to be carried out only once using the teacher SDF network, so to focus the optimization process on near-surface points encoding fine detail of the surface, rather than distant points for which we are satisfied with a coarser approximation.

### 3.2 cuNPT: CUDA Neural Path-Tracer

Our real-time path-tracer supporting the previously discussed KiloNeuS neural object representation as well as other classic computer graphics primitives has been built upon a publicly available and parallel CUDA-based adaptation [Allen 2021] of Peter Shirley’s *Ray-Tracing in One Weekend* books series [Shirley 2018]. In the following we describe the extensions made to the rendering kernel to support KiloNeuS models, for which a high-level diagram is shown in Figure 5.

**3.2.1 Rendering kernel.** For each ray, the rendering kernel finds the closest intersection point with an object bounding volume, if it exists. If the bounding box does not contain a KiloNeuS object, it follows the execution flow described in [Shirley 2018]; otherwise it starts sphere-tracing in the bounding volume intersection interval, looking for the first ray intersection with the implicitly defined neural surface. If sphere-tracing converges on the SDF 0-level set, the rendering kernel then computes its analytic normal and its color



**Figure 7:** Left to right: normals only, surface color only and path-traced (denoised) cuNPT renderings of multiple KiloNeuS models of resolution  $k = 16$  (4096 MLPs each) in a cornellbox. Scenes: DTU scans 65, 69, 114 resp.

value by querying the color network at the point of intersection. The KiloNeuS neural object representation employed in this case study does not include material properties, which at this time the path-tracer assumes to be Lambertian.

**3.2.2 Neural Network inference.** The neural network inference step is performed by CUDA device functions that take as input the spatial coordinates of a point, encode it, and perform a series of matrix multiplications and apply non-linear activation functions (whose results are always one-dimensional vectors) to output either a SDF value or a color value. Intermediate results are stored in a per-thread register memory of constant size, since global memory access would lead to extreme performance degradation. This memory, depending on the GPU used, usually has a limited capacity. The use of small MLPs with small intermediate results also helps in this regard.

## 4 EXPERIMENTAL RESULTS

In this section, we evaluate KiloNeuS and our cuNPT path-tracer. First, we want to quantify our neural representation’s ability to learn complex 3D surfaces and their view-independent appearance by comparing its result for commonly used metrics against recent related proposals. Then, we evaluate our path-tracer’s performances in rendering such objects under global illumination. We expect our method to reach interactive frame-rates, and to reconstruct surfaces with high fidelity; however, we know that view-independency will cause color differences with test views of reconstructed scenes and that this may have an impact on our results.

**Table 1: Quality metrics evaluation of KiloNeuS over the DTU dataset, compared with competitor evaluations. Methods marked by \* were evaluated on the NeRF-Synthetic dataset, which we consider to be an easier test case than DTU, given the significantly higher PSNR and SSIM achieved by NeRF on the former.**

| Metric  | Method                                   | Avg   |
|---------|--|-------|
| SSIM ↑  | KiloNeuS (MLP grid: $16^3$ )             | 0.890 |
|         | NeuS                                     | 0.820 |
|         | NeRF (DTU)                               | 0.826 |
|         | NeRF*                                    | 0.947 |
|         | KiloNeRF* (MLP grid: $16^3$ )            | 0.950 |
|         | PlenOctree* (cache: $512^3$ )            | 0.958 |
| PSNR ↑  | FastNeRF* (cache: $1024^3$ , factors: 8) | 0.941 |
|         | KiloNeuS (MLP grid: $16^3$ )             | 16.63 |
|         | NeuS                                     | 28.55 |
|         | NeRF (DTU)                               | 29.31 |
|         | NeRF*                                    | 31.01 |
|         | KiloNeRF* (MLP grid: $16^3$ )            | 31.00 |
| LPIPS ↓ | PlenOctree* (cache: $512^3$ )            | 31.71 |
|         | FastNeRF* (cache: $1024^3$ , factors: 8) | 29.97 |
|         | KiloNeuS (MLP grid: $16^3$ )             | 0.215 |
| LPIPS ↓ | NeRF*                                    | 0.081 |
|         | KiloNeRF* (MLP grid: $16^3$ )            | 0.030 |
|         | PlenOctree* (cache: $512^3$ )            | 0.053 |
|         | FastNeRF* (cache: $1024^3$ , factors: 8) | 0.053 |

## 4.1 Quantitative evaluation

The experiments were conducted on some of the DTU datasets [Aanæs et al. 2016] for which the NeuS authors made their pre-trained models publicly available. These were used as teacher models for distillation training and to conduct qualitative and quantitative baseline comparisons. All KiloNeuS models have resolution  $k = 16$  (for a total of 4096 MLPs each) and were trained by distillation from the respective pretrained NeuS teacher model using the *Adam* [Kingma and Ba 2015] optimizer with a learning rate of  $\alpha = 5e-4$ .

