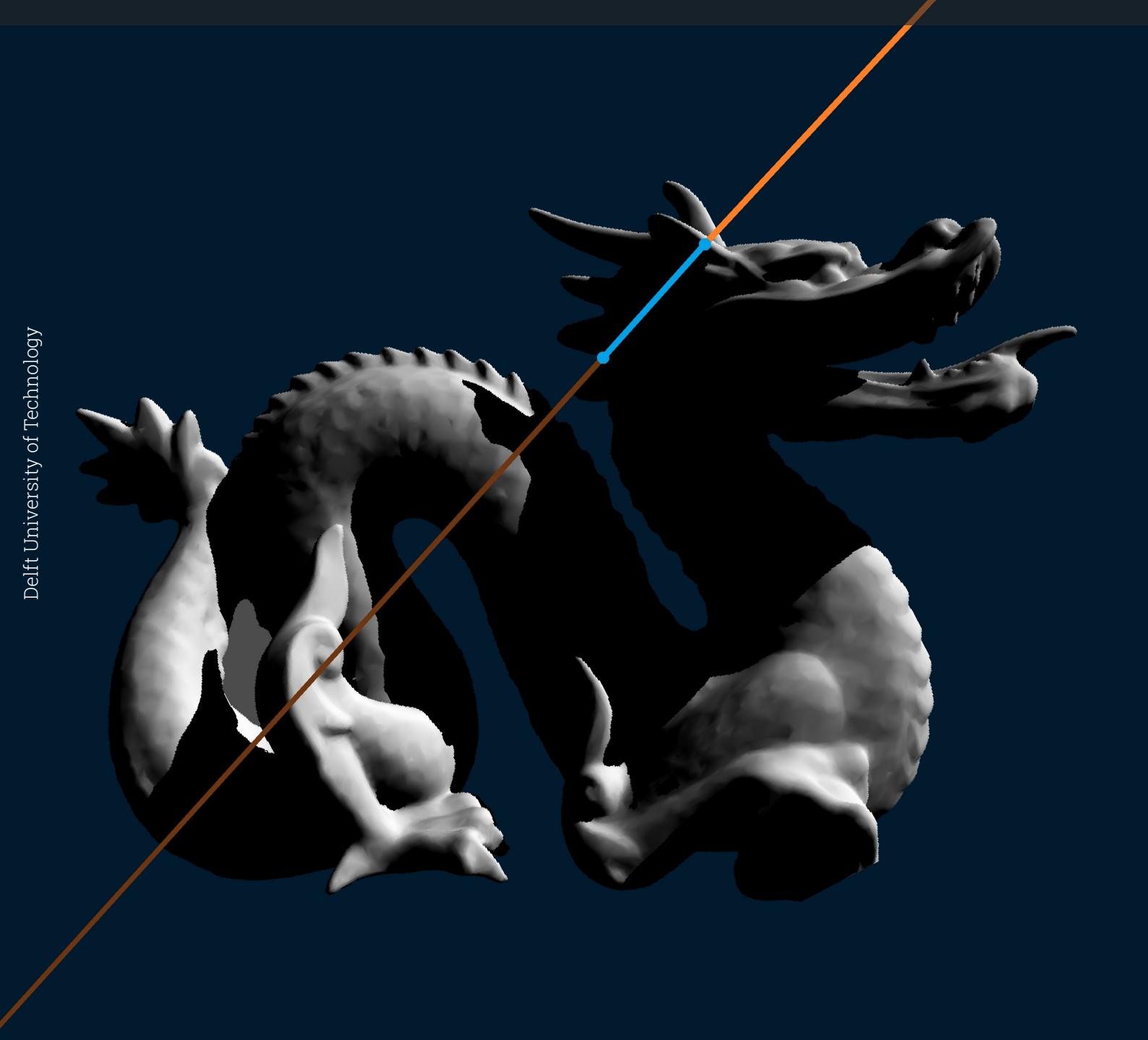
Neural Shadow Ray Queries

Computer Science Master's Thesis

T.H.B. Spanhoff



Delft University of Technology

TU Delft

Neural Shadow Ray Queries

Computer Science Master's Thesis

by

T.H.B. Spanhoff

Tygo
Spanhoff

Thesis Advisor:	R. Marroquim
Daily co-supervisor:	J.A. Rijsdijk
Project Duration:	November, 2025 - June, 2026
Research Group:	Computer Graphics & Visualization, TU Delft

TU Delft

Abstract

Rendering shadows remains a fundamental problem in computer graphics, with existing approaches balancing performance against visual quality. Traditional shadow mapping techniques are efficient but suffer from aliasing artifacts due to its discretized nature. On the other hand, ray tracing can produce high quality shadow without discretization but is often too demanding for consumer hardware.

This thesis proposes a neural shadow representation that models a continuous mapping from light rays to occluder depth. Instead of relying on discretized buffers, we represent each object's shadow with a fully connected neural network. Given a ray origin and direction, the network predicts the depth at which the ray intersects geometry. This allows for depth tests similar to shadow mapping while supporting continuous input. The model is trained using ray-traced supervision with a dead-zone loss function that encourages the model to output depth corresponding to a position inside the occluder's geometry. It does not rely on discretized shadow maps and supports fully dynamic scenes, point lights, and directional lights.

Our results show that this method can achieve visual quality comparable to medium-resolution shadow maps while eliminating aliasing. The proposed method also has a unique ability to overfit to restricted light configurations. In scenes such as outdoor scenes where light movement is limited, the model's capacity is concentrated on relevant directions, allowing it to capture finer geometric details. While limitations remain regarding inference speed and high-frequency geometry, the proposed method demonstrates unique strengths that open up new trade-offs, particularly in scenarios with partially restrained lighting configurations and large scale but low-frequency geometry such as landscapes.

Contents

Abstract i

1 Introduction 1

2 Related work 3
2.1 Background . . . 3
2.1.1 Shadow Mapping . . . 3
2.1.2 Ray tracing . . . 3
2.2 Neural rendering . . . 3
2.2.1 Neural shadow rendering . . . 4

3 Methodology 6
3.1 Overview . . . 6
3.2 Shadow representation . . . 7
3.3 Neural Network Architecture . . . 8
3.3.1 Input parameterization . . . 8
3.3.2 Frequency encoding . . . 8
3.4 Training Data Generation . . . 8
3.4.1 Poisson disk sampling . . . 9
3.4.2 Quadtree-based importance sampling . . . 9
3.4.3 Ray tracing ground truth . . . 9
3.5 Dual-depth Supervision . . . 10
3.6 Rendering pipeline . . . 11
3.6.1 Constructing model input and culling . . . 11
3.6.2 Depth testing . . . 11

4 Results 12
4.1 Setup . . . 12
4.1.1 Test scenes . . . 13
4.2 Optimization . . . 13
4.2.1 Sampling method . . . 13
4.2.2 Hyperparameters . . . 14
4.3 Comparison with shadow mapping . . . 15
4.4 Performance . . . 18
4.4.1 Inference speed . . . 18
4.4.2 Memory usage . . . 20

5 Discussion 21
5.1 Limitations . . . 21
5.2 Advantages . . . 21
5.3 Implications . . . 22
5.4 Future work . . . 22
5.4.1 Improving quality and performance . . . 22
5.4.2 Extensions . . . 23

6 Conclusion 24

7 Reflection 25
7.0.1 Research in broader CS field . . . 25
7.0.2 Implications and applicability . . . 25

8 AI Usage 27

References 28

1 Introduction

Shadows are an important part of computer graphics, both for creating realistic rendered scenes and for providing depth cues. Yet, efficiently rendering high-quality shadows remains a challenge. Traditional shadow maps are widely used due to their performance, but suffer from aliasing caused by their discretized nature. These artifacts can be reduced with larger resolution or cascaded shadow maps, but at the cost of increased memory usage and rendering time. Ray tracing can be used to test for shadows per pixel in the view port and eliminate aliasing , but the cost of traversing Bounding Volume Hierarchies (BVH) and performing intersection tests is still too computationally expensive for most general-purpose hardware.

In recent years, there have been promising advancements in the research surrounding Neural Rendering [15] and its potential to encode and predict geometric signals. Papers such as Random-Access Neural Compression of Material Textures [16] have shown that neural networks can compactly represent material textures while maintaining fast random access. Several papers have used neural techniques in view-space to more efficiently render lighting cues such as shadows [6, 4]. Although these papers had promising results, their models still rely on discretized shadow maps. This makes visiblity prediction an inherently ill-posed problem, as the models lack access to sub-pixel geometry or layered depth information. Here, the convolutional neural networks function primarily as a post-processing step for creating visually appealing smooth and soft shadows that are not necessarily geometry-based.

To overcome the limitations of intermediate discretization, several methods directly map rays to geometric properties. The Neural Intersection Function [8] represents visibility as a continuous function of rays, using a multilayer perceptron to replace part of the BVH-traversal with forward inferences. N-BVH [18] is even more dependent on the BVH, but replaces the last few layers with a neural network to predict numerous geometric properties. Since the last few layers are very wide, this allows a more compact representation of the scene without losing visual quality. However, they still require traversal of spatial structures, which add time and complexity.

In this thesis, we propose a neural shadow representation that directly learns a global continuous mapping from rays to occluder depth. For each object, this representation is implemented using a fully connected multilayer perceptron (MLP). To help with high frequencies, a frequency encoding is used on the input, inspired by NeRF [10]. To resolve ray aliasing, the MLP only considers rays which originate from the minimal bounding sphere of the object. Any ray can be mapped to the minimal bounding sphere with a simple ray-sphere intersection, for which the neural network then predicts the depth to occluding geometry. This allows for depth testing in a similar fashion to traditional shadow maps while allowing for real-valued sampling without the aliasing that causes artifacts with shadow mapping.

Furthermore, the method relaxes supervision during training with a technique inspired by dual shadow maps [19]. Instead of targeting a single point, we use two targets per ray sample: a lower and upper bound, which correspond to the depths of the ray's first entry and exit point with the occluder. We set the loss within these bounds to zero, allowing the model to output anywhere within the geometry. We further improve the training process with quadtree-based importance sampling. We divide our training samples into several camera angles and create an orthogonal silhouette render for each. A quadtree is created on this silhouette, which

results in smaller quads near the high-frequency silhouette borders. Evenly dividing samples among quads results in a simple importance-sampling strategy where samples are focused on high-frequency parts of the geometry. Finally, we use Poisson-Disc sampling to guarantee minimum distances between origins, leading to evenly distributed coverage.

We show that this representation can achieve visual quality similar to traditional shadow maps while eliminating aliasing. Furthermore, the model has a unique and natural ability to overfit to restricted light paths. In scenes such as outdoor landscapes, where the sun only rotates on a single axis compared to the landscape, the model is able to concentrate its learning capacity on the relevant light directions. This effectively reduces the problem's dimensionality and enables the model to learn higher-frequency geometric details than it could in a fully dynamic configuration.

