Neural Prefiltering for Correlation-Aware Levels of Detail

PHILIPPE WEIER, Saarland University, Germany and DFKI, Germany
TOBIAS ZIRR, Intel Corporation, Germany
ANTON KAPLANYAN, Intel Corporation, USA
LING-QI YAN, University of California, Santa Barbara, USA
PHILIPP SLUSALLEK, Saarland University, Germany and DFKI, Germany

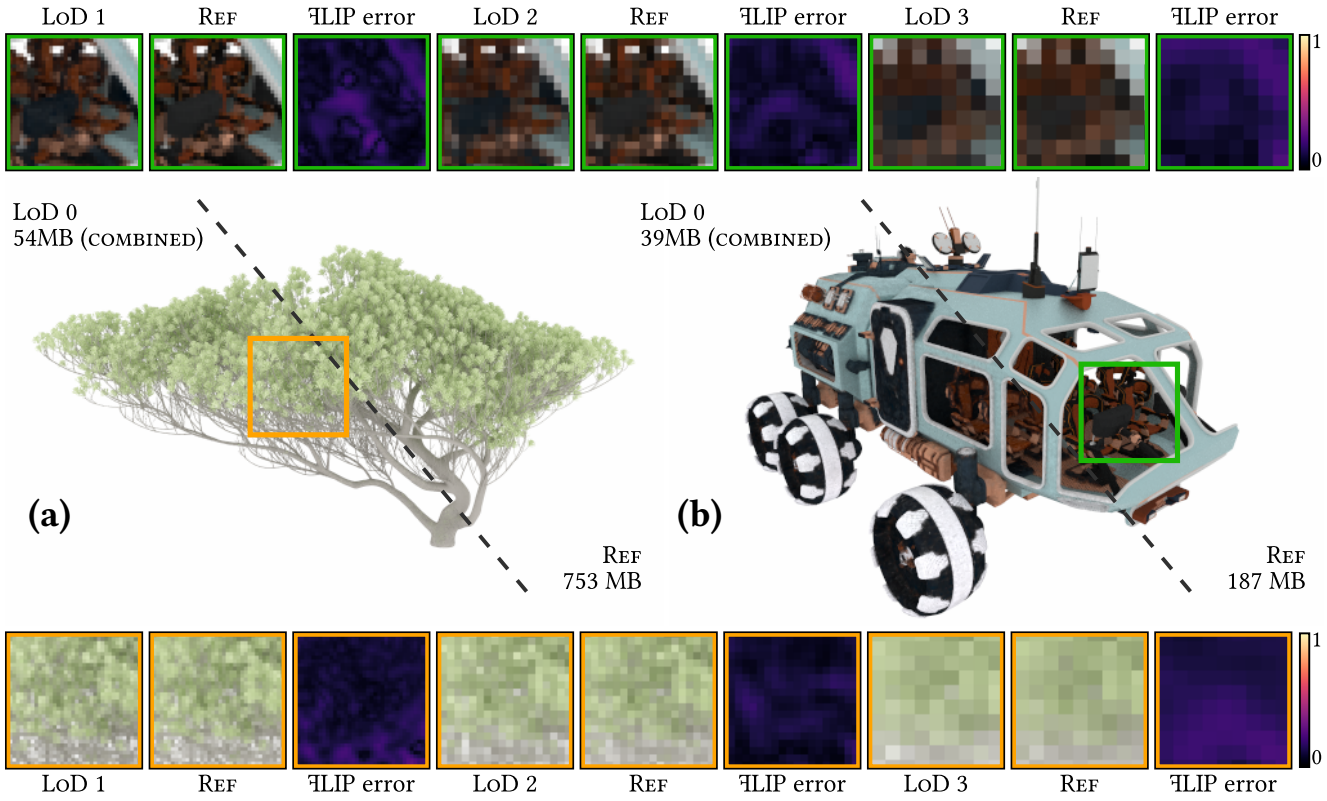

Fig. 1. Our physically-based prefiltering technique can accurately represent details at various scales, both for unstructured **(a)** and correlated (structured) geometry **(b)**. We demonstrate this by rendering two assets at various levels of detail (LoD) with our neural approach. The highest resolution is rendered at 512 x 512 pixels, while insets correspond to lower resolution LoDs rendered at the appropriate resolution, the storage includes the cost of LoD 0-6. Our neural pipeline approximates the full intra-voxel light transport to achieve high compression rates and enable predictable rendering of a wide range of assets without any prior assumptions on materials or geometry.

We introduce a practical general-purpose neural appearance filtering pipeline for physically-based rendering. We tackle the previously difficult challenge of aggregating visibility across many levels of detail from local information only, without relying on learning visibility for the entire scene. The high adaptivity of neural representations allows us to retain geometric correlations along rays and thus avoid light leaks. Common approaches to prefiltering decompose the appearance of a scene into volumetric representations with physically-motivated parameters, where the inflexibility of the fitted models limits rendering accuracy. We avoid assumptions on particular types of geometry or materials, bypassing any special-case decompositions. Instead, we directly learn a compressed representation of the intra-voxel light transport. For such high-dimensional functions, neural networks have

Authors' addresses: Philippe Weier, Saarland University, Saarland Informatics Campus, Saarbrücken, Germany and DFKI, Germany, weier@cg.uni-saarland.de; Tobias Zirr, Intel Corporation, Karlsruhe, Germany, tobias.zirr@intel.com; Anton Kaplanyan, Intel Corporation, Mercer Island, USA, anton.kaplanyan@intel.com; Ling-Qi Yan, University of California, Santa Barbara, Santa Barbara, USA, lingqi@ucsb.edu; Philipp Slusallek, Saarland University, Saarland Informatics Campus, Saarbrücken, Germany and DFKI, Germany, philipp.slusallek@dfki.de.

0730-0301/2023/8-ART
https://doi.org/10.1145/3592443

proven to be useful representations. To satisfy the opposing constraints of prefiltered appearance and correlation-preserving point-to-point visibility, we use two small independent networks on a sparse multi-level voxel grid. Each network requires 10–20 minutes of training to learn the appearance of an asset across levels of detail. Our method achieves 70–95% compression ratios and around 25% of quality improvements over previous work. We reach interactive to real-time framerates, depending on the level of detail.

CCS Concepts: • **Computing methodologies → Ray tracing**.

Additional Key Words and Phrases: levels of detail, prefiltering, precomputed light transport, neural representations, physically-based rendering

ACM Reference Format:
Philippe Weier, Tobias Zirr, Anton Kaplanyan, Ling-Qi Yan, and Philipp Slusallek. 2023. Neural Prefiltering for Correlation-Aware Levels of Detail. *ACM Trans. Graph.* 42, 4 (August 2023), 16 pages. https://doi.org/10.1145/3592443

1 INTRODUCTION

Recent advances in physically based rendering enabled a growing base of users to create realistic lifelike images and animations, resulting in an increasing demand for the efficient handling of complex virtual scenes in the rendering of highly detailed, physically based appearance. Rendering with uniform convergence remains the main challenge, where it is desired to avoid unnecessary sampling in pixels with simple content, yet avoid excessive latency for complex pixels containing, e.g., vegetation or hair that requires more extensive light transport simulation. A holy grail of rendering is a method with *constant rendering cost* per pixel, regardless of the underlying scene complexity. Moreover, while graphics processing units (GPUs) have allowed for acceleration and efficiency improvements in many parts of rendering algorithms, their limited amount of onboard memory limit practical rendering of complex scenes, making compression increasingly important.

The family of level of detail (LoD) techniques offloads parts of the complexity handling in rendering to preprocessing. In part, this can happen by identifying and pruning any detail with negligible impact on the final image. However, such simplification techniques are prone to noticeable changes in the frequency content of resulting images, causing an undesirably flat or artificial appearance. Advanced prefiltering techniques tackle this problem by preresolving or fitting parts of the appearance, e.g., resulting from fine geometric details aggregating within a pixel. Thus, the detailed appearance of complex assets can often be preserved, while reducing the amount of data to be stored and processed at render time.

In physically based rendering, preaggregation of appearance properties not only reduces the computational effort of shading samples, but also translates to faster rendering convergence in final pixels. However, prefiltering often creates noticeable new problems, particularly when combining the appearance of many prefiltered components in larger scenes. A common limitation of previous work is the loss of accurate visibility for rays crossing separately prefiltered regions. Combining the appearance of multiple prefiltered regions is often ambiguous when tracking long-range visibility with geometric structure spread across multiple regions.

One robust solution to this problem relies on retaining visibility correlations in the distribution of prefiltered geometry. So far, such solutions either require hand-crafted heuristics, or rely on expensive non-local analysis of global scene structure. We propose a correlation-preserving method that aggregates only local intra-regional visibility. We leverage the high adaptability of recent multiscale neural representations to minimise the loss of spatial localisation for visibility tracking.