We report our results in Table 1 and Figure 8. It is important to observe that the reported quality metrics behave differently from other listed methods, as we represent objects independently of their lighting in training renderings. Consequently, some of the metrics comparing our syntheses and the respective test views are worse than other methods due to the resulting difference in color, as can be seen in Figure 6. This observation is supported by KiloNeuS managing to achieve competitive results for the SSIM metric, which measures the perceived change in structural information, thus giving less importance to such differences in color.

## 4.2 Rendering results

*Image quality.* Figure 7 presents normals, surface color and path-traced renderings of KiloNeuS models. We run our path tracer with 30 maximum samples per sphere-traced ray and apply denoising on

**Table 2: Comparison of performance metrics of our approach to competitors. Competitor data is reported from original publications, averaging over available evaluations. The test data for NeRF, KiloNeRF, FastNeRF and PlenOctree is comparable to ours, being composed of scenes each containing one neural object of modest complexity, while NeuS is evaluated on our same data. All models are rendered with volumetric rendering, except where specified otherwise.**

| Method                                     | Memory (GB) | FPS  |
|--|-------------|------|
| KiloNeuS (MLP grid: $16^3$ , sphere-trace) | 0.7         | 46   |
| NeRF                                       | 0.014       | 0.01 |
| KiloNeRF (MLP grid: $16^3$ )               | 0.4         | 45   |
| NeuS                                       | 0.017       | 0.01 |
| NeuS (sphere-traced)                       | 0.017       | 0.1  |
| FastNeRF (cache: $1024^3$ , factors: 8)    | > 10        | 238  |
| FastNeRF (cache: $768^3$ , factors: 6)     | 4.1         | 714  |
| PlenOctree (cache: $512^3$ )               | 1.9         | 168  |
| PlenOctree (cache: $256^3$ )               | 0.3         | 410  |

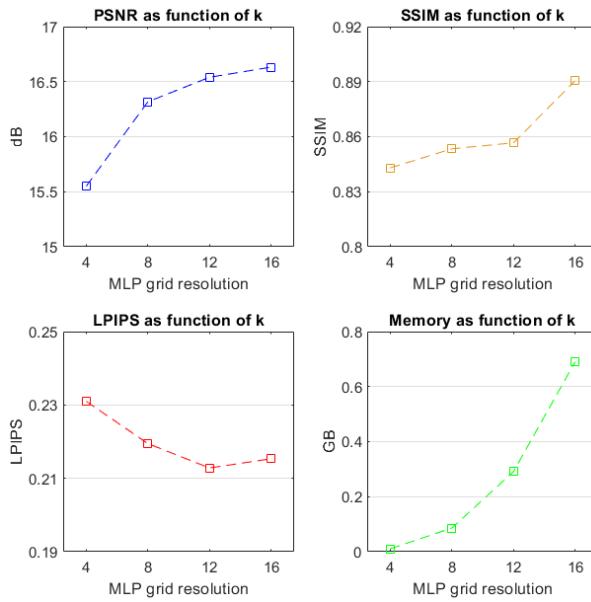
the resulting images for more accurate results. We use Intel Open Image Denoise [Intel 2018] for this purpose.

*Performance.* We tested cuNPT on a machine powered by an Intel Xeon E3-1200 v6 processor, 16 GB DDR3 RAM and a NVIDIA GeForce RTX 3090 24 GB, benchmarking its performance. To gain a clear indication of the computational cost for the rendering of our neural representation, we limit our performance analysis to scenes whose only element is a KiloNeuS object positioned in front of the virtual camera; doing so, all the rays traced will intersect with its bounding volume going through sphere-tracing steps and, in case of a surface hit, additional MLP inferences. Under this setting, at a resolution of  $800 \times 800$  and with 1 sample-per-pixel, the renderer reaches interactive frame rates, averaging  $\sim 46$  FPS. However, it is worth noting that the cost might be largely scene- and view-dependent, particularly with other shapes and objects being present in the scene. The evaluation of our execution performance can be found in Table 2.