The contributions of this paper are threefold:

- A continuous neural representation for hard shadows that predicts in ray-space and still works as a drop-in replacement for traditional shadow maps. The representation allows for continuous input without intermediate discretization or the need for spatial acceleration structures.
- A training strategy with ray-traced supervision that is relaxed using dual depth bounds.
- An evaluation of the proposed method to existing shadowing techniques in terms of visual quality, memory usage, and performance.

2

Related work

2.1. Background

2.1.1. Shadow Mapping

Shadow mapping [20] remains the dominant approach for real-time shadow rendering due to its combination of performance and simplicity. The method renders depth from the light's perspective into a shadow map, after which shaded points are tested for visibility by comparing their light-space depth to the stored depths in the shadow map [20]. Shadow mapping is efficient because it requires just one additional render and a texture lookup for every fragment during shading, but it is susceptible to aliasing. Techniques such as cascaded shadow maps [22] or higher-resolution shadow maps can partially mitigate these issues at the cost of performance and memory usage, but no shadow mapping technique can truly eliminate aliasing and resolution limitations as shadow mapping relies on discretization [14]. Our method addresses this limitation by replacing the discretized shadow map with a continuous neural representation of light-space visibility.

2.1.2. Ray tracing

Ray tracing computes visibility by tracing rays through the scene and testing for intersections with geometry. A vast number of ray-triangle intersection tests can be skipped through the traversal of spatial data structures such as BVHs. Ray tracing can produce physically accurate shadows by tracing rays from the shaded points to light sources and testing for visibility. Although its use is increasing and several techniques have been applied to make ray tracing more efficient [2], ray-traced shadows still require spatial acceleration structures and high-performance hardware to be viable in real-time. In contrast, our method maps rays directly to visibility without requiring traversal of a spatial hierarchy.

2.2. Neural rendering

In recent years, neural rendering has emerged as an alternative or an extension of various parts of the rendering pipeline [15]. Instead of explicitly storing geometry or textures, a neural network models a continuous field that represents geometric information. It can be more compact or faster than traditional discrete techniques without losing fine detail due to discretization.

One of the most notable examples is NeRF [10], which models an entire 3D scene with a neural network that outputs density and radiance when sampled at a given 3D position and viewing direction. While mainly used for novel view synthesis, NeRF demonstrates that neural networks are powerful enough to model and render continuous representations of complex geometry, with recent variants even enabling real-time rendering. [21]. Neural techniques have also been applied to textures [16]. Vaidyanathan et al. neurally encode multiple channels and mipmap levels of material textures. These neural networks can be more compact and more detailed than explicit encodings while still allowing for random-access sampling. These results suggest that neural networks are capable of representing high-frequency functions in a compact and continuous way, highlighting their potential for shadow representations as well.

2.2.1. Neural shadow rendering

Existing research on neural shadow rendering can be broadly divided into view-space and ray-space methods. View-space methods infer visibility from rasterized buffers, whereas ray-space methods model visibility as a function of rays.

View-space methods

View-space methods predict screen-space shadows from rasterized buffers, often in screen space or light space. This can include screen-space depth maps, emitter-space depth maps (shadow maps), and it can also include screen space shadows cast by shadow maps. They typically use a convolutional neural network to leverage the spatial structure in such buffers.

Neural Shadow Mapping by Datta et al. [6] proposes a neural extension to basic shadow mapping that produces high-quality hard and soft shadows. Their method uses several G-buffers as input to a convolutional neural network with a U-net architecture, including shadow maps and view-space depth and normals. Their training is supervised by ray-traced screen-space hard and soft shadows, while the loss is also measured after applying small perturbations to the camera and emitter, to reduce instabilities introduced by shadow map aliasing.

More recent work by Chen et al. iterates on this approach by removing all extra G-buffers from the features, relying only on a number of hard shadows [4]. These shadows are created by sampling along the boundary of an area light, creating virtual point lights with corresponding shadow maps. The view-space hard shadows from the shadow maps are used as input to the U-net network to output high-quality soft shadows.

Although screen-space methods can create visually convincing shadows, their reliance on discretized shadow maps prevents them from accurately predicting sub-pixel occlusion. Furthermore, the network does not have access to layered depth information, making visibility prediction inherently ill-posed. These approaches perform neural inference as a post-processing step instead of directly modeling the light-space visibility function as a neural field. The required number of shadow maps may also increase with the emitter's area to prevent hard shadow contours from becoming visible in the soft shadow. In contrast, our method directly maps rays to visibility and avoids intermediate discretization altogether.

Ray-space methods

Ray-space methods aim to model visibility directly from rays, bypassing the need for intermediate discretization.

The Neural Intersection Function (NIF) [8] represents visibility as a function of rays, by mapping a ray's origin and direction to a visible probability between 0 and 1. One core problem with this representation is that infinitely many rays share the same direction while differing in origin. The Neural Intersection Function solves this by encapsulating the occluder in an Axis-Aligned Bounding Box (AABB). For every ray, the intersection point of the ray and the AABB is given to the neural network as the ray's origin, collapsing the infinite number of overlapping rays to a single representation.

However, for concave geometry, visibility strongly depends on the position of the ray's origin. In these cases, using only the AABB intersection point would prevent the model from correctly predicting self-shadowing. To address this problem, NIF uses two neural networks: one for rays originating within the AABB and another for outside rays. To handle self-shadowing, the inner network takes the distance from the center of the AABB to the hit point as an additional input.

The authors of NIF also note that since AABBs are convex , the boundaries can be represented with 2D spherical coordinates. Although this reduces the dimensionality of the problem, it is harder for the neural network to learn the high-frequency geometry from only two dimensions. Instead, the authors opt for a grid encoding, which increases dimensionality but aids the network in learning high frequencies. Even though NIF is able to speed up ray casting with up to 1.53x while preserving image quality, it is overfit to the viewpoint and has only been evaluated for secondary ray casting.

Although NIF uses a shallow BVH to split the scene into AABBs, Neural BVH [18] takes this a step further and replaces only the last few layers with neural leaf nodes. Nevertheless, these layers are the widest and thus their approach manages to reach a compression ratio of 13x across an entire scene. Their core idea is that neural networks can more accurately predict the geometric properties of intersection locations when this

intersection is close to the input location. This is achieved with the BVH, since a deep BVH creates bounding volumes that more tightly surround the object.

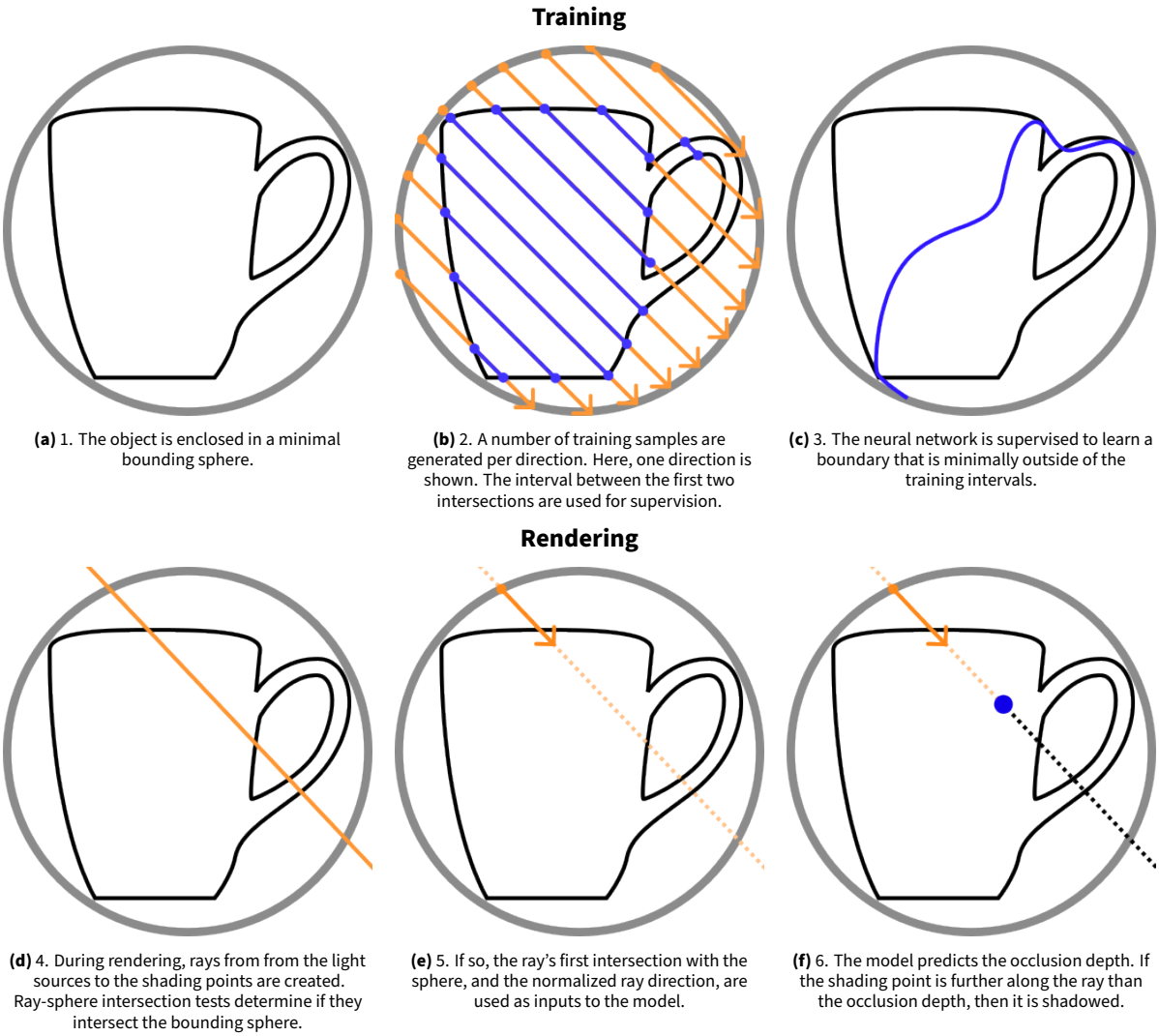
Additionally, the neural network does not just take the ray origin and direction as input, but instead takes multiple uniformly sampled points along the ray as input for a closer minimum distance between these points and the actual intersection. This does raise issues with self-shadowing, but the geometry within a neural leaf is assumed to be so small that it is assumed to be convex. Although it manages a high compression ratio, it still relies on a spatial acceleration structure, the traversal of which takes some time. As such, it does not compete with ray tracing methods in terms of speed. Our method does not rely on BVHs, offering the potential for faster queries.

Unlike NIF, which is limited to static scenes, and N-BVH, which replaces only the deepest levels of a BVH with neural approximations, we propose a global continuous mapping from rays to occluder depth for direct shadow evaluation. Using a ray-space representation and dual depth bounds [19], our model works as a drop-in replacement for shadow maps without intermediate discretized buffers or spatial hierarchies.

3

Methodology

3.1. Overview



(a) 1. The object is enclosed in a minimal bounding sphere.

(b) 2. A number of training samples are generated per direction. Here, one direction is shown. The interval between the first two intersections are used for supervision.