Visibility is an important aspect of prefiltering, where our approach provides improvements by using recent advances in neural representations. Moreover, we also employ neural representations to fit the high-dimensional function of appearance. Thus, we manage to achieve unprecedented scene compression rates (up to 96% space savings for the most complex assets), while controlling the computational cost with compact, fixed-size neural building blocks without sacrificing image quality (see Fig. 1). Moreover, our pipeline achieves interactive framerates even at high image resolutions. Concretely, our main contributions are:

- A modular prefiltering technique that is compatible with existing rendering systems and pipelines, with a scalable appearance decomposition based only on prefiltering local information within a single voxel (Section 3),
- A compact fixed-size neural representation for accurate prefiltered appearance within individual voxels, without prior assumptions about the underlying materials (Section 4.2),
- A visibility classifier that scales well from uncorrelated volume-like unstructured geometry to correlated surface-like geometry with strict structure and space separation (Section 4.3),
- A method for increasing the capacity of binary neural fields by way of computing per-voxel optimal classification thresholds (Section 4.4).

The source code of our implementation is publicly available at https://github.com/WeiPhil/neural_lod.

2 RELATED WORK

Our technique draws on insights from many long-studied fields of computer graphics, as well as recent advances in neural graphics.

Geometric Level of Detail. Modern production rendering systems employ many mesh simplification algorithms. Silhouette and shape-preserving decimation techniques [Garland and Heckbert 1997; Kobbelt et al. 1998; Puppo and Scopigno 1997] are widely used, often with artist supervision, as is stochastic decimation [Cook et al. 2007], where remaining triangles are re-scaled and re-shaded to preserve the appearance at different scales. Such algorithms, while approximate, lead to fast and practical solutions suitable for large-scale production scenes. To reduce loss of faithful appearance and necessary artist supervision, solutions with automated quality evaluation in the loop include an automatic neural mesh simplification pipeline [Potamias et al. 2022] and a differentiable mesh simplification pipeline [Hasselgren et al. 2021]. While the latter pipeline needs an initial guess for the simplified geometry, it can jointly optimise for the shape and material parameters using an image-based loss function.

Macroscopic Aggregation of Details. Approximate macroscopic aggregate appearance implicitly occurs in many rendering techniques. Efficient real-time global illumination methods [Crassin et al. 2011; Ritschel et al. 2012] employ approximation on coarse sample sets

or proxy representations of a virtual scene. For high-quality appearance of directly visible aggregate high-frequency detail, Heitz et al. [2012] divide scene surfaces at multiple scales into a sparse voxel octree with local geometry and aggregate distribution information. Improved distributions for matching the resulting shading with the appearance of initially aggregated details, such as the SSGX microflake normal [Heitz et al. 2015] and shadowing [Loubet and Neyret 2018] distributions, have enabled efficient appearance-preserving downsampling for many complex aggregate phenomena such as foliage and hair-like fibres. However, these methods motivated by volumetric representations are limited in representing the aggregation of simpler opaque structures such as angled connections of walls or crossings of beams, where the structural correlation of visibility is not tracked across multiple regions, resulting in a loss of spatially correlated appearance characteristics that are apparent by their view-dependent visibility or occlusion.

In order to improve on the preservation of macroscopic structured geometry, Loubet et al. [2017] classify the geometry into macro- and micro-surfaces. While this separation allows them to apply the appropriate simplification techniques separately, they rely on a binary distinction between volume and mesh that may be inaccurate in practice, where a continuous mixture of the two in the same region is common across scales. Vicini et al. [2021] cover this spectrum by introducing a non-exponential transmittance model with a continuous parameter defining the degree of uncorrelated volumetric-like vs correlated surface-like behavior in a region. One drawback of fitting this model is the requirement of a costly *non-local* optimisation to account for correlated behavior across larger stretches of connected voxels for many directions. Similarly to previous work, their shading model relies on fitting SGGX distributions, limiting generalisation to a wider range of materials.

Precomputed Light Transport. Preprocessing of a scene to capture the local light transport operator, which distributes any incident light to the reflected radiance resulting from many-bounce indirect light scattering, is a popular way [Lehtinen 2007; Sloan et al. 2002] of allowing fast shading evaluation for changing light conditions at the time of rendering. Many variations of such techniques have been developed, some with a focus on composability (tabulated diffuse scattering [Blumer et al. 2016], density-informed neural predictions [Kallweit et al. 2017]). Neural representations have generally proven a good fit for capturing high-dimensional transport functions of entire scenes [Rainer et al. 2022; Ren et al. 2013], around objects [Lyu et al. 2022] or even spatially varying BRDFs including strong parallax effects [Kuznetsov et al. 2021, 2022]. We expand on their modularity by decomposing the transport around a prefiltered asset into chained neural functions at various scales, to arrive at a compressible implicit neural representation of multiscale appearance.

Appearance Prefiltering. Pre-resolving the appearance of microgeometry and materials for fast run-time evaluation has been subject to extensive study. Bruneton and Neyret [2011] survey suitable approaches for many common phenomena ranging from smoothed reflections due to bumpy surfaces, on to colour shifts due to microsurface occlusions, to robust high-frequency shadowing, and finally to generic tools for decomposition into (classic) filterable representations. Applying a specially-crafted combination of many such classic representations for far-field appearance of macro-scale geometry, realistic vegetation coverage with robust transitions from closeups to planetary-scale overviews has been presented [Bruneton and Neyret 2012]. Preresolved glinty highlights due to easily missed meso-scale geometry as represented in high-resolution normal maps [Yan et al. 2014] and compression thereof [Deng et al. 2022] using tailored decompositions has been effective using classic approaches. Neural building blocks proved effective for less narrowly scoped problems such as the aggregate appearance of fur [Zhu et al. 2022], multi-layer surfaces [Wang et al. 2022a], and many-frequency shading phenomena [Xu et al. 2022].

Implicit Neural Representations. Neural Radiance Fields (NeRF) [Mildenhall et al. 2020] and recent adaptations to interactive and real-time graphics [Müller et al. 2022] are related appearance capturing methods using implicit neural representations forming neural fields [Xie et al. 2022] for novel view synthesis, where the high adaptability of neural representations paired with sparse [Chan et al. 2021; Müller et al. 2022] and compressed [Takikawa et al. 2022] coordinate encodings has proven effective for the efficient storage of high-dimensional functions, representing high-frequency content in the spatial and angular domain. We base our neural representations on the hash-grid encoding and fused MLP implementation presented by Müller et al. [2022]. We complement previous multi-scale neural representations designed for grid-free encodings [Barron et al. 2021, 2022], by leveraging implicit encoding of scale when accounting for the voxel geometry of parametric grid encodings. Neural reflectance fields for texturing [Baatz et al. 2021] capture lower-dimensional volume-like appearance similarly to previously discussed classic representations [Heitz et al. 2015; Loubet and Neyret 2018]. We similarly employ neural fields to represent visibility and reflectance information, but retain long-range correlated visibility even at coarse levels of detail. Furthermore, we capture appearance directly from individual light transport path samples rather than indirectly learning from pre-generated image data sets. Since these indirections in training are unnecessary for our work, we limit this section and refer an interested reader to a recent survey [Tewari et al. 2022] discussing such decompositions of NeRF-like representations for the purpose of re-lighting or altering real-world captured appearance.

Neural Level of Details. Recently, Bako et al. [2023] proposed a deep appearance prefiltering method that, similarly to our method, handles both surfaces and volumes, while accounting for surface correlations. Their approach relies on the training of albedo, phase function, phase slice, and visibility mask for a total of four encoder-decoder networks. Their representation is then encoded into a per-voxel latent vector to be decoded within a volumetric beam-tracer at render time. For accurate reconstructions, they store 256 scalars per voxel (~1KB) and report a training time of 0.5–2 days on a cluster of 256 NVIDIA V100 GPUs. In contrast, our approach only requires around seven scalars per voxel to be stored on average and training our representation with two MLPs and resp. feature grids for each requires 10–20 minutes on a single consumer GPU.

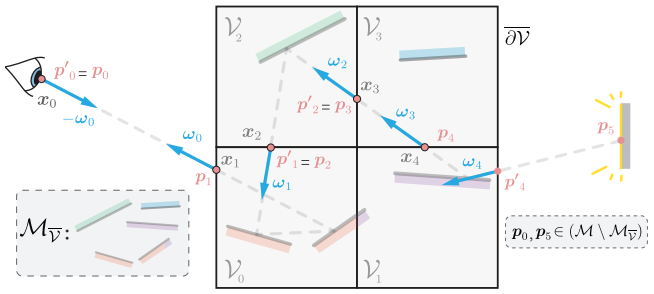

Fig. 2. Our generalised voxel path space is defined on a manifold $\mathcal{M}'$ where only voxel boundaries and non-voxelised geometry exist as scattering surfaces in light transport. We indicate the aggregated transport on the original manifold $\mathcal{M}$ replaced by $\mathcal{M}'$ and illustrate a path of length 4 where three vertices are voxel scattering events, i.e. $\boldsymbol{c}_j = 1$, and one interior vertex $\boldsymbol{x}_3$ is a null-scattering event, i.e. $\boldsymbol{c}_3 = 0$.