## 4.3 Limitations

The main limitation of our approach is that the volumetric appearance being learned by the NeuS Color network, and consequently by our KiloColor network, contains the lighting conditions at the time of dataset image capture baked on top of the true color of the object’s surface: what we would ideally like to learn. This drawback is further addressed in Section 4.4, in which we also discuss potential future solutions. Another limitation is that, compared to the two MLPs of the NeuS representation, KiloNeuS models have a larger memory footprint; a model of resolution 16 takes  $\sim 0.35$  GB for each of the two grid of MLPs ( $\sim 0.08$  MB per MLP), for a total of  $\sim 0.7$  GB. We did not focus on memory optimizations, and it is clear that there may be room for improvements; for example: while the former always have fixed memory requirements, the latter’s memory occupancy could be made proportional to the size of the object in the bounding volume by discarding MLPs representing empty

areas of space and keeping only those in the vicinity of surfaces. Nevertheless, as can be seen in Table 2, the memory footprint of our continuous representation can be seen as more cost-effective than other methods using caching to skip the MLP query and speed-up rendering times [Garbin et al. 2021; Yu et al. 2021]; those, depending on the level of detail of the targeted radiance field discretization, have a memory occupancy that ranges from as low as 0.3 GB up to more than 10 GB.



**Figure 8: Variation of various performance metrics against resolution of the KiloNeuS MLP grid, evaluated and averaged over the DTU dataset.**

#### 4.4 Future work

In the future we would like to further investigate the impact on visual quality of the size ratio between individual MLPs and their represented scene space. Finding the perfect trade-off would allow an optimal use of the representational capacity of all models involved in the active representation of the scene.

Furthermore, concurrent approaches like NeRD [Boss et al. 2021] and NeRFactor [Zhang et al. 2021] factorized the appearance of the learned object into additional neural fields encoding *albedo* and *reflectance* in terms of a spatially varying *Bidirectional Reflectance Distribution Function* (svBRDF). Their methods could be used as a template to extend the framework of the NeuS representation, encoding each of these additional properties in different MLPs. From there, they could easily be integrated into our pipeline, allowing the path-tracer to simulate different types of materials.

Lastly, it would be desirable to move the integration of the KiloNeuS renderer from the academic path-tracer on which we built our prototype to a production environment such as NVIDIA Omniverse [NVIDIA 2022] or Blender, to make the approach more easily accessible to the scientific community and to support more advanced rendering features.

## 5 CONCLUSIONS

In this paper we proposed KiloNeuS, a new neural object representation specifically designed for its real-time rendering under global illumination. We motivated the rationale behind KiloNeuS design choices and further compared its peculiarities with the characteristics of other well-known representations. Our evaluations showed reconstruction performance comparable with the state of the art, with the additional contribution of enabling global illumination. We then presented cuNPT, a custom made real-time path-tracer capable to render KiloNeuS objects in scenes containing other instances of classic representations, for which we showed examples of realistic light interactions between objects of different types. Our performance evaluation of cuNPT showed that our method can achieve competitive efficiency of both memory occupation and rendering time, reaching interactive frame rates.

## ACKNOWLEDGMENTS

This work is partially supported by the ERC grant no 802554 (SPEC-GEO).