(c) 3. The neural network is supervised to learn a boundary that is minimally outside of the training intervals.

(d) 4. During rendering, rays from from the light sources to the shading points are created. Ray-sphere intersection tests determine if they intersect the bounding sphere.

(e) 5. If so, the ray's first intersection with the sphere, and the normalized ray direction, are used as inputs to the model.

(f) 6. The model predicts the occlusion depth. If the shading point is further along the ray than the occlusion depth, then it is shadowed.

Figure 3.1: Step-by-step visualization of training and rendering.

An object's hard shadow is represented by a direct and continuous mapping from rays to occluder depth. Us-

ing rays from shading points to the light source, this allows for depth testing akin to shadow mapping. We use a neural network to predict this function, allowing for continuous in- and output. To reduce the input space to a finite domain, we restrict rays that are input to the model to originate on an approximately minimal bounding sphere surrounding the object. During rendering, we calculate the positions where rays enter the bounding sphere. This position, together with the light direction, serve as input to the model. The occluder depth that the model then outputs the depth along the ray that defines the transition between lit and shadowed. Training samples are generated by generating random rays originating on the bounding sphere and finding the first two intersections between the ray and the occluder. With a dead-zone loss function (Section 3.5), we stimulate the network to predict in between these bounds for as many rays as possible.

3.2. Shadow representation

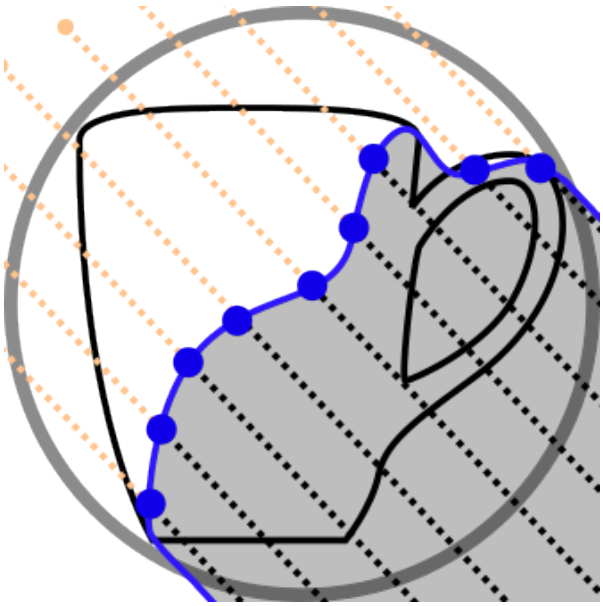


Figure 3.2: A function models the boundary between lit and shadowed.

For a ray that intersects geometry, the transition between illuminated and shadowed regions occurs at a single depth along the ray. We therefore represent an object's shadow by predicting this depth directly with the following function:

$$f : (\mathbf{o}, \mathbf{d}) \rightarrow t_s$$

where $\mathbf{o} \in \mathbb{R}^3$ is the ray origin on the minimal bounding sphere enclosing the object, and $\mathbf{d} \in \mathbb{R}^3$ is a normalized direction vector. The scalar $t_s \geq 0$ denotes the parametric distance along the ray such that all points $\mathbf{o} + t\mathbf{d}$ with $t > t_s$ lie in shadow with respect to the object. t_s determines the position of the "shadow boundary" along the ray (Figure 3.1f). When this shadow boundary is completely within the first two layers of geometry as seen from the light source, it correctly models the occluder's shadow [19] (Figure 3.2).

To test whether a point p in the scene is in the shadow of the object, a ray is cast from the light source to this point. If this ray intersects the object's minimal bounding sphere (Figure 3.1d), the direction of this ray and the first intersection point with the sphere serve as inputs to the function S to find t_s (Figure 3.1e). Similarly to regular shadow mapping, a depth test can determine whether p is occluded by comparing its distance to a light source with t_s: p is occluded by a light source l if and only if $dist(p, l) > t_s$. Unlike shadow mapping, the representation is continuous and not discretized into texels.

Representing the occluder depth instead of occlusion changes the problem of predicting hard shadows from a classification problem to a regression problem. The main advantage of this representation is that it naturally forces a single shadow boundary, which is physically accurate for hard shadows. If the model were to directly output occlusion, it would have to implicitly learn that there is a single transition point on a ray between lit and shadowed regions. This is no easy task, as the sample set would have to contain many points on a single ray before the model can implicitly learn the single transition.

3.3. Neural Network Architecture

The function is represented using a fully connected neural network per object. ReLU activations are used after every layer. To improve the network's ability to represent high-frequency detail, a positional encoding is applied to both the origin and direction input vectors (Section 3.3.2).

3.3.1. Input parameterization

The shadow function could in principle operate on arbitrary rays. However, this domain is unbounded because multiple ray origins along the same line of sight would correspond to the same visibility queries: If a ray r_1 intersects the geometry, then any ray r_2 that passes through the origin of r_1 and has the same direction as r_1 will intersect the geometry at the exact same location. Representing all these rays independently would add unnecessary redundancy.

Instead, we can limit the input domain by only taking into account origins on an object's minimal bounding sphere. For any ray, a ray-sphere intersection test determines the origin to use as input to the function. This essentially changes the input domain from unbounded $\mathbb{R}^3$ to a two-dimensional manifold embedded in $\mathbb{R}^3$. And since this domain is finite, it allows training samples to be normalized to the range $[0, 1]$.

However, it must be noted that this representation could fail when the light ray originates from within a concave part of the geometry. While our implementation does not take this into account, this issue can be prevented by exclusively mapping ray origins outside of the bounding sphere to the sphere's surface, while leaving ray origins within the bounding sphere unchanged. This does not matter for our experiments, as they were performed with directional lighting, for which rays never originate from within the scene.

3.3.2. Frequency encoding

Multilayer perceptrons are known to suffer from a spectral bias: lower frequencies are learned more easily than high-frequency functions [13]. As a result, MLPs tend to reproduce smooth variations before capturing fine details. Since our shadow representation has high-frequency borders near object silhouettes, training a model on raw origins and directions causes it to underfit to the data.

Following NeRF's strategy [10], we use a frequency encoding to help the model represent higher frequencies. Instead of just providing the network with the raw ray origin and direction, the input coordinates are mapped to a set of sinusoidal basis functions with increasing frequencies. This gives the model access to low- and high-frequency variations of the input, making it easier to represent the sharp changes that are present in our shadow representation.

For an input scalar x, the encoding is defined as

$$\gamma(x) = \left(x, \sin(2^0\pi x), \cos(2^0\pi x), \ldots, \sin(2^{L-1}\pi x), \cos(2^{L-1}\pi x)\right),$$

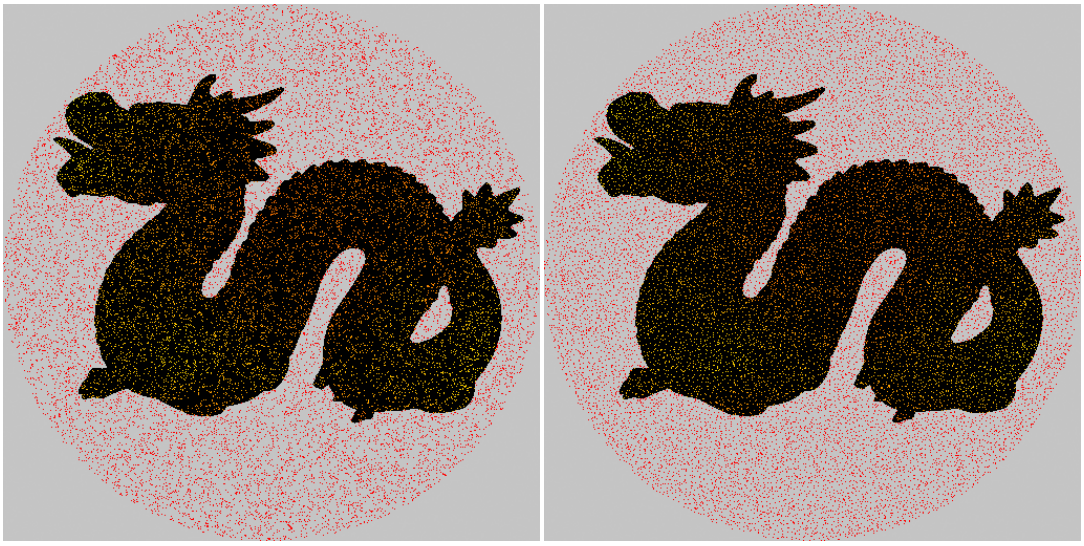
Where L is the number of frequencies. The encoding is applied separately to all six scalars in the position and direction coordinates, increasing the total number of dimensions from six to $6(1 + 2L)$. The model predicts the unencoded scalar value corresponding to the occluder depth t_s.

3.4. Training Data Generation

Training data is generated offline using ray tracing. While it is possible to sample origins and rays uniformly, this approach has two issues. First of all, uniformly sampled sets of points are bound to have areas with high and low sample densities, which hinder the network when learning areas with smaller sample densities. To solve this, we use Poisson disk sampling [5] to sample rays in a *blue noise* pattern; blue noise contains high-frequency randomness while containing virtually no low frequency signals. This sampling strategy results in even sampling density, while the high-frequency randomness prevents the model from learning structures that are not based on the ground truth.

The other issue with uniform sampling is that only a small portion of the rays will pass close by the object, which is where the highest frequencies commonly exist in our representation. To get the model to accurately predict these borders, an incredibly large number of samples would be required without selective sampling. Instead, we use an importance sampling strategy based on quadtrees which is simple to implement and focuses samples on the high-frequency borders.

3.4.1. Poisson disk sampling



(a) Uniform sampling contains holes and clusters

(b) Poisson-disk sampling creates a blue noise, which contains high-frequency randomness but an even distribution in the low frequencies

View directions are sampled from an object's minimal bounding sphere using Poisson disk sampling to ensure approximately uniform coverage, while also ensuring high-frequency randomness to prevent unwanted patterns from appearing in the model's predictions. Poisson disk sampling ensures that every point has a minimal given distance r from every other point [5]. This can be done with an *active list* [3]. A random point is placed in the active list, after which the following steps are repeated until the active list is empty:

- A point p is randomly selected from the active list
- Up to k times, a new point is generated in the disk between radius r and $2r$ around p
- If this new point is at least a distance r from every other point in the sample set, accept it.
- If none of the k candidates were valid, p is removed from the active list. p is still kept in the sample set.