3 VOXEL-BASED TRANSPORT DECOMPOSITION

To preresolve and compress light transport within regions of varying detail and scale, we subdivide the space around prefiltered assets with a voxel grid (Section 3.1). We split up the path integral as introduced by Veach [1997] accordingly, such that all light transport on subpaths contained by individual voxels can be integrated separately. This leads to a clear distinction between the *intra-voxel transport*, representing light transported inside a single voxel, and the *inter-voxel* transport, representing the light transport inbetween voxels and interactions with other non-voxelised parts of the scene (Section 3.2). For appearance prefiltering techniques, analysing the decomposed transport allows us to understand the approximations introduced by a typical far-field assumption (Section 3.3). Our analysis leads to a natural separation of the rendering process into an appearance and a visibility component, which we approximate with corresponding neural representations (Section 4).

3.1 Generalised Voxel Path Space

The path space $\mathcal{P}$ contains all transport paths $\bar{\mathbf{p}} = (\boldsymbol{p}_0, ..., \boldsymbol{p}_{k-1}) \in \mathcal{P}$ connecting light sources at points $\boldsymbol{p}_{k-1}$ to a camera at $\boldsymbol{p}_0$ via any number of interactions at inner vertices $\boldsymbol{p}_j$ [Veach 1997]. Given a path throughput function $T(\bar{\mathbf{p}})$, a camera sensitivity $W(\bar{\mathbf{p}})$ and the light emission function $L(\bar{\mathbf{p}})$ the path integral computing pixel values I_j is defined as:

$$I_j = \int_{\mathcal{P}} W(\bar{\mathbf{p}}) T(\bar{\mathbf{p}}) L(\bar{\mathbf{p}}) d\mu(\bar{\mathbf{p}}), \quad \mathcal{P} = \bigcup_{k=2}^{\infty} \mathcal{M}^k,$$
$$\mu(P) = \sum_{k=2}^{\infty} \mu^{(k)}\left(P \cap \mathcal{M}^k\right), \quad d\mu^{(k)}(\bar{\mathbf{p}}) = \prod_{j=0}^{k-1} dA(\boldsymbol{p}_j).$$

where $\mathcal{M}^k$ denotes the set of scene surfaces concatenated k times for paths of length k and the product measure $\mu^{(k)}(P)$ multiplies the standard area measure A on $\mathcal{M}$ to measure subsets of paths $P \subseteq \mathcal{P}$.

In order to split light transport into clearly separated regions, we subdivide the space around prefiltered assets into a grid of m disjoint voxels, where each voxel $\mathcal{V}$ represents a virtual volume with boundary $\partial\mathcal{V}$. We then extend the path space $\mathcal{P}$ to include inter-voxel interactions that lie on the voxel boundary $\partial\mathcal{V}$, by way of adding null-scattering interactions, as typically used when modelling inter-volume transitions [Woodcock 1965] (illustrated in Fig. 2). This extension of $\mathcal{M}$ and thus $\mathcal{P}$ does not alter the value of the path integral computing pixel values I_j. In summary, intra-voxel vertices belong to the original scene manifold $\mathcal{M}$ while inter-voxel connections that cross voxel boundaries lie on $\partial\mathcal{V}$.

In order to describe the full light transport based on the pre-aggregated transport within voxels, we define the manifold $\mathcal{M}'$ that replaces voxelised geometry with the voxel grid, given by the union of all the surfaces inside voxels defined as $\mathcal{M}_{\overline{\mathcal{V}}} = \bigcup_{i=1}^{m} (\mathcal{M} \cap \mathcal{V}_i)$ and the union of all boundaries $\overline{\partial\mathcal{V}} = \bigcup_{i=1}^{m} \partial\mathcal{V}_i$, respectively:

$$\mathcal{M}' = \left(\mathcal{M} \setminus \mathcal{M}_{\overline{\mathcal{V}}}\right) \cup \overline{\partial\mathcal{V}}.$$

In order to define a path space on this manifold, it is easiest to keep track of the correct light directions along a path directly, rather than by the relative location of successive path vertices (which coincide on inner voxel boundaries). Therefore, we construct our generalised voxel path space $\mathcal{X}$ on an angular path parameterisation. However, when replacing any intra-voxel scattering on $\mathcal{M}_{\overline{\mathcal{V}}}$ by voxel boundary scattering events, we need to account for an added degree of freedom arising from the fact that scattering may change not only the direction but also the position of light passage at voxel boundaries. For such scattering, we extend a respective vertex $\boldsymbol{x}_j \in \mathcal{X}$ to comprise both an incident light direction $\boldsymbol{\omega}_j$ and a position of pre-scattering incidence $\boldsymbol{p}'_j \in \partial\mathcal{V}$. Adapting ideas from Jakob [2013], we separate the possible types of scattering events using a configuration indicator variable $\boldsymbol{c}$, that is, $\boldsymbol{x}_j = \boldsymbol{\omega}_j$ if $\boldsymbol{c}_j = 0$ (on external geometry $(\mathcal{M} \setminus \mathcal{M}_{\overline{\mathcal{V}}})$, or on passing voxels without scattering) and $\boldsymbol{x}_j = (\boldsymbol{p}'_j, \boldsymbol{\omega}_j)$ if $\boldsymbol{c}_j = 1$ (for aggregate voxel scattering events). Consequently, each path $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, ..., \boldsymbol{x}_{l-1}) \in \mathcal{X}$ comes with a corresponding configuration vector $\bar{\boldsymbol{c}} = (\boldsymbol{c}_0, ..., \boldsymbol{c}_{l-1}) \in \{0, 1\}^l$. A rigorous definition of the path space $\mathcal{X}$ and its corresponding measures can be found in the App. A. It is important to note that despite the angular path parameterisation, the locations $\boldsymbol{p}_j$ of vertices $\boldsymbol{x}_j$ on the manifold $\mathcal{M}'$ are easily recovered, by recursive raytracing against $\mathcal{M}'$ in directions $-\boldsymbol{\omega}_j$, starting from the respective preceding points $\boldsymbol{p}_{j-1}$ and $\boldsymbol{p}'_{j-1}$. Fig. 2 illustrates the recovered vertex locations $\boldsymbol{p}_j$ as well as a complete path from the camera to a light source in our generalised path space.

3.2 Splitting Intra- and Inter-Voxel Transport

Equipped with a generalised voxel path space $\mathcal{X}$, we define the associated voxel-based throughput function $T'(\bar{\boldsymbol{x}})$ and path integral:

$$T'(\bar{\boldsymbol{x}}) = \prod_{j=1}^{l-2} \begin{cases} T_{\mathcal{V}}(\boldsymbol{p}_j, \boldsymbol{\omega}_{j-1}; \boldsymbol{p}'_j, -\boldsymbol{\omega}_j) \left|\boldsymbol{\omega}_j \cdot \mathbf{n}'_j\right| & \text{if } \boldsymbol{c}_j = 1, \\ f(\boldsymbol{\omega}_{j-1}, \boldsymbol{p}_j, -\boldsymbol{\omega}_j) \left|\boldsymbol{\omega}_j \cdot \mathbf{n}_j\right| V_{\mathcal{M}}(\boldsymbol{p}_j, \boldsymbol{p}_{j+1}) & \text{otherwise.} \end{cases}$$
$$I_j = \int_{\mathcal{X}} W(\bar{\boldsymbol{x}}) T'(\bar{\boldsymbol{x}}) L(\bar{\boldsymbol{x}}) d\mu'(\bar{\boldsymbol{x}}). \tag{1}$$

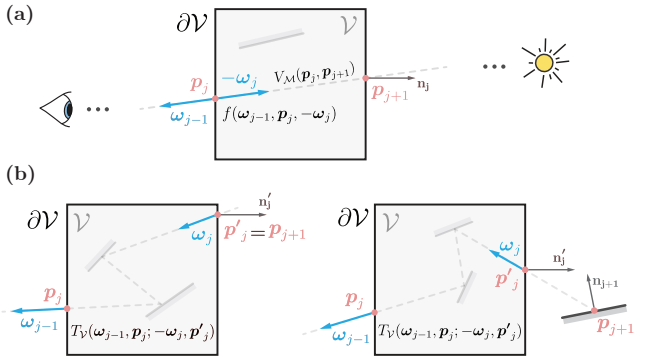

Fig. 3. Two configurations are possible in our generalised voxel path space. **(a)** If $c_j = 0$ the subsequent interaction is entirely defined by a single direction. **(b)** If $c_j = 1$, an additional positional offset $\boldsymbol{p}'_j$ of pre-scattering incidence is tracked. Subsequent vertex positions $\boldsymbol{p}_{j+1}$ fall either onto the next voxel boundary or onto any non-voxelised geometry in $(\mathcal{M} \setminus \mathcal{M}_{\overline{\mathcal{V}}})$.

The throughput $T'(\bar{\boldsymbol{x}})$ is composed of two terms central to our derivation: the *point-to-point visibility function* $V_{\mathcal{M}}$ and the *intra-voxel transport function* $T_{\mathcal{V}}$. Fig. 3 illustrate the two possible configurations in our generalised path space.