## REFERENCES

- Henrik Aanæs, Rasmus Ramsbel Jensen, George Vogiatzis, Engin Tola, and Anders Bjørholm Dahl. 2016. Large-Scale Data for Multiple-View Stereopsis. *International Journal of Computer Vision* (2016), 1–16.
- Roger Allen. 2021. Accelerated ray tracing in one weekend in Cuda. <https://developer.nvidia.com/blog/accelerated-ray-tracing-cuda/>
- Matan Atzmon and Yaron Lipman. 2020. SAL: Sign Agnostic Learning of Shapes From Raw Data. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13–19, 2020*. IEEE, 2562–2571. <https://doi.org/10.1109/CVPR42600.2020.00264>
- Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. 2021. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12684–12694.
- Xu Chen, Yufeng Zheng, Michael J Black, Otmar Hilliges, and Andreas Geiger. 2021. SNARF: Differentiable Forward Skinning for Animating Non-Rigid Neural Implicit Shapes. In *International Conference on Computer Vision (ICCV)*.
- Valentin Deschaintre, Miika Aittala, Frédéric Durand, George Drettakis, and Adrien Bousseau. 2018. Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (ToG)* 37, 4 (2018), 1–15.
- Valentin Deschaintre, Miika Aittala, Frédéric Durand, George Drettakis, and Adrien Bousseau. 2019. Flexible svbrdf capture with a multi-image deep network. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 1–13.
- Stephan J. Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. 2021. FastNeRF: High-Fidelity Neural Rendering at 200FPS. <https://doi.org/10.48550/ARXIV.2103.10380>
- Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. 2020. Implicit Geometric Regularization for Learning Shapes. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13–18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 3789–3799. <http://proceedings.mlr.press/v119/gropp20a.html>
- John C Hart et al. 1993. Sphere tracing: Simple robust antialiased rendering of distance-based implicit surfaces. In *SIGGRAPH*, Vol. 93. 1–11.
- Philipp Henzler, Niloy Mitra, and Tobias Ritschel. 2019. Escaping plato’s cave using adversarial training: 3d shape from unstructured 2d image collections. In *Proceedings of the International Conference on Computer Vision 2019 (ICCV 2019)*, Vol. 2019. IEEE, Intel. 2018. <https://www.openimage denoise.org/index.html>
- Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings*. Yoshua Bengio and Yann LeCun (Eds.). <http://arxiv.org/abs/1412.6980>
- Jiahui Lei and Kostas Daniilidis. 2022. CaDeX: Learning Canonical Deformation Coordinate Space for Dynamic Surface Representation via Neural Homeomorphism. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. <https://cis.upenn.edu/~leijh/projects/cadex>
- Zhengqin Li, Kalyan Sunkavalli, and Manmohan Chandraker. 2018. Materials for masses: SVBRDF acquisition with a single mobile phone image. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 72–87.
- David B Lindell, Julien NP Martel, and Gordon Wetzstein. 2021. AutoInt: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14556–14565.
- Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020. Neural Sparse Voxel Fields. *NeurIPS* (2020).
- Stephen Robert Marschner. 1998. *Inverse rendering for computer graphics*. Cornell University.
- Quan Meng, Anpei Chen, Haimin Luo, Minye Wu, Hao Su, Lan Xu, Xuming He, and Jingyi Yu. 2021. GNeRF: GAN-based Neural Radiance Field without Posed Camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.
- Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019*. Computer Vision Foundation / IEEE, 4460–4470. <https://doi.org/10.1109/CVPR.2019.00459>
- 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*.
- Thomas Neff, Pascal Stadlbauer, Mathias Panger, Andreas Kurz, Joerg H Mueller, Chakravarthy R Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. 2021. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 45–59.
- Michael Niemeyer, Lars M. Mescheder, Michael Oechsle, and Andreas Geiger. 2019. Occupancy Flow: 4D Reconstruction by Learning Particle Dynamics. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 – November 2, 2019*. IEEE, 5378–5388. <https://doi.org/10.1109/ICCV.2019.00548>
- NVIDIA. 2022. Nvidia Omniverse™ platform. <https://developer.nvidia.com/nvidia-omniverse-platform>
- Michael Oechsle, Songyou Peng, and Andreas Geiger. 2021. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5589–5599.
- Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019*. Computer Vision Foundation / IEEE, 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. 2021. Nerfies: Deformable Neural Radiance Fields. *ICCV* (2021).
- Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. 2020. Convolutional Occupancy Networks. In *European Conference on Computer Vision (ECCV)*.
- Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2020. D-NeRF: Neural Radiance Fields for Dynamic Scenes. <https://doi.org/10.48550/ARXIV.2011.13961>
- Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. 2021. Derf: Decomposed radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14153–14161.
- Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. 2021. KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. *arXiv preprint arXiv:2103.13744* (2021). <https://arxiv.org/abs/2103.13744>
- Peter Shirley. 2018. Ray Tracing in One Weekend-The Book Series. [raytracing.github.io](https://raytracing.github.io)
- Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. 2020. Implicit Neural Representations with Periodic Activation Functions. In *arXiv*.
- Vincent Sitzmann, Semon Rezhikov, William T. Freeman, Joshua B. Tenenbaum, and Frédéric Durand. 2021. Light Field Networks: Neural Scene Representations with Single-Evaluation Rendering. In *Proc. NeurIPS*.
- Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. 2021. NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis. In *CVPR*.
- Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nitin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. <https://doi.org/10.48550/ARXIV.2006.10739>
- Garvita Tiwari, Nikolaos Sarafianos, Tony Tung, and Gerard Pons-Moll. 2021. Neural-GIF: Neural Generalized Implicit Functions for Animating People in Clothing. In *International Conference on Computer Vision (ICCV)*.
- Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. 2020. Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene From Monocular Video. <https://arxiv.org/abs/2012.12247> [cs.CV]
- Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021a. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *arXiv preprint arXiv:2106.10689* (2021). <https://arxiv.org/abs/2106.10689>
- Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. 2021b. NeRF—: Neural Radiance Fields Without Known Camera Parameters. *arXiv preprint arXiv:2102.07064* (2021). <https://arxiv.org/abs/2102.07064>
- Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5752–5761.
- Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul Debevec, William T. Freeman, and Jonathan T. Barron. 2021. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–18.