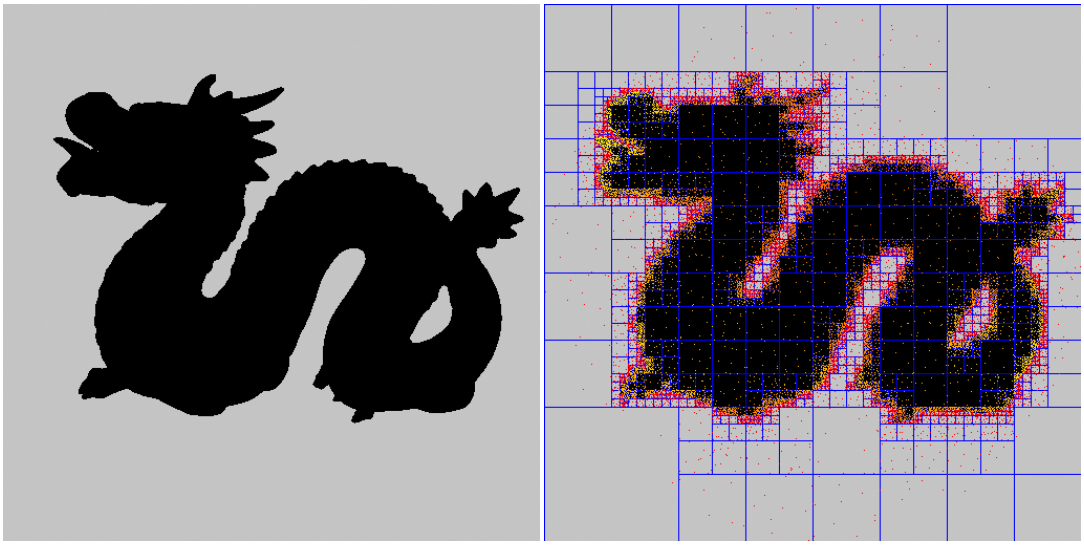
Here, k is a hyperparameter that controls how often the algorithm tries to find a valid candidate. A larger k can allow the algorithm to fit more points in a space, but it leads to a slower runtime. To improve performance, sampled points are sorted into a grid of bins to allow fast spatial look-ups that do not require iterating over every single sample. [3].

3.4.2. Quadtree-based importance sampling

For each Poisson disk sampled view direction $\mathbf{d}$, a silhouette rendering of the object is created by projecting it orthogonally to $\mathbf{d}$. Regions near the border of this silhouette correspond to discontinuities in the shadow function and therefor contain high frequencies. To focus our training samples on these regions, we construct a quadtree [7] on the silhouette and subdivide quads that contain the silhouette border until every quad is completely inside or outside the silhouette, or reaches a minimum size. This results in a higher density of quads near the borders of the silhouette. m points are divided evenly across quads, using Poisson disk sampling to generate point positions within the quad. For every point, we use the L2-distance to the center of the render to calculate $\mathbf{o}$ on the bounding sphere such that $\mathbf{o} + t\mathbf{d}$ intersects this point. This procedure results in an importance sampling strategy where more rays pass close by the border of the object, which is where the highest frequencies are due to visibility discontinuities.

3.4.3. Ray tracing ground truth

After sampling a ray, ray tracing is used to find its intersections with the occluder. Both the first and second intersection, t_{lb} and t_{ub} respectively, are used as targets for the neural network. When the ray only intersects the geometry once, which can happen in non-closed geometry such as landscapes, t_{ub} is set to ∞. When the ray completely misses the occluder, t_{lb} is set to be the total length of the ray-portion inside the bounding sphere, and t_{ub} is set to ∞. This ensures that for a ray $\mathbf{o} + t\mathbf{d}$, $t_{lb}...t_{ub}$ is the maximum range for which any



(a) For every viewpoint, an orthogonal render of the object is created.

(b) A quadtree is created from this render such that the inside of quads is either entirely white or black. By sampling an even number of rays in every quad, we obtain a strategy that focuses samples on high-frequency borders.

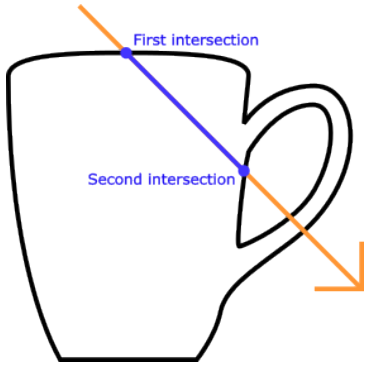


Figure 3.5: The first two intersections between the ray and the geometry determine the lower and upper bounds used for supervision.

point $\mathbf{o} + t_{pred}\mathbf{d}$ where $t_{lb} \leq t_{pred} \leq t_{ub}$ is inside of the occluder or outside of the bounding sphere.

3.5. Dual-depth Supervision

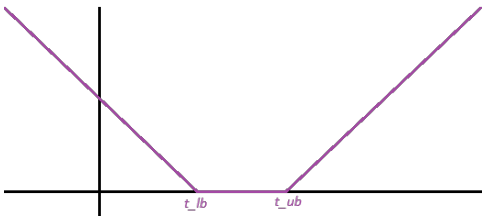


Figure 3.6: The loss function takes two parameters: t_{lb} and t_{ub}, between which the loss is 0. Outside these bounds, the loss is the L1-distance to the nearest bound.

The model takes advantage of dual depth bounds, inspired by dual depth mapping [19]. Instead of creating a shadow map from the closest depths to the light source, dual depth mapping takes into account the first two depth layers from the light. These two layers can then be used to determine a depth bias that reduces aliasing. The core concept is that a shadow map's depth values can be anywhere in between these two layers, without affecting the shadows produced; the first depth layer is "in front of the shadow map", so it will be lit, and the second layer is "behind the shadow map", so it will be in shadow.

We apply this concept to increase the target range during supervised training, using a dead-zone loss function based on two targets per sample: a lower bound t_{lb} and an upper bound t_{ub}. The loss of the network's prediction t_{pred} is the L1-distance to the nearest bound when it is out of bounds, and 0 otherwise:

$$L(t) = \begin{cases} t_{lb} - t & t < t_{lb}, \\ t - t_{ub} & t > t_{ub}, \\ 0 & otherwise \end{cases}$$

Any prediction within the interval $[t_lb, t_ub]$ produces the same correct shadows. Penalizing differences within this interval would encourage the network to learn arbitrary details that do not contribute to shadow accuracy. The dead-zone loss function avoids this by treating all values in this interval as equally correct.

3.6. Rendering pipeline

Each dynamic object in a scene is represented by its own neural shadow model and corresponding bounding sphere.

3.6.1. Constructing model input and culling

To keep the method space and time efficient, intermediate values are stored in screen-space buffers. First, a depth map is rendered from the camera viewpoint. A CUDA kernel then transforms every pixel in the depth map to its world-space coordinate $\mathbf{p}$ using the affine inverse of the view-projection matrix. For every light or sample point in an area light, a ray $r = \mathbf{p} + t\mathbf{l}$ is constructed, where $\mathbf{l}$ is the normalized vector from $\mathbf{p}$ towards the light.

As the model is trained on ray origins that lie on the object's bounding sphere, a ray-sphere intersection test with the bounding sphere is used to get the origin to use as input to the neural network. Rays that completely miss the bounding sphere are assigned a value of 0 in an additional boolean mask, while intersecting rays are assigned a value of 1. The mask is then compacted using a parallel prefix-sum operation [1], producing an index list that contains the indices of all rays for which to run inference. Rays that do not intersect the bounding sphere are ignored, significantly reducing the number of forward inferences when the object occupies only a small portion of the screen.

3.6.2. Depth testing

For rays that intersect the bounding sphere, the model predicts an output t_{pred}. This predicted depth is then compared to the shaded point's distance to the light in the same fashion as it would be with traditional shadow maps; If t_{pred} of forward inference on r is smaller than the distance between $\mathbf{p}$ and the light source, then the point $\mathbf{p}$ is occluded in the direction of r. For soft shadows or multiple light sources, the results of all rays can be aggregated to find the light intensity at $\mathbf{p}$. This process is similar to ray tracing shadows, but the entire BVH-traversal and intersection calculation is replaced by a single bounding volume and a neural network that directly outputs occluder depth for depth testing.

If r is occluded by any object, the process can be terminated early without running inference on other object's corresponding MLP's. To increase the chance of early termination, it is sensible to order the testing of different objects with a metric such as the average occlusion percentage of an object in its bounding sphere over all directions.

4

Results

4.1. Setup

All experiments were performed using an implementation based on the tiny-cuda-nn framework [11]. The experiments were executed on a system equipped with an AMD Ryzen 7 9800X3D processor, an NVIDIA GeForce RTX 5070 Ti with 16 GB of memory, and 32 GB of DDR5 RAM.

Unless stated otherwise, all experiments use the configuration below. The configuration was based on the hyperparameter optimization in section 4.2.

Encoding

- Encoding type: `Frequency`
- Number of frequencies: 10

Network Architecture

- Network type: `CutlassMLP`
- Activation function: `ReLU`
- Output activation: `ReLU`
- Hidden layers: 8
- Neurons per layer: 256

Optimization

- Optimizer: `Adam`
- Learning rate: 1×10^{-3}
- Loss function: `Custom Dead-Zone Loss Function (Section 3.5)`
- Batch size: 65 536
- Training steps: 1 000 000

Rendering

- Screen resolution: 1200×800

Model and sampling

- Default model: `dragon`
- Default sampling method: `Poisson Disk sampling with quadtree importance`
- Training set size: 250,000,000 samples (10,000 samples for 25,000 viewpoints)

The proposed method is evaluated against conventional shadow maps with resolutions ranging from 128×128 up to 2048×2048. All comparisons assume directional lighting, although our model also supports point lights. For shadow mapping, the scale of the directional shadow map is set to tightly enclose the object.

To quantitatively evaluate shadow quality, all methods are compared against a high-resolution reference rendered using a 32768×32768 shadow map. At this resolution, aliasing artifacts are visually negligible and the resulting images are treated as ground truth.

For each experiment, 2000 viewpoints are evaluated, consisting of 1000 viewpoints looking at the cast shadows and 1000 viewpoints aimed at the object's surface to evaluate self-shadowing. Each viewpoint is rendered for our method and all shadow map resolutions.

Image quality is measured using LPIPS [23], PSNR, and SSIM [17]. These are widely used image similarity metrics. A higher PSNR or SSIM indicates a higher similarity with the image, while the LPIPS value decreases when image similarity increases. LPIPS uses deep learning to measure perceptual similarity [23]. Because LPIPS is the state-of-the-art in measuring human perceptual similarity, our analysis will focus on this metric.

4.1.1. Test scenes

The model was evaluated on three different models with varying characteristics:

- **Dragon**: A common benchmark model containing both low- and high-frequency geometric detail [9].
- **Fence**: A challenging scene with thin geometry, high-frequency structures, and small depth ranges. Such thin features are known to produce artifacts in conventional shadow mapping techniques.
- **Landscape**: A scene with mostly low frequencies. It consists of a single plane with extrusions. With our method, many ray samples will only contain a lower bound.

For the landscape scene, we also train a model variant that learns to predict depth for only a single axis of light directions. This simulates scenes where fully dynamic lighting is not required because light only originates from a limited set of directions, such as outdoor scenes lit by sunlight. Common techniques such as shadow mapping and ray tracing do not fully take advantage of such cases, and often treat them as fully dynamic scenes. Our neural method is naturally able to overfit in these scenarios. This reduces the complexity of the problem and allows the model to predict higher-quality shadows for the relevant light directions.