The intra-voxel transport for voxel boundary points $\boldsymbol{p}_i$ and $\boldsymbol{p}_o$ with light entering from direction $\boldsymbol{\omega}_i$ at $\boldsymbol{p}_i$ and exiting at $\boldsymbol{p}_o$ into direction $\boldsymbol{\omega}_o$ is measured by:

$$T_{\mathcal{V}}(\boldsymbol{p}_o, \boldsymbol{\omega}_o; \boldsymbol{p}_i, \boldsymbol{\omega}_i) := \int_{\mathcal{P}_{\mathcal{V}}} T_{\boldsymbol{\omega}_i, \boldsymbol{p}_i \to \boldsymbol{p}_o, \boldsymbol{\omega}_o}(\bar{\mathbf{v}}) d\mu(\bar{\mathbf{v}}), \qquad (2)$$

$$\mathcal{P}_{\mathcal{V}} := \bigcup_{n=1}^{\infty} (\mathcal{M} \cap \mathcal{V})^n,$$

where $\bar{\mathbf{v}}$ denotes an intra-voxel path. We add the boundary vertices $\mathbf{v}_0 = \boldsymbol{p}_o$ and $\mathbf{v}_{n+1} = \boldsymbol{p}_i$ on $\partial\mathcal{V}$ to conveniently define the intra-voxel throughput function $T_{\boldsymbol{\omega}_i, \boldsymbol{p}_i \to \boldsymbol{p}_o, \boldsymbol{\omega}_o}(\bar{\mathbf{v}})$, which contains the standard terms of the path-space measurement contribution function inbetween the voxel entry and exit events:

$$T_{\boldsymbol{\omega}_i, \boldsymbol{p}_i \to \boldsymbol{p}_o, \boldsymbol{\omega}_o}(\bar{\mathbf{v}}) = f(\boldsymbol{\omega}_o, \boldsymbol{p}_o, -\boldsymbol{\omega}_0) \prod_{j=1}^{n+1} G(\mathbf{v}_{j-1}, \mathbf{v}_j) f(\boldsymbol{\omega}_{j-1}, \mathbf{v}_j, -\boldsymbol{\omega}_j),$$

where G and f correspond to the geometric term and the reflectance distribution function, respectively. Note that for the voxel boundary points $\mathbf{v}_{n+1} = \boldsymbol{p}_i, \mathbf{v}_0 = \boldsymbol{p}_o \in \partial\mathcal{V}$, the null-scattering distributions are delta functions that result in the marginalisation with respect to both endpoint positions and directions.

We observe in Eq. (1) that our generalised voxel path space leads to a natural decomposition of the full light transport. The point-to-point visibility function fully describes the geometric content of any voxel and preserves internal correlations. This property is especially important in preventing light leaks on long-range free flight paths that are a typical problem of previous LoD geometry aggregation methods. Our *visibility* network, which we introduce in Sec. 4.3, learns an implicit neural representation for $V_{\mathcal{M}}(\boldsymbol{p}_o, \boldsymbol{p}_i)$. We also learn a neural representation of the intra-voxel throughput $T_{\mathcal{V}}(\boldsymbol{p}_o, \boldsymbol{\omega}_o; \boldsymbol{p}_i, \boldsymbol{\omega}_i)$ in a dedicated *appearance* network introduced in Sec. 4.2.

3.3 Far-field light transport simplifications

In practice, learning the fully globally correlation-preserving intra-voxel transport $T_{\mathcal{V}}(\boldsymbol{p}_o, \boldsymbol{\omega}_o; \boldsymbol{p}_i, \boldsymbol{\omega}_i)$ is too complex for large voxels, since it does not reduce the spatial resolution of the learned signal for voxels with increasing sizes across LoDs, while the capacity for voxels remains fixed. This reduces the capacity remaining for learning a high-quality angular reflectance representation. Therefore, we rely on the typical far-field approach of reducing the scattering on voxel boundary points to position-invariant voxel transport functions with only directional dependencies. Note that, while this approach removes positional inter-voxel correlations of multi-bounce indirect scattering paths, our separately learnt intra-voxel visibility function $V_{\mathcal{M}}(\boldsymbol{p}_j, \boldsymbol{p}_{j+1})$ stays fully correlation-preserving. For voxel boundary points $\boldsymbol{p}_i, \boldsymbol{p}_o \in \partial\mathcal{V}$ with incident and exitant directions $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$, the average throughput $\varphi_{\mathcal{V}}(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i)$ of (projected) scattering surfaces $\mathcal{M} \cap \mathcal{V}$ captured by $\mathcal{V}$ is:

$$\varphi_{\mathcal{V}}(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i) := \frac{\iint_{\partial\mathcal{V}} T_{\mathcal{V}}(\boldsymbol{p}_o, \boldsymbol{\omega}_o; \boldsymbol{p}_i, \boldsymbol{\omega}_i) |\boldsymbol{\omega}_o \cdot \mathbf{n}_o| |\boldsymbol{\omega}_i \cdot \mathbf{n}_i| \, dA(\boldsymbol{p}_i, \boldsymbol{p}_o)}{\int_{\partial\mathcal{V}} (1 - V_{\mathcal{M}}(\boldsymbol{p}_o, \mathcal{R}_{\partial\mathcal{V}}(\boldsymbol{p}_o, -\boldsymbol{\omega}_o))) |\boldsymbol{\omega}_o \cdot \mathbf{n}_o| \, dA(\boldsymbol{p}_o)}. \qquad (3)$$

Here, the raytracing operator $\mathcal{R}_{\partial\mathcal{V}}(\boldsymbol{p}_o, \boldsymbol{\omega}_o)$ finds the next boundary point along a ray from $\boldsymbol{p}_o$ in direction $-\boldsymbol{\omega}_o$.

The position-free average intra-voxel throughput then behaves much like any bidirectional scattering distribution function (BSDF), which reduces our practical throughput function back to one case:

$$T(\bar{\boldsymbol{x}}) = \prod_{j=1}^{l-2} \Big(\varphi_{\mathcal{V}_j}(\boldsymbol{\omega}_{j-1}, -\boldsymbol{\omega}_j) \left(1 - V_{\mathcal{M}}(\boldsymbol{p}_j, \boldsymbol{p}_{j+1})\right) + f(\boldsymbol{\omega}_{j-1}, \boldsymbol{p}_j, -\boldsymbol{\omega}_j) \left|\boldsymbol{\omega}_j \cdot \mathbf{n}_j\right| V_{\mathcal{M}}(\boldsymbol{p}_j, \boldsymbol{p}_{j+1}) \Big) 1_{c_j = 0},$$

$$I_j = \int_{\mathcal{X}} W(\bar{\boldsymbol{x}}) T(\bar{\boldsymbol{x}}) L(\bar{\boldsymbol{x}}) d\mu'(\bar{\boldsymbol{x}}). \qquad (4)$$

In conclusion, under a far-field assumption, we only need our appearance network to learn the average throughput $\varphi_{\mathcal{V}}(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i)$. As we will see in Sec. 5.2, generating training samples for the network reduces to generating Monte Carlo estimates of Eq. (3).

4 CORRELATION-AWARE NEURAL PREFILTERING

Building on the theory from the previous section, we now introduce our neural solution to prefilter a scene at various scales.

4.1 Pipeline Overview

Our pipeline comprises multiple stages, including the voxelisation of the scene, and the training of both our appearance and visibility networks, with the latter being fine-tuned through optimised classification thresholds as a preprocess. Finally, the rendering of the prefiltered scene. An overview of our pipeline is illustrated in Fig. 4.

Representation. The first step of our pipeline involves a voxelisation process, where the scene is spatially partitioned into a grid of disjoint voxels. For a given voxel scale, we then store the binary occupancy of each voxel in a sparse grid. We define the smallest level of detail, LoD 0, to correspond to the highest voxel resolution and set it to 512^3. The subsequent LoDs define a geometrically decreasing sequence stopping at the resolution of 8^3 voxels. For all the voxels and the underlying scene geometry, we train two

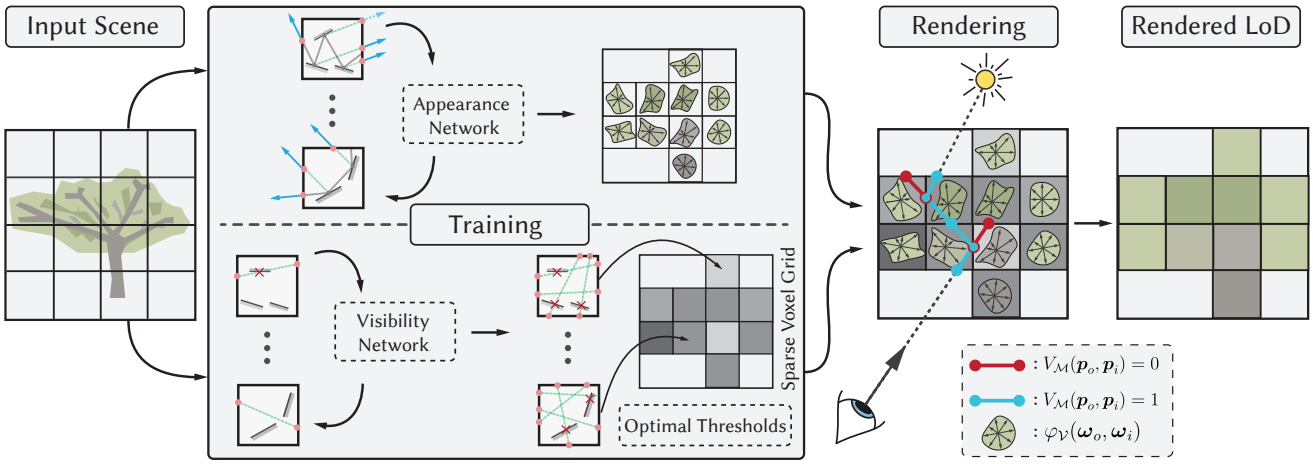

Fig. 4. Our neural asset prefiltering pipeline consists of a preprocessing part running local ray and path sampling on input assets to train our neural LoD representations for visibility and appearance, and a runtime evaluation part integrated with rendering.

networks each for a different component of the voxel-based global light transport defined in Sec. 3. The appearance network learns a filtered representation of the intra-voxel light transport $\varphi_{\mathcal{V}}(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i)$ defined in Eq. (3), while the visibility network is responsible for learning the unbiased visibility function $V_{\mathcal{M}}(\boldsymbol{p}_o, \boldsymbol{p}_i)$. The architecture of each network is described in Sec. 4.2 and Sec. 4.3, respectively. Both networks use a multi-resolution hash grid encoding, which has shown remarkable performance [Müller et al. 2022]. However, it can also introduce undesired artefacts due to potential hash collisions. In Sec. 4.4, we show that these collisions can be identified and addressed by applying a set of optimised thresholds to our visibility classifier.

Training. To benefit from the multi-scale aspect of the method, we train each network on every voxel and level of detail *simultaneously*, i.e., a single network can query the function for any voxel at any scale. In Sec. 5, we describe our strategy for sampling voxels at a particular scale and how the inputs of our networks are sampled to compute reference values.

Rendering. Once our appearance and visibility networks are trained, we estimate Eq. (4) through Monte Carlo integration as follows:

(1) For each camera ray with direction $-\boldsymbol{\omega}_o$, we query the sparse voxel grid for the next non-empty voxel $\mathcal{V}$.
(2) For this non-empty voxel, we obtain the associated entry and exit points $\boldsymbol{p}_o$ and $\boldsymbol{p}_i$ and query the visibility network to evaluate the visibility segment $V_{\mathcal{M}}(\boldsymbol{p}_o, \boldsymbol{p}_i)$.
(3) If the segment is occluded, we sample a direction $\boldsymbol{\omega}_i$, query the appearance network to evaluate $\varphi_{\mathcal{V}}(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i)$ and update the path throughput. Otherwise, we go to step (2) to continue the grid traversal.
(4) We then recursively trace a ray starting from $\boldsymbol{p}_o$ until eventually, the ray either escapes the scene or hits a light source and the ray's contribution is added to the final image.

Note that setting the ray origin to $\boldsymbol{p}_o$ in step (4) directly follows from our theory as it corresponds to the vertex recovered through raytracing against the scene manifold $\mathcal{M}'$ on which our generalised voxel path space is defined.

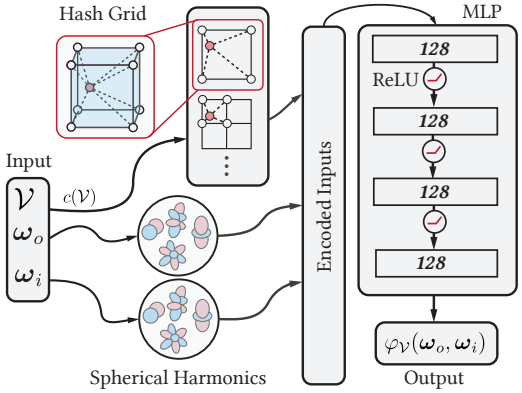

Fig. 5. Our appearance network architecture, where $c(\mathcal{V})$ denotes the centre of the voxel $\mathcal{V}$.

4.2 Appearance Learning

We now describe our network structure and how the average throughput function $\varphi_{\mathcal{V}}(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i)$ is learned for every voxel and scale.

Network Architecture. The input of the appearance network includes the outgoing light direction towards the sensor $\boldsymbol{\omega}_o$, the incident light direction $\boldsymbol{\omega}_i$ and the centre position of its associated voxel $\mathcal{V}$. The latter implicitly encodes the level of detail since voxel centres across different levels form a geometric sequence of 3D positions that never overlap. The output corresponds to the pre-integrated throughput $\varphi_{\mathcal{V}}(\boldsymbol{\omega}_o, \boldsymbol{\omega}_i)$. On the other hand, the network is a dense multilayer perceptron of 4 hidden layers with rectified linear unit (ReLU) activation, each containing 128 neurons without any output activation function. The full architecture is illustrated in Fig. 5.

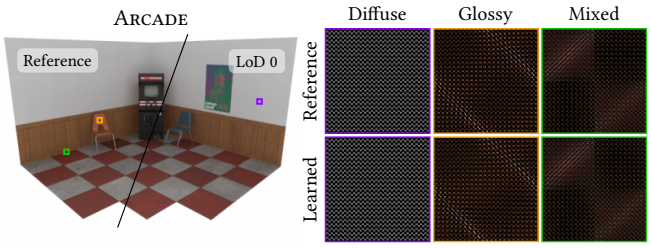

Fig. 6. On the left, we compare the reference to the highest resolution of our neural prefiltering pipeline. The insets visualize the learned throughput functions $\varphi_{\mathcal{V}}(\omega_o, \omega_i)$. We flatten discretised 2D spherical coordinates (θ, ϕ) onto a 1D axis. Different values of ϕ are consecutive, ϕ_o varies horizontally, ϕ_i varies vertically. The first inset shows a voxel on the diffuse grey wall, and the orange inset represents a glossy material on the chair. The last inset shows a voxel where multiple scattering between a diffuse-brown and glossy-red material was aggregated.

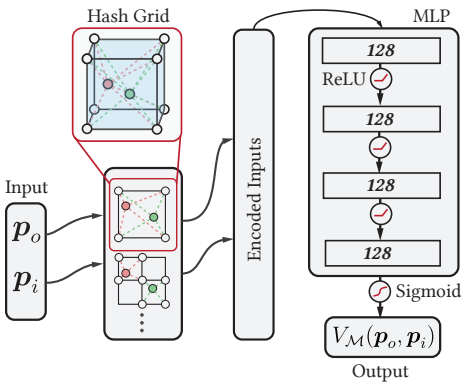

Fig. 7. Our visibility network architecture. Our recurrent composite encoding enables a single hash grid to be used by both inputs.

Input Encodings. The two directions are encoded using spherical harmonics coefficients up to degree 8. Spherical harmonics have the advantage of beings fast to compute and evaluate [Sloan 2008]. On the other hand, the voxel position is encoded using a multiresolution hash grid [Müller et al. 2022]. We set the number of grid levels to match the number of levels of detail, and the base resolution of the hash grid to the lowest resolution processed. We also set the per-level scaling to 2. To control the trade-off between performance and quality, we adjust the number of learned features per level to 2, 4, or 8. In our experiments, we found that using a hash table size $S = 2^{19}$ consistently performs well, although the hash table size can also be used to fine-tune the trade-off.

Loss function. While our goal is to learn an average throughput function that only depends on ω_o and ω_i, generating reference samples for our network still involves the computation of the full intra-voxel throughput. As shown in Sec. 3, this includes an integration over all surfaces contained in a voxel. We can estimate this integral using a path tracer and obtain Monte Carlo samples that can directly be used by our network for training. We construct the estimator for Eq. (3) in Sec. 5.2. We use the noise-to-noise *relative* L_2 loss introduced by Lehtinen et al. [2018], to correctly learn the expected value even when the computed throughput estimates are noisy. As shown in Fig. 6, our network is able to learn a variety of different materials and their combinations appearing in a single voxel.

4.3 Visibility Learning

In the following section, we will delve into the specifics of the visibility network architecture and introduce our novel recurrent composite encoding, which allows us to cut the number of trainable feature parameters in half. In Sec. 4.4, we discuss how we compute optimal thresholds for our binary visibility classifier and show how they significantly improve the stability of the reconstruction.

Network Architecture. Our goal is to learn, for every voxel $\mathcal{V}$, a neural representation of the visibility term $V_{\mathcal{M}}(\boldsymbol{p}_o, \boldsymbol{p}_i)$. Therefore, the network's input only consists of the two vertices on the boundary of the voxel. As we will see shortly, the visibility network does not need to explicitly store the level of detail or the voxel position information. The network structure is similar to that of the appearance network, consisting of 4 fully connected layers with 128 neurons each and ReLU activations. The only difference is that the output layer uses a sigmoid activation function, which constrains the output values between 0 and 1. We interpret the output as the probability that the points $\boldsymbol{p}_o$ and $\boldsymbol{p}_i$ are mutually visible, i.e. the segment $\boldsymbol{p}_o, \boldsymbol{p}_i$ is unoccluded. The visibility network's final architecture is depicted in Fig. 7.