4.2. Optimization

4.2.1. Sampling method

Table 4.1 compares different strategies for generating training rays. The combination of Poisson Disk sampling and quadtree-based importance sampling, described in sections 3.4.1 and 3.4.2 respectively, achieves the best visual quality across all evaluation metrics.

Uniform sampling produces the lowest quality results. This is expected, as it distributes rays evenly across both high-frequency areas as well as low-information areas such as empty space. This results in undersampling of the high frequencies where rays either narrowly hit or miss the object.

Uniform quadtree sampling focuses samples on this shadow boundary, leading to an improvement in the evaluation metrics. However, uniform quadtree sampling does not guarantee full coverage of the scene. Some regions might be underrepresented, while other areas will have a redundant number of ray samples.

The best achieving method combines quadtree-based importance sampling with Poisson Disk sampling. Since Poisson Disk sampling evenly divides rays in low frequencies, this combination both focuses rays on the higher frequency border while reducing under-sampled regions.

These results suggest that the distribution of training rays has a substantial impact on the quality of the learned shadow representation. Focusing samples near shadow boundaries improves reconstruction quality

in these high-frequency regions, while Poisson Disk sampling reduces under sampled regions. Consequently, this combination of Poisson Disk sampling with quadtree-based importance was used for all remaining experiments.

Method	LPIPS ↓	PSNR ↑	SSIM ↑
Poisson Disk Quadtree sampling	0.0392	27.21	0.9769
Uniform Quadtree sampling	0.0453	26.50	0.9746
Uniform Sampling	0.0539	25.54	0.9712

Table 4.1: Different sampling strategies sorted by LPIPS.

4.2.2. Hyperparameters

The optimal learning rate depends on multiple factors, including geometry shape, model size, and input encoding. Optimizing the learning rate for every experiment was not feasible within the scope of this project. Instead, the dragon occluder with the setup from Section 4.1 served as the baseline for all experiments. The dragon contains both low and high frequencies, making it representative of general scene complexity.

Figure 4.1 shows the training loss for several learning rates over two million iterations. A learning rate of 0.001 resulted in the smallest loss, while both higher and lower rates resulted in a higher loss after 2,000,000 iterations.

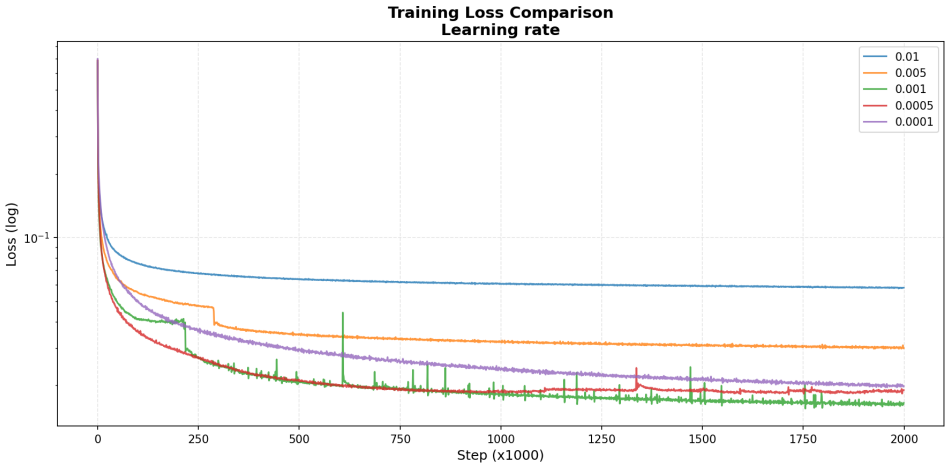


Figure 4.1: Training loss for different learning rates

Figure 4.2 compares the training loss for varying numbers of frequencies. An encoding with ten frequencies achieved the lowest training loss and LPIPS score.

Using fewer frequencies, such as 5 or the identity encoding, limits the ability of the model to represent higher frequencies, leading to underfitting. Increasing the number of frequencies beyond ten does not improve performance and instead leads to a higher loss. Since both the LPIPS and the training loss increased, this suggests that the issue is not to do with overfitting. Instead, an encoding with more frequencies increases complexity with no gains in the capacity to model high frequencies, making convergence more difficult.

Method	LPIPS ↓	PSNR ↑	SSIM ↑
10 Frequencies	0.0379	27.39	0.9772
30 Frequencies	0.0496	27.13	0.9751
5 Frequencies	0.0539	25.35	0.9677
20 Frequencies	0.0590	25.83	0.9698

Table 4.2: Neural networks with different numbers of frequencies, sorted by LPIPS

Figure 4.2: Training loss for different encodings

4.3. Comparison with shadow mapping

Figure 4.3 shows the training loss across several scenes. As expected, performance decreases as the scene complexity increases. The mesh fence is the most difficult with its thin geometry, while the Dragon and landscape scenes are easier to model because they have smoother and thicker geometry. The restricted single-axis lighting setup is the easiest case, as it reduces the complexity of the problem and allows the model to overfit to a smaller input domain.

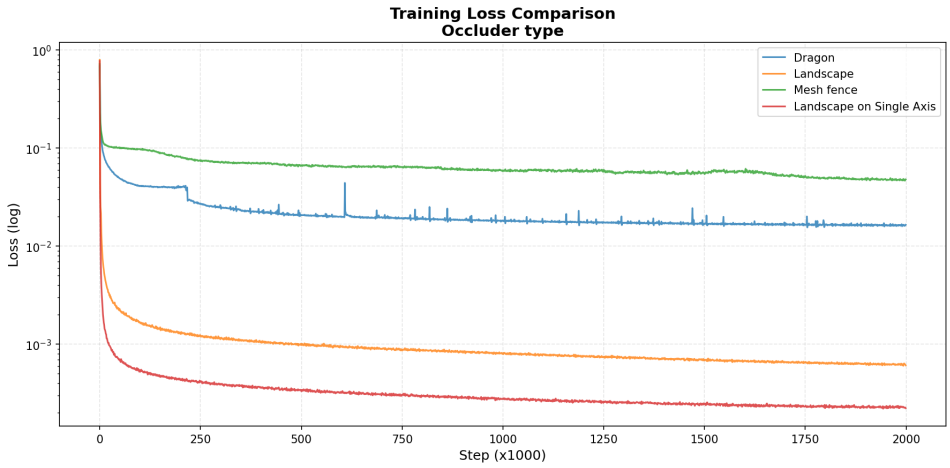


Figure 4.3: Training loss on different models

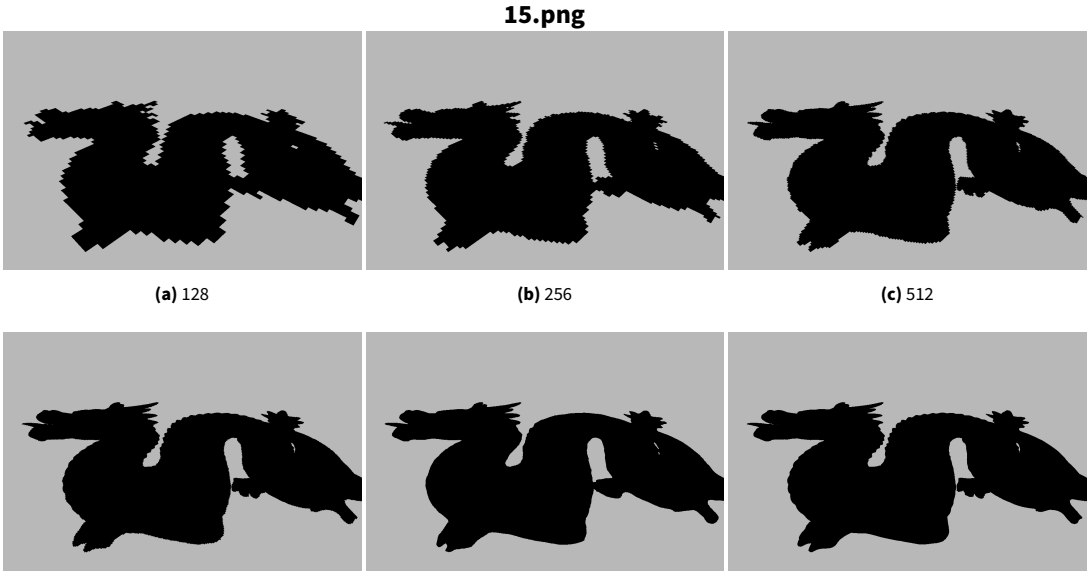
Table 4.3 shows that our method achieves a lower LPIPS than a 256×256 shadow map for the dragon model, while requiring no explicit shadow-map rasterization. Visually, the neural representation preserves smooth shadow boundaries and avoids the strong aliasing artifacts visible at lower shadow-map resolutions, as in Figure 4.5.

However, the 512×512 and higher-resolution shadow maps still outperform the neural representation in all similarity metrics. This is primarily due to the neural method's difficulty with understanding high frequencies, such as in the hind claws in Figure 4.4.

The mesh fence model is significantly more challenging because of its thin features and the high-frequency grid. Our method is unable to produce shadows of the diagonally overlapping mesh, and instead produces shadows in one direction only, as seen in Figure 4.6. Quantitatively, the neural method performs only better than a 128×128 shadow map (Table 4.4).

Method	LPIPS ↓	PSNR ↑	SSIM ↑
2k by 2k shadowmap	0.0095	38.50	0.9963
1k by 1k shadowmap	0.0207	35.48	0.9930
512 by 512 shadowmap	0.0329	32.46	0.9875
Our method	0.0379	27.39	0.9772
256 by 256 shadowmap	0.0465	29.42	0.9793
128 by 128 shadowmap	0.1541	24.63	0.9303

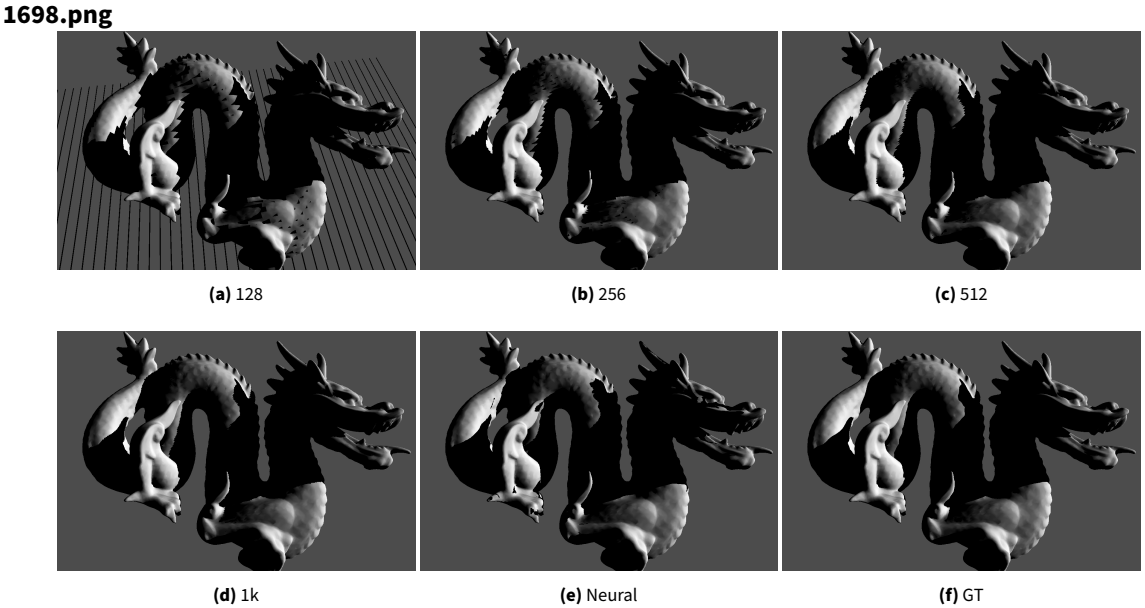
Table 4.3: Comparison of our method and directional shadow maps with varying resolutions on the dragon object.