Recurrent Composite Encoding. In order to encode the two positions into a data structure with trainable feature parameters, a natural approach is to use two separate encodings. In most applications, inputs do not have a precise mapping between them. However, in our case, if two separate encodings are learned for each input, it is possible to swap $\boldsymbol{p}_o$ and $\boldsymbol{p}_i$ at inference without any consequence on the inferred visibility. We leverage this simple observation to learn a *single* hash-grid encoding for *both* inputs, effectively halving the footprint of the learned weights. To do so, we implement a novel *recurrent* composite encoding where multiple inputs share a single learnable representation, such as a multiresolution hash grid. This implies that during training, gradients for repeated inputs are back-propagated through the shared representation, and, at inference, the same data structure is queried by the recurrent inputs. Note that this is only possible because of the specific parameterisation we chose for the inputs. The same approach could not be used if we parameterise visibility as a point on the voxel boundary and an associated direction, for example.

Object Space vs. Local Voxel Space Encoding. While the input directions for the appearance network are naturally defined in local voxel space, both local and object space encoding are valid candidates for the visibility network. We experimented with both approaches and found that object space encoding for boundary positions provides better results. This removes any explicit dependency on the level of detail for which visibility is queried, as the incident and exitant locations on the boundary fully encode all relevant information. However, opportunities for exploiting shared visibility information across detail levels arise, as illustrated in Fig. 8.

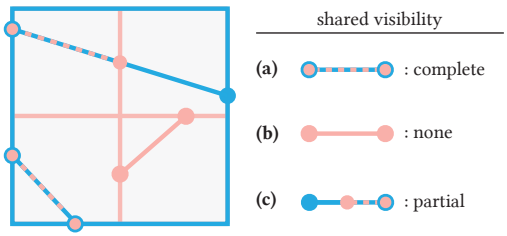

Fig. 8. Three different cases of shared visibility information when boundary points are encoded in object space. If a queried visibility segment between two boundary points is shared between two levels, the exact same information is learned by the network **(a)**. On the other hand, if the segment only exists at the higher resolution level of detail, no information is shared **(b)**. Finally, when a segment spans multiple higher-resolution voxels, different scenarios are possible **(c)**. If a segment in a higher-resolution segment is occluded, the lower-resolution segment is always occluded. And similarly, if the lower-resolution is not occluded, the two higher-resolution segments cannot be occluded either.

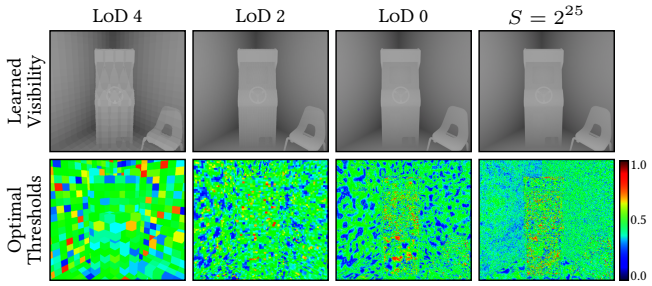

Fig. 9. Our optimal thresholds automatically handle correlated artefacts due to collisions in the hash grid and significantly improve the stability of the visibility reconstruction across detail levels. Visualising the computed thresholds also reveals the correlation *across* LoDs, indicating that the network successfully leverages joint multi-resolution learning. In this example, we used a hash table size of $S = 2^{19}$ and 2 features per grid level. As seen in the last column rendered at LoD 0, increasing S to 2^{25} reduces collision artefacts but does not contribute further to the reconstruction quality.

Loss function. Getting reference data for the visibility network consists of generating two points on the voxel boundary and checking the visibility for a ray traced from $\boldsymbol{p}_o$ to $\boldsymbol{p}_i$. We do this jointly with appearance training as detailed in Sec. 5.2. Since our visibility network is a classifier, we use the binary cross-entropy (BCE) loss which Simard et al. [2003] showed to have faster convergence and better generalisation than a default L_2 loss.

4.4 Optimal Thresholding for Correlation-Aware Visibility

To retain spatial localisation of occlusion events such that it allows correlation-preserving visibility tracking, we train the visibility network as a classifier that predicts binary point-to-point visibility. In order to arrive at a binary prediction, finding an optimal threshold can significantly improve a classifier's performance [Manning et al. 2008]. A *weighted* F-Measure [Harbecke et al. 2022] quantifies the performance of a model given imbalanced training samples. We maximize it *per voxel* to find the optimal threshold given the respective distributions of occlusions and visibility. Once a threshold is fixed, the predictions either belong to the *positive* or the *negative* class [Bishop 2006]. We can compute the rate of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) for a test set, i.e., a set of samples with associated reference values. The precision and recall [Manning et al. 2008] are $\frac{\mathrm{TP}}{\mathrm{TP+FP}}$ and $\frac{\mathrm{TP}}{\mathrm{TP+FN}}$, respectively. The harmonic mean of these metrics leads to the (unweighted) F-Measure:

$$F_\beta = (1+\beta^2) \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{(\beta^2 \cdot \mathrm{precision}) + \mathrm{recall}}, \tag{5}$$

where $0 \leq \beta < 1$ puts more weight on precision (avoiding FP) and $\beta > 1$ puts more weight on recall (avoiding FN). Typical values for β are 1, 0.5 or 2.0. Since this F-Measure depends on which class was labelled as positive, maximising it for imbalanced data sets such as ours can lead to severe bias [Branco et al. 2016]. A common solution is to compute the F-Measure with alternating classification labels and maximise a weighted sum of each result based on the frequency of corresponding positive training samples. Different weighting strategies lead to different trade-offs [Harbecke et al. 2022]. In our application, avoiding *light leaks* in structured opaque geometry is essential. Therefore, we adapt the weighted F-Measure such that the contained F-Measure for positive labelling of occlusion puts more weight on recall:

$$F_{\mathrm{weighted}} = \frac{F_2^{\mathrm{occ}} \cdot n_{\mathrm{occ}} + F_1^{\mathrm{vis}} \cdot n_{\mathrm{vis}}}{n_{\mathrm{occ}} + n_{\mathrm{vis}}}, \tag{6}$$

where the measures of each class are computed as in Eq. (5), and n_{occ} and n_{vis} represent the number of occluded and visible segments in the sampled references, respectively. In practice, a grid search over 100 equally-spaced thresholds in the interval $[0, 1]$ with 1000 visibility samples per voxel proved sufficient to find optimal thresholds per voxel. The combined optimisation time for all LoDs lies between 20–120 seconds depending on the scene's sparsity. Fig. 9 illustrates the learned visibility for different levels of detail using the computed thresholds. The optimal thresholds significantly improve the reconstruction quality. In particular, it effectively resolves correlated compression artefacts, which we attribute to visible hash collisions in the hash grid encoding. Otherwise, increasing the hash table size reduces these occurrences at the cost of a higher memory footprint. In either case, Fig. 10 shows that light leaks are effectively reduced by the threshold optimization.

4.5 Efficient Ray Traversal

In order to make the rendering process described in Sec. 4 practical, we apply two essential optimisations in our pipeline.

Correlation-preserving Stochastic Traversal. Compared to a stochastic approach that generates collisions with a certain probability at every traversed voxel boundary, our correlation-preserving binary thresholding naturally requires more traversal steps for long unoccluded light paths. Reintroducing a small stochastic component in an occlusion-preserving way can drastically improve render time. To achieve this, we apply a variation of the traditional Russian Roulette method based on the continuous visibilities inferred during grid traversal: We use the inferred probability of non-collision in

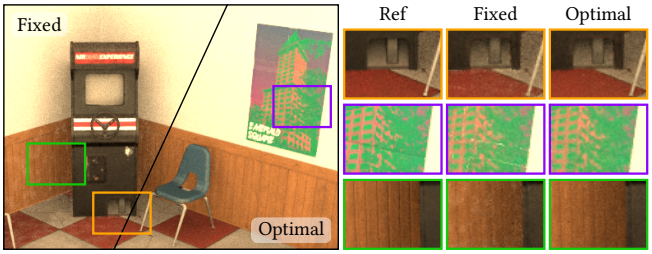

Fig. 10. We demonstrate the difference between rendering with a global fixed visibility threshold at 0.5 and applying optimal thresholds per voxel. Rendering LoD 0 at a higher pixel resolution with a strong constant [6,4,2] RGB illumination reveals multiple correlated light leaks for the fixed threshold variant, while our computed thresholds successfully address the problem.