(a) 128 **(b)** 256 **(c)** 512

(d) 1k **(e)** Neural **(f)** GT

Figure 4.4: Comparison of shadow map resolutions, neural rendering, and ground truth for a pure shadow frame of the dragon model.



(a) 128 **(b)** 256 **(c)** 512

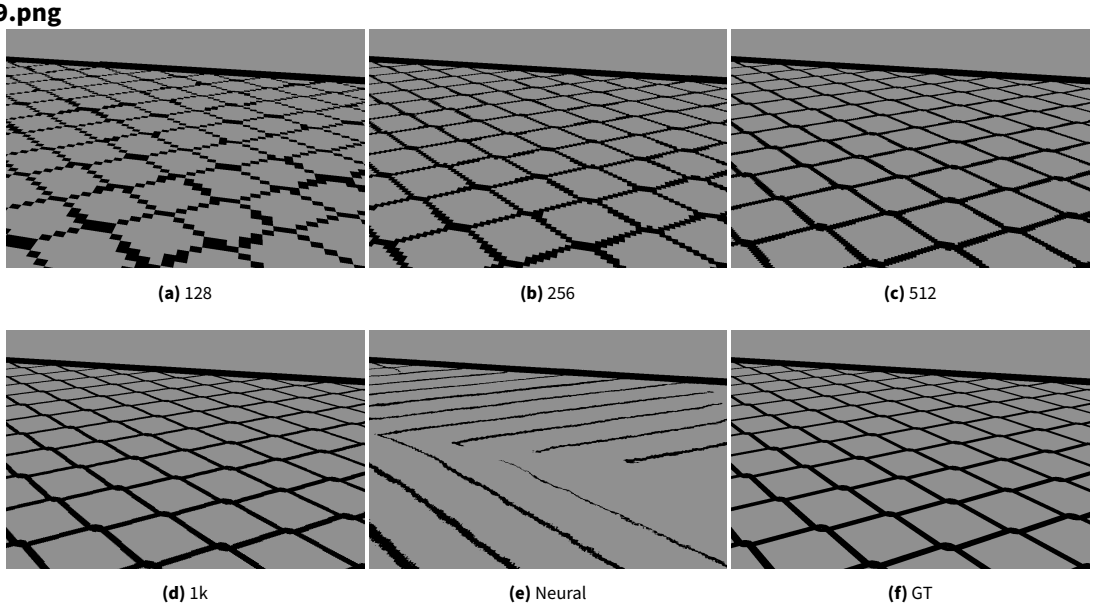
(d) 1k **(e)** Neural **(f)** GT

Figure 4.5: Comparison of shadow map resolutions, neural rendering, and ground truth for a self-shadowing frame of the dragon model.

Method	LPIPS ↓	PSNR ↑	SSIM ↑
2k by 2k shadowmap	0.0274	31.59	0.9807
1k by 1k shadowmap	0.0564	28.59	0.9634
512 by 512 shadowmap	0.0896	25.60	0.9349
256 by 256 shadowmap	0.1187	22.62	0.8954
Our method	0.1652	17.29	0.8282
128 by 128 shadowmap	0.2171	18.81	0.8166

Table 4.4: Mesh fence similarity with ground truth.

1349.png



(a) 128 **(b)** 256 **(c)** 512

(d) 1k **(e)** Neural **(f)** GT

Figure 4.6: Comparison of shadow map resolutions, neural rendering, and ground truth across certain frames of the mesh fence model.

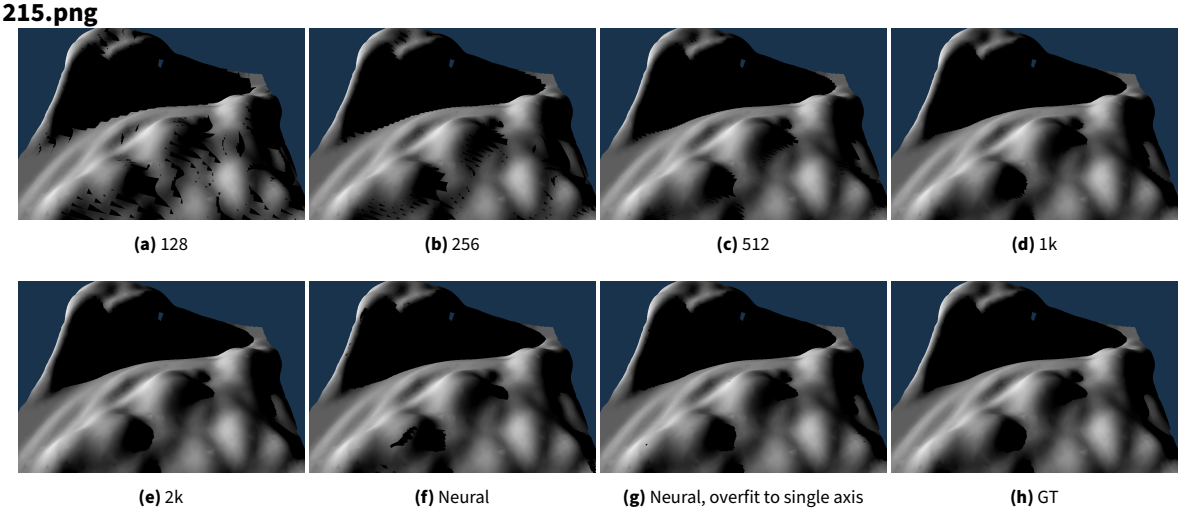
The landscape scene is easier to approximate than the dragon and fence, as the scene contains predominantly low-frequency geometry. As such, the neural representation achieves a higher visual quality than in the previous scene. Restricting the rotation of the lighting to a single axis further improves the quality, outperforming a 512×512 shadow map.

For both the single-axis and general neural model, a learning rate of 0.0001 was used, as this lower learning rate lead to better convergence for simple geometry. Both models were also evaluated with a dedicated set of 1000 viewpoints that are aimed at the landscape's center. Lighting in both scenes is restricted to one axis to allow for comparison between the two.

Method	**LPIPS** ↓	**PSNR** ↑	**SSIM** ↑
2k by 2k shadowmap	0.0060	51.24	0.9981
1k by 1k shadowmap	0.0122	48.06	0.9962
Our method (single axis)	0.0210	40.52	0.9936
512 by 512 shadowmap	0.0240	44.13	0.9924
Our method	0.0284	39.04	0.9876
256 by 256 shadowmap	0.0549	39.53	0.9836
128 by 128 shadowmap	0.1297	32.09	0.9612

Table 4.5: Comparison of the neural shadow representation and conventional shadow mapping on the landscape scene. Restricting the lighting domain to a single rotational axis substantially improves reconstruction quality.

215.png



(a) 128 **(b)** 256 **(c)** 512 **(d)** 1k

(e) 2k **(f)** Neural **(g)** Neural, overfit to single axis **(h)** GT

Figure 4.7: Comparison of shadow map resolutions, neural rendering, and ground truth across certain frames of the landscape model with single light direction axis.

Overall, the experiments show that the neural shadow representation is capable of approximating shadow maps with a quality roughly comparable to medium-resolution shadow maps while avoiding explicit rasterization artifacts. Performance depends strongly on the scene complexity, with the best results in low-frequency or restricted-lighting scenes.

4.4. Performance

4.4.1. Inference speed

Training the model described in Section 4.1 for 1,000,000 iterations takes approximately 3 hours and 20 minutes on the hardware described in Section 4.1.

Inference time depends both on the size of the MLP and on the portion of the viewport that is taken up by the projected bounding sphere. This is because inference is only run for pixels on the screen for which the ray from the corresponding shading point to the light intersects the object's bounding sphere.

In addition, the inference cost increases with the size of the input encoding, the number of light sources, and the number of objects that cast shadows. The inference speed is not directly dependent on the object's shape.

Different objects whose shadows are represented by MLPs of the same size will have the same inference speed, since forward inference consists of predictable dense matrix multiplications. Nevertheless, geometry with more complex structures can require larger MLPs to accurately represent its shadows compared to low-frequency or repetitive structures.

To investigate the trade-off between shadow quality and inference cost, we trained additional variants of the dragon and landscape models using 64 and 128 neurons per hidden layer, while network depth and other hyperparameters stay unchanged. Table 4.6 summarizes the resulting LPIPS scores, average frame times, and model sizes.

Scene	Layer width	Frametime	LPIPS	MLP Size
Dragon	64 neurons	11.5 ms	0.0894	160 KiB
Dragon	128 neurons	13.8 ms	0.0487	575 KiB
Dragon	256 neurons	29.1 ms	0.0379	2175 KiB
Landscape	64 neurons	13.2 ms	0.0279	160 KiB
Landscape	128 neurons	13.8 ms	0.0187	575 KiB
Landscape	256 neurons	32.8 ms	0.0210	2175 KiB

Table 4.6: Performance and memory usage for Dragon and Landscape models with different numbers of neurons per hidden layer.

Increasing the network width consistently improves shadow quality for the dragon model, albeit with diminishing returns. This is visualized in figure 4.8. A network width of 128 neurons significantly improves the reconstruction quality from an LPIPS of 0.0894 to 0.0487 while increasing the frametime by only 2.3 ms. However, a model width of 256 neurons achieves a quality of 0.0379 in terms of LPIPS. This is still significantly better than the 128-wide model, but the rendering time is more than doubled. Since MLPs are fully connected, the required number of matrix multiplications increases quadratically with network width. Furthermore, the small difference in frametime between the 64-wide and 128-wide models suggests that most of the rendering time for those models consists of overhead that does not scale with layer width, meaning that further decreasing the model width will also give diminishing returns.

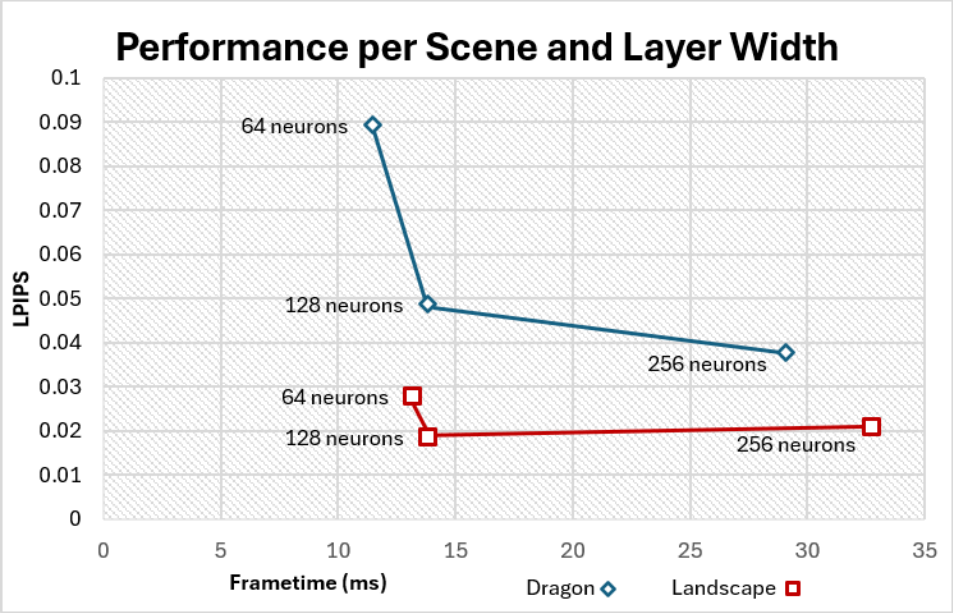


Figure 4.8: LPIPS and frametime for different numbers of neurons per hidden layer. Wider MLPs perform better on the dragon but suffer from overfitting on the landscape

The results for the landscape scene are drastically different. Most interestingly, visual quality decreases as the MLP width increases. This suggests that the model with 256 neurons per layer is too fit to the landscape training set, which could cause high-frequency shadows that do not exist in the landscape. As such, it is important to fit the model parameters to the scene that is to be represented.

4.4.2. Memory usage

Using 32-bit floats, the eight hidden layers in the MLP consist of $8 \times 256 \times 256$ = 524288 parameters. The implementation does not use a bias per layer. With ten frequencies, the input layer consists of 126×256 = 32256 parameters, and the output layer contains 256 parameters. Using 32-bit floats, the entire MLP takes up $4 \times (524288 + 32256 + 256)$ = 2,227,200 bytes or 2.12 MiB. Since MLPs are fully connected, the parameter space increases linearly with network depth and quadratically with layer width (Table 4.6).

In practice, the majority of GPU memory usage in our implementation is taken up by auxiliary buffers that store and pass data between OpenGL and CUDA. These buffers are not strictly required if the neural inference is closely integrated within the graphics pipeline.

Representation	**Memory**
2k by 2k shadowmap	16 MiB
1k by 1k shadowmap	4 MiB
Our method	2.12 MiB
512 by 512 shadowmap	1 MiB
256 by 256 shadowmap	256 KiB
128 by 128 shadowmap	64 KiB

Table 4.7: Analytic memory comparison using 32-bit floats for both neural network parameters and shadowmap texels.

5 Discussion

5.1. Limitations

Our method offers a drop-in replacement for shadow mapping with the continuous function space of ray tracing, allowing it to represent shadows with a quality comparable to medium-resolution shadow maps while avoiding discretization artifacts. However, the experiments also reveal several limitations that currently prevent the method from competing with traditional approaches.

The most significant limitation is the representation of high-frequency structures. This is especially clear in the mesh fence scene, where the model is unable to accurately represent the many sharp transitions caused by the thin mesh. While positional encoding does improve the method's ability to represent higher frequencies, it is not enough for this model. This suggests that either the network capacity is insufficient, or that the frequency encoding is not expressive enough for the model's complexity.

The occluder depth representation is also sensitive to self-shadowing artifacts. This is because any prediction slightly below the valid interval immediately results in a false positive for occlusion, producing self-shadowing artifacts. This is only the case for the lower bound; if the model predicts above the upper bound, then this only creates incorrect shadows when other geometry happens to be present between the erroneous prediction and the true upper bound. As such, the lower and upper bounds do not contribute equally to the visual error. This suggests that future representations should treat lower-bound violations more strictly than upper-bound violations.

Finally, rendering speed remains a challenge. Although inference relies on predictable dense matrix multiplications, the method remains slower than traditional shadow mapping approaches. The experiments show that smaller models drastically reduce rendering time, but at the cost of visual quality for objects containing high frequencies. As a result, the method shows a clear trade-off between quality and performance.

5.2. Advantages

Despite these limitations, the experiments also reveal situations in which neural shadow representations may be particularly effective. The experiments with the single-axis light rotation show that the model is able to direct its capacity towards restricted lighting configurations, something that traditional techniques cannot exploit. Aside from scenes illuminated by the sun rotating on a single trajectory, this property is useful in various other scenarios. Characters and vehicles are primarily illuminated from above, both in indoor scenes with ceiling lights and in outdoor scenes with street lighting. Instead of a full sphere, the input domain can be restricted to a hemisphere or an even smaller angular region if the scene size limits the maximum angle between the light and the object.

Furthermore, the required MLP size for quality shadows scales with geometry complexity, and not with geometry size. This makes the method most suitable for large, low-frequency geometry such as terrains, whereas shadow maps need to be of large resolution regardless of how complex they are. In the case of landscapes, our method also benefits from large bounds during supervision. This is because landscapes can be modeled as a

2D manifold, causing most of the light rays coming from above to have just one intersection with the geometry. This allows our method to predict any value above the lower bound and still generate correct shadows.

Another advantage is the nature of the neural artifacts. Although our method produces errors, these artifacts tend to be less structured than shadow mapping's discretization artifacts. Stair-stepping and aliasing patterns are highly regular and therefor very noticeable, whereas the neural artifacts are less regular. In natural environments, such as natural terrains and foliage, these artifacts could blend into the scene and be perceived as less distracting.

5.3. Implications

The results suggest that the method is unlikely to immediately replace shadow maps in fully dynamic real-time applications. In particular, scenes containing thin geometry, sharp edges, or other high-frequency characteristics remain challenging for the proposed method. These types of features are most common in man-made environments.

At the same time, the results indicate that there are applications where neural shadows may be competitive with traditional shadowing methods. The most promising example is outdoor scenes. These scenes often contain large-scale and low-frequency landscapes, and they are commonly illuminated by the sun, which follows a restricted path. Both of these properties align with the strengths of the neural method. Furthermore, traditional shadow maps struggle with outdoor scenes as the large scale requires shadow maps with a large resolution, especially in directional lighting.

5.4. Future work

The results show that neural shadow representations can approximate shadows with the level of medium-resolution shadow maps, but they also highlight several limitations, as discussed in Section 5.1. The future work can be divided into two directions. The first direction focuses on improving the performance and quality of the current method, and the second direction explores extensions that are made possible by the neural representation of our method.

5.4.1. Improving quality and performance

In neural models, quality and performance are closely connected. Increasing the size of the network generally improves prediction quality at the cost of inference speed, while reducing its size improves the performance at the cost of accuracy. As such, improvements that make the representation more efficient can often be traded for higher quality, and vice versa.

High frequencies

To improve the model's performance in representing high frequencies, the problem size can be reduced by splitting the geometry into several smaller parts using a bounding volume hierarchy. This allows the models to focus on local detail. Similar to NIF [8], the BVH could consist of only a few layers while the rest is represented by MLPs. Additionally, other encodings can be explored. While our method uses frequency encoding, recent methods have demonstrated the effectiveness of hashgrid encodings, such as Instant NGP [12], which reduces the required MLP size for varying problems without sacrificing quality using a multiresolution hash table of trainable feature vectors.

Improved loss

Section 5.1 showed that lower-bound violations are more likely to introduce incorrect shadows than upper-bound violations. It is worth exploring asymmetric loss functions that assign greater punishment to errors under the lower bound, as these types of errors are guaranteed to create false shadows. Additionally, the loss function could have a small gradient within the bounds that slightly encourages predictions towards the middle. A more drastic improvement could be made by having the model output a confidence value, where shading points will only be shadowed if they both fail the depth test and if the confidence value is larger than 0.5. This allows the model to use the advantages of both the depth-represented model and the confidence model: the depth-representation allows the input to be restricted to points on a single sphere, while the confidence value allows the model to represent discontinuities with a continuous function by transitioning the confidence value smoothly from below to above 0.5, so discontinuities can be approximated with a continuous function.

Directional lighting

When rendering scenes with directional lighting, a feature vector can be calculated from the light direction. This requires two neural networks: one network that calculates feature vectors from light directions and one network that predicts occluder depth from feature vectors and pixel locations. This allows one part of the computation to be calculated per frame instead of per pixel.

Exploiting symmetry

The problem of predicting occluder depth within a bounding sphere is inherently symmetric; every ray has an oppositely directed ray that intersects the occluder in the same positions. However, rather than explicitly forcing this symmetry, our method relies on the neural network to learn this symmetry implicitly. This allows us to represent non-hitting rays by assigning them a depth outside of the bounding sphere. Although negative depths can also be used to represent non-hitting rays, this approach is more prone to self-shadowing artifacts. Due to the continuous nature of neural networks, predictions near the occluder boundary will interpolate between the occluder depth and negative values, which is more likely to lead to self-shadowing than interpolation between occluder depth and high values.

5.4.2. Extensions

Neural shadow representations enable extensions that can be incorporated more naturally than with traditional techniques. Because the shadow function is represented with a learned model, additional parameters can be incorporated directly into the network inputs, allowing a single MLP to represent more complex configurations.

Soft shadows

Similar to ray-traced soft shadows, our method can be used to render soft shadows by performing multiple inferences per view-space pixel, where each inference will be performed on a ray pointing towards a different sample on an area light. However, this approach scales linearly with the number of samples.

Instead, soft shadows could be incorporated into the neural representation. Unlike hard shadows, soft shadows cannot be represented with binary visibility. Thus, a soft shadow representation should not output the depth of a shadow boundary, but should output the light intensity. However, shadows of an area light are much more difficult to learn than hard shadows. This is because hard shadows depend only on the occluder and the direction from the light to the occluder, while soft shadows also depend on the rotation of the area light, the translation between the occluder and the area light, and the translation from the shading point to the occluder. Since both of these factors can change in dynamic scenes, a neural representation would need to have many input parameters, of which the translation is a fundamentally unbounded input.

Although the general soft-shadow problem is challenging, several specific cases may be well suited for neural representations. One example is directional soft shadows, which are commonly used to approximate sunlight. While directional soft shadows do not exist in theory, the sun is sufficiently far away that the translation to the scene can be considered constant, while simultaneously taking up a non-negligible area in the sky. In such cases, the required shadow function representation is not too different from that of hard shadows. The direction from the area light to the occluder defines a 3D field of light intensity, which can be sampled with the 3D position of the shading point relative to the occluder:

$$f : (\mathbf{p}, \mathbf{d}) \rightarrow I \in [0, 1]$$

In this case, $\mathbf{p} \in \mathbb{R}^3$ is not restricted to lying on a sphere, but is instead any unbounded 3D position. The function does not output shadow boundary depth, but directly outputs light intensity.