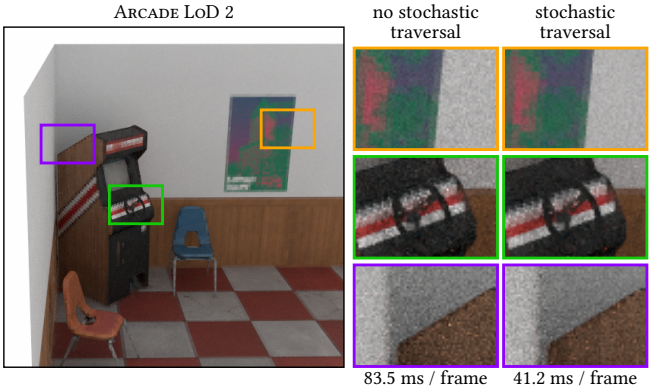

Fig. 11. Our stochastic traversal trades variance for efficiency. We compare to our approach without stochastic traversal at LoD 2. Each method is rendered for 10 seconds at a higher pixel resolution to better show the noise difference. Our approach consistently achieves lower variance in the same time budget.

each voxel as a lower bound for the survival probability of a ray traversing that voxel. When the network infers a low probability of occlusion for a queried segment, the survival probability is high and the path should never be terminated in that voxel. To limit the amount of variance introduced by this approach, we use very low survival probabilities overall. We compute an upper bound, such that, at most 0.001% of the rays traversing all voxels along the diagonal are terminated at the highest grid resolution. This results in a survival probably around 99.25% for a 512^3 resolution, and leads to a good balance between variance and speed increase. Fig. 11 illustrates the benefit of the described stochastic traversal strategy.

Compaction. In our framework, reaching the next scattering event requires the traversal of multiple non-empty voxels, and several visibility inferences might be required until a queried segment is occluded. In a wavefront approach, this is problematic as a single ray can become a bottleneck. Therefore, to balance the workload on the GPU, we apply compaction after each batch of visibility inferences. We also apply compaction after each scattering event, as done in a typical wavefront path tracer. Interested readers can find more information on parallel stream compaction in Billeter et al. [2009].

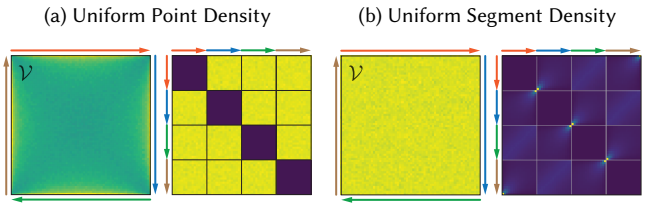

Fig. 12. We compare the resulting densities when sampling two points uniformly on the boundary (while avoiding pairs that land on the same side) **(a)** and our approach **(b)**. In each subfigure, the left inset shows the resulting density of segments inside the voxel while the right inset represents the density of points $\boldsymbol{p}_o$ and $\boldsymbol{p}_i$ on the boundary.

5 TRAINING THE REPRESENTATIONS

In this section, we describe our strategy for selecting a voxel at a particular level of detail and how we sample the appearance and visibility network inputs to generate the required training data.

5.1 Training Distribution

Our training data is generated on the fly by sampling the full domain for all levels of detail and generating MC estimates for all training inputs. To avoid the expensive generation of converged ground truth reference data, we rely on noise-to-noise training [Lehtinen et al. 2018]. Sampling the training inputs is comprised of two steps, a discrete sampling of LoD and voxel index; and a continuous sampling of the selected voxel domain.

Voxel and LoD Sampling. In order to select a voxel for learning, we sample a sparse voxel from the highest resolution and a level of detail uniformly. If the LoD does not correspond to the highest resolution, we determine the corresponding voxel at the lower resolution instead. This approach automatically allocates more samples to lower-resolution voxels that contain more active voxels from the higher resolution. We found this sampling method to improve convergence compared to uniformly sampling a voxel across all levels of detail. We also experimented with approaches that distribute samples non-uniformly across levels of detail, but performance significantly varies among different scenes, and we found the above-described approach to generalise better.

Sampling the Integration Domain. Learning either the visibility or the appearance of a selected voxel requires us to sample vertices on the voxel boundary. The only requirement when generating reference samples for a network is to cover the input domain entirely. However, their density determines their weight in the minimisation of the loss function. A naive approach might sample the two points uniformly on the boundary while discarding samples with two points on the same face. Fig. 12a shows that this does not, however, lead to a uniform sampling of the geometry within the voxel. Therefore, we optimise the density of the segments formed between $\boldsymbol{p}_o$ and $\boldsymbol{p}_i$. As shown in Fig. 12b, this density becomes uniform when sampling voxel boundaries according to their projected areas as seen from uniformly sampled directions $\boldsymbol{\omega}_o$ on the unit sphere.

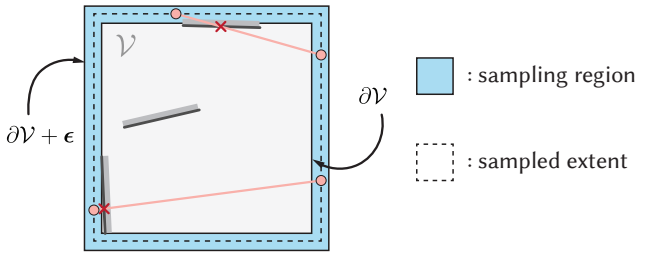

Fig. 13. Our sampling domain regularisation addresses floating point errors that can arise when surfaces are close to the original voxel boundary.

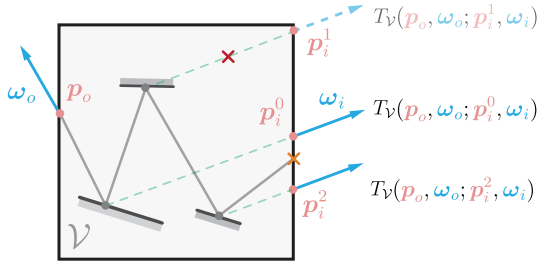

Fig. 14. To obtain reference samples for the appearance network, we can, in a single pass, aggregate multiple throughput samples for different exitant position $\boldsymbol{p}_i$. A red cross indicates a failed shadow test, while an orange cross indicates path termination at the voxel boundary.

To sample the projected area for each voxel face, given a random direction $\boldsymbol{\omega}_o$, we compute the dot product of each face normal $\mathbf{n}$ with $\boldsymbol{\omega}_o$ and discard faces with negative results. We then sample from the remaining faces proportionally to their dot product. Finally, we sample a point $\boldsymbol{p}_o$ on the face uniformly at random. The opposite point of incidence $\boldsymbol{p}_i$ is given by the intersection between the voxel boundary and a ray originating at $\boldsymbol{p}_o$ traced in direction $-\boldsymbol{\omega}_o$.

Learning Domain Regularisation. For certain assets, the original geometry might land precisely on the voxel boundary. In that case, due to floating point precision issues, the networks might learn incorrect visibility. To avoid this problem without introducing bias, we learn a slightly larger domain by extruding the voxel boundary outwards, see Fig. 13. For the sampling procedures described earlier, this only amounts to an additional uniform sampling of the outward voxel extent range. In practice, we found that extending the domain by 0.5% of the original voxel extent fixes the issue. Note that during inference, we still query the original voxel boundary, only the learned domain is extended.

5.2 Constructing the Appearance Estimator

We use noise-to-noise training [Lehtinen et al. 2018] with unbiased MC estimates to train the average voxel throughput function $\varphi_\mathcal{V}(\boldsymbol{\omega}_o,\boldsymbol{\omega}_i)$ in Eq. (3), which involves path integration. We sample trained incident and exitant directions $\boldsymbol{\omega}_o$ and $\boldsymbol{\omega}_i$ uniformly for each voxel. The visible geometric area of geometry contained by a voxel $\mathcal{V}$ as given in the denominator of Eq. (3) is sampled by rejection sampling: As described in the previous section, we uniformly generate a point $\boldsymbol{p}_o$ on the projected voxel boundary as seen along rays with direction $\boldsymbol{\omega}_o$ with a proposal distribution $p_{\boldsymbol{\omega}_o}(\boldsymbol{p}_o) \propto |\boldsymbol{\omega}_o \cdot \mathbf{n}_o|$. Ray tracing against the geometry within the voxel tests, if the sampled point lies within the visible geometric area and should be accepted. Note that all sampled $\boldsymbol{p}_o$, accepted and rejected, can still be used for visibility training, as the rejection criterion directly determines the reference visibility for any sampled positions. Point samples without ray intersections are rejected for appearance training, resulting in the training distribution $p^\Phi_{\boldsymbol{\omega}_o}(\boldsymbol{p}_o)$ for accepted samples:

$$p^\Phi_{\boldsymbol{\omega}_o}(\boldsymbol{p}_o) := \frac{p_{\boldsymbol{\omega}_o}(\boldsymbol{p}_o)}{\int_{\partial\mathcal{V}} \left(1 - V_\mathcal{M}(\boldsymbol{p}_o, \mathcal{R}_{\partial\mathcal{V}}(\boldsymbol{p}_o, -\boldsymbol{\omega}_o))\right) p_{\boldsymbol{\omega}_o}(\boldsymbol{p}_o) dA(\boldsymbol{p}_o)}.$$

Thus, we can sample the point of exitance $\boldsymbol{p}_o$ and replace the outer integral in Eq. (3) by an MC estimation using $p^\Phi_{\boldsymbol{\omega}_o}(\boldsymbol{p}_o)$ to arrive at a simplified equation:

$$\varphi_\mathcal{V}(\boldsymbol{\omega}_o,\boldsymbol{\omega}_i) = \mathbb{E}\left[\int_{\partial\mathcal{V}} T_\mathcal{V}(\boldsymbol{p}_o,\boldsymbol{\omega}_o;\boldsymbol{p}_i,\boldsymbol{\omega}_i)\,|\boldsymbol{\omega}_i \cdot \mathbf{n}_i|\,dA(\boldsymbol{p}_i)\right].$$

Note that the point of incidence $\boldsymbol{p}_i$ is fully determined by the light interactions within the voxel and the fixed incident direction $\boldsymbol{\omega}_i$ due to the null-scattering distribution $f(\boldsymbol{\omega}_n,\boldsymbol{p}_i,\boldsymbol{\omega}_i) = \delta(\boldsymbol{\omega}_n - \boldsymbol{\omega}_i)/|\mathbf{n}_i \cdot \boldsymbol{\omega}_n|$ on the voxel boundary contained in the throughput $T_\mathcal{V}(\boldsymbol{p}_o,\boldsymbol{\omega}_o;\boldsymbol{p}_i,\boldsymbol{\omega}_i)$. The first interaction in the path integral of $T_\mathcal{V}(\boldsymbol{p}_o,\boldsymbol{\omega}_o;\boldsymbol{p}_i,\boldsymbol{\omega}_i)$ is equally fixed by the constraint of the sampled $\boldsymbol{p}_o$ and $\boldsymbol{\omega}_o$ as represented by the null-scattering interaction on the exitant path vertex. To estimate the remaining nested integrals, we then apply standard path tracing from the exitant direction and sample inner directions of incidence with corresponding BSDF importance sampling densities $p(\boldsymbol{\omega}_j|\boldsymbol{\omega}_{j-1},\mathbf{v}_j)$. We transform all directional distributions to the area measure, cancelling corresponding Jacobian determinants with respective parts of the geometry terms, in order to arrive at the final MC estimation:

$$\varphi_\mathcal{V}(\boldsymbol{\omega}_o,\boldsymbol{\omega}_i) = \mathbb{E}\left[\sum_{n=1}^{\infty} \frac{\prod_{j=1}^{n} f(\boldsymbol{\omega}_{j-1},\mathbf{v}_j,\boldsymbol{\omega}_j)\,|\boldsymbol{\omega}_j \cdot \mathbf{n}_j|}{\prod_{j=1}^{n-1} p(\boldsymbol{\omega}_j|\boldsymbol{\omega}_{j-1},\mathbf{v}_j)} V_\mathcal{M}(\mathbf{v}_n,\boldsymbol{p}_i)\right].$$

Note that in practice the sum does not have an infinite number of terms, since paths will cross the voxel boundary after a number of interactions, reducing all subsequent terms to zero. As illustrated in Fig. 14, the derived estimator is realised by recursive ray tracing, starting from a sampled point $\boldsymbol{p}_o$ in direction $\boldsymbol{\omega}_o$, accumulating the throughput for any light incident from direction $\boldsymbol{\omega}_i$ if it is unshadowed by other geometry within the voxel, and importance sampling a new direction for ray tracing proportional to the scattering distribution at the hit point. This procedure is repeated until a sampled ray crosses the boundary of the currently trained voxel. Shadow testing is done by tracing an additional shadow ray from each encountered interaction within the voxel to the voxel boundary in direction $\boldsymbol{\omega}_i$ to detect any other blocker geometry contained therein.

6 RESULTS

We present results of applying our neural prefiltering pipeline to scenes of different structure and complexity. We analyze the memory footprint and performance of our method, as well as the perceptual

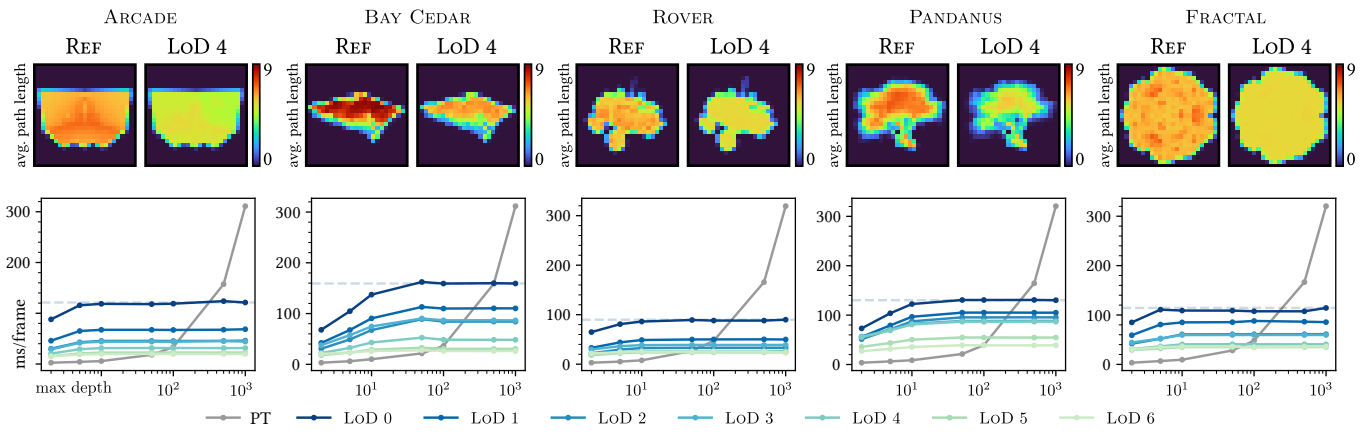

Fig. 15. Top, the average path length at LoD 4 compared against a brute-force path tracing, demonstrating the effectiveness of our approach to compress the light transport. Bottom row compares performance across levels of detail and path traced references for fixed 1024^2 images. The plots report mean render time for 128 samples/pixel and different maximum path lengths with the dashed line indicating the time for the highest LoD at a max depth of 10^3. We achieve interactive to real-time performance across resolutions.

ꟻLIP error [Andersson et al. 2020], which approximates the difference perceived by humans when alternating between one rendered image and one corresponding ground truth. In addition to the results presented here, we provide in our supplemental material a more detailed set of comparisons and a video demonstrating the temporal stability of our method. We implemented our method in a GPU wavefront path tracer, using *nanovdb* [Museth 2021] to represent a sparse grid storing one visibility threshold per voxel, and the *tiny-cuda-nn* library [Müller 2021] with scalars set to half-precision floats for optimised network training and inference, to which we added the binary cross entropy loss and our recurrent grid encoding. We use a single NVIDIA RTX 3080 for all our experiments.

Training Times. To train our representations using the losses described in Sec. 4.2 and 4.3, we use the Adam optimiser [Kingma and Ba 2014] with a learning rate of $\eta = 0.005$ for both networks with their respective grid encodings. Computation time for a single optimisation step varies depending on scene complexity, where most effort is spent tracing rays and transport paths within voxels. For path tracing efficiency in appearance training, we apply standard albedo-based Russian Roulette to bound the theoretically infinite number of scatter events in an unbiased way.

Table 1 reports training times for all our test scenes as displayed in Fig. 16. Visibility training is generally slower to converge due to using a higher number of encoding parameters. Both visibility and appearance representations can be trained in the order of minutes, moving preprocessing to neural representations closer to practicality for real-world asset pipelines, compared to previous work that required hours to days of training [Bako et al. 2023; Vicini et al. 2021]. Note that our appearance decomposition into two representations allows partial re-train when only materials are changed. For fast, lower-quality previews reduced parameter counts may be used.

LoD Efficiency. By design, our method achieves high compression ratios when applied to large quantities of geometric primitives or to materials with high-resolution microdetail. We measure storage savings between 80% and 90% for assets with high-frequency geometric details, as listed in Table 2. However, compressed storage is not our primary objective. While in principle, all sufficiently tessellated scenes could be compressed this way, efficient representation by our method mainly targets high-frequency appearance involving many-bounce indirect scattering (such as, e.g. light scattering between leaves). Fig. 15 demonstrates that our method effectively collapses many-bounce shading evaluations to a lower number of voxel appearance evaluations representing the full, aggregated light transport. We thus reach higher uniformity of path lengths across pixels, with lower remaining differences, where long-range correlations cannot yet be preresolved at the respective LoD scales.

To evaluate the runtime efficiency, we rendered all levels of detail as displayed in Fig. 15 at a fixed resolution and plotted the average render times as a function of maximum path length. While we cannot claim perfectly exponential scaling with reduced detail levels in our current prototype implementation, we already do see speedups up to 5× between the highest and the lowest levels of detail. Within our prototype, we cannot yet compete with raw real-time path tracing performance.

Table 1. Mean and total training time for all our scenes after 100 steps. Depending on the scene complexity, 8000–12000 steps are needed for convergence. In the Fractal scene, in contrast to others, paths sampled during training are stuck within the geometry for longer, resulting in prolonged training times for the appearance network