Animation

Another interesting extension involves animated geometry. Given a 3D animation consisting of several frames, the index of the frame can be used as an additional parameter to the neural representation. This allows the single model to predict shadows for an entire animation. This approach could be suitable for neural methods when consecutive frames are highly correlated, as this would allow the model to exploit temporal similarity. One example where this representation could be useful are movement animations that play when a video game character performs a certain action.

6

Conclusion

This thesis introduced a continuous and global mapping from rays to occluder depth to render real-time shadows, implemented with a neural network. By directly mapping a ray's origin and direction to a depth value, the method does not rely on discretized intermediate representations, eliminating the aliasing artifacts that shadow maps suffer from.

To help the model focus its learning capacity on the occlusion boundaries, a training strategy was proposed. The loss function uses a lower and upper target per sample and only assigns a loss outside of these bounds. These bounds are set to correspond to the first two ray-geometry intersections, allowing the model to represent occluder depth as any value within this interval.

Experiments show that the model can represent both low- and high-frequency geometry, achieving a visual quality comparable to 512×512 traditional shadow maps. Additionally, the combination of quad-tree based importance sampling and Poisson Disk sampling significantly improve reconstruction quality by focusing rays on shadow boundaries and ensuring full coverage. The results also show that the model has the unique ability to take advantage of limited light directions, as it can naturally overfit to restricted domains and predict shadows of higher visual quality than the general model. This makes the model well-suited for large outdoor environments, which often contain low-frequency geometry and illumination from a restricted light path.

Nevertheless, critical areas for improvement remain. The results show that the neural network fails to capture high-frequency details such as thin geometry and that prediction errors can lead to self-shadowing artifacts. Furthermore, optimizations are needed for the inference speed to be comparable with shadow mapping.

Ultimately, this thesis showed that neural fields can serve as an effective replacement for traditional shadowing techniques. Although optimizations are needed to compete with traditional methods, neural methods have several unique strengths that could make them especially well-suited for specific scenarios.

7

Reflection

7.0.1. Research in broader CS field

This research was motivated by a broader trend in computer graphics to replace traditional data structures with neural representations. While many computer graphics techniques rely on explicit representations such as textures and meshes, recent developments have shown that neural networks can act as compact representations of complex data.

Within this context, it is natural to explore neural methods for shadow casting as the most common method, shadow mapping, suffers from discretization artifacts whereas neural representations don't have to rely on discretization at all. The results show that it is possible for a neural representation to achieve the visual quality of medium-sized shadow maps while eliminating the typical staircase artifacts that stem from shadow maps. At the same time, the neural method takes more time to render shadows while struggling to capture high frequencies and prevent unwanted self-shadowing.

As such, the method presented in this thesis is not directly competitive with existing methods. However, the experiments do highlight specific strengths and weaknesses. While further optimizations are required, the neural method does have unique strengths that make it especially useful for specific scenarios such as outdoor environments. So, aside from introducing several new approaches such as the neural occluder-depth representation, the thesis also shows the potential of neural methods for shadow casting. This could motivate new research into variants and optimizations of this neural shadowing method.

7.0.2. Implications and applicability

The proposed method is particularly useful for developers of games and other interactive 3D applications. Unlike traditional shadow rendering techniques, neural shadow representations can be trained offline and distributed along with the application. In principle, this moves computational cost from consumer hardware to developers, where specialized hardware and longer computing times are often available.

However, the results also suggest that the method is not yet suitable as a general replacement for shadow mapping. The rendering costs are still too high and the model still produces artifacts in scenes with thin geometry or high-frequency structures. Neural methods often cannot guaranty complete accuracy as they are deterministic methods, and it is impossible to test every sample in a continuous domain. As such, traditional deterministic methods such as ray tracing remain preferable in scenarios where exact visibility is required, or where there is no strict requirement for fast computation time. An example is measuring shadows for optimal solar panel placement.

Furthermore, it is worth saying that the proposed method primarily competes with existing techniques in interactive applications. In static or partially static scenes, shadows can often be baked into textures or other precomputed representations, making real-time shadow rendering unnecessary. Similarly, in precomputed animations, shadow rendering does not have to be performed in real time, allowing computationally expensive techniques such as ray tracing to be used for rendering.

Nevertheless, neural methods still offer some advantages in these scenarios. While pre-baked shadows are

less prone to aliasing than light-space shadow maps, they are still discrete representations that can require large textures to prevent oversampling for large landscapes. Neural alternatives could reduce memory usage and allow continuous shadows here. Additionally, neural methods could also benefit from static lighting, as the results in this thesis have shown that the model naturally overfits to restricted lighting configurations.

Ultimately, the proposed method is most useful for interactive applications where exact visibility is not required. The method is able to move computational power from the consumer to the developer. This saves computational requirements from many general-purpose machines to a select number of specialized computers, drastically reducing the cumulative number of computations.

8

AI Usage

Generative AI tools were used during this project to support with writing, programming, and finding information. Its use was limited to assistance and had no role in the core research, methodology, and analysis of the thesis.

During writing, the LLM's ChatGPT and NotebookLM were used to provide feedback on draft texts. This includes suggestions in grammar, structure, or for clarity. While the LLM's suggested better phrasings at times, no writing was directly copied and pasted from an LLM's output, aside from tables that were reformatted from script output to LaTeX. All content is thoroughly reviewed and remains my responsibility.

LLM's occasionally helped with finding references and debugging code. All references in this thesis are verified and read before use. LLM's were also used to generate python scripts for running experiments or parsing results. These tools were not used to generate large parts of the source code, but they were used to generate examples of library usage, where it was not clear to me from the library's documentation.

I retain full responsibility for the accuracy, validity, originality, and reliability of all content included in my work. All generative AI output was critically evaluated.

References

[1] Markus Billeter, Ola Olsson, and Ulf Assarsson. “Efficient stream compaction on wide SIMD many-core architectures”. In: Aug. 2009, pp. 159–166. DOI: 10.1145/1572769.1572795.

[2] Jakub Boksansky, Michael Wimmer, and Jiří Bittner. “Ray Traced Shadows: Maintaining Real-Time Frame Rates”. In: *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs*. Ed. by Eric Haines and Tomas Akenine-Möller. Apress, 2019.

[3] Robert Bridson. “Fast Poisson Disk Sampling in Arbitrary Dimensions”. In: *SIGGRAPH Sketches*. 2007.

[4] Ran Chen et al. “Real-time neural soft shadow synthesis from hard shadows”. In: *Graph. Models* 141.C (Jan. 2026). ISSN: 1524-0703. DOI: 10.1016/j.gmod.2025.101294. URL: https://doi.org/10.1016/j.gmod.2025.101294.

[5] Robert L. Cook. “Stochastic Sampling in Computer Graphics”. In: *ACM Transactions on Graphics* 5.1 (1986), pp. 51–72.

[6] Sayantan Datta et al. “Neural Shadow Mapping”. In: *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings*. SIGGRAPH ’22. ACM, Aug. 2022, pp. 1–9. DOI: 10.1145/3528233.3530700. URL: http://dx.doi.org/10.1145/3528233.3530700.

[7] Raphael Finkel and Jon Bentley. “Quad Trees: A Data Structure for Retrieval on Composite Keys.” In: *Acta Inf.* 4 (Mar. 1974), pp. 1–9. DOI: 10.1007/BF00288933.

[8] Shin Fujieda, Chih Chen Kao, and Takahiro Harada. “Neural Intersection Function”. In: *High-Performance Graphics - Symposium Papers* (2023), pp. 43–53. DOI: 10.2312/HPG.20231135. URL: https://diglib.eg.org/handle/10.2312/hpg20231135.

[9] Stanford University Computer Graphics Laboratory. *The Stanford 3D Scanning Repository: The Dragon*. https://graphics.stanford.edu/data/3Dscanrep/. Accessed: 2026-06-22.

[10] Ben Mildenhall et al. *NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis*. 2020. arXiv: 2003.08934 [cs.CV]. URL: https://arxiv.org/abs/2003.08934.

[11] Thomas Müller. *tiny-cuda-nn*. Version 2.0. Apr. 2021. URL: https://github.com/NVlabs/tiny-cuda-nn.

[12] Thomas Müller et al. “Instant Neural Graphics Primitives with a Multiresolution Hash Encoding”. In: *CoRR* abs/2201.05989 (2022). arXiv: 2201.05989. URL: https://arxiv.org/abs/2201.05989.

[13] Nasim Rahaman et al. *On the Spectral Bias of Neural Networks*. 2019. arXiv: 1806.08734 [stat.ML]. URL: https://arxiv.org/abs/1806.08734.

[14] Daniel Scherzer, Michael Wimmer, and Werner Purgathofer. “A Survey of Real-Time Hard Shadow Mapping Methods”. In: *Comput. Graph. Forum* 30 (Mar. 2011), pp. 169–186. DOI: 10.1111/j.1467-8659.2010.01841.x.

[15] Ayush Tewari et al. *State of the Art on Neural Rendering*. 2020. arXiv: 2004.03805 [cs.CV]. URL: https://arxiv.org/abs/2004.03805.

[16] Karthik Vaidyanathan et al. “Random-Access Neural Compression of Material Textures”. In: *ACM Trans. Graph.* 42.4 (July 2023). ISSN: 0730-0301. DOI: 10.1145/3592407. URL: https://doi.org/10.1145/3592407.

[17] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[18] Philippe Weier et al. “N-BVH: Neural ray queries with bounding volume hierarchies”. In: *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers 24*. SIGGRAPH ’24. ACM, 2024, pp. 1–11. DOI: 10.1145/3641519.3657464. URL: http://dx.doi.org/10.1145/3641519.3657464.

[19] Daniel Weiskopf and Thomas Ertl. “Shadow Mapping Based on Dual Depth Layers”. In: *Eurographics 2003 - Short Presentations*. Eurographics Association, 2003. DOI: 10.2312/egs.20031069.

[20] Lance Williams. “Casting Curved Shadows on Curved Surfaces”. In: *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH ’78)*. 1978, pp. 270–274.

[21] Alex Yu et al. *PlenOctrees for Real-time Rendering of Neural Radiance Fields*. 2021. arXiv: 2103.14024 [cs.CV]. URL: https://arxiv.org/abs/2103.14024.

[22] Fan Zhang et al. “Parallel-split shadow maps for large-scale virtual environments”. In: VRCIA ’06. Hong Kong, China: Association for Computing Machinery, 2006, pp. 311–318. ISBN: 1595933247. DOI: 10.1145/1128923.1128975. URL: https://doi.org/10.1145/1128923.1128975.

[23] Richard Zhang et al. “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric”. In: *CoRR* abs/1801.03924 (2018). arXiv: 1801.03924. URL: http://arxiv.org/abs/1801.03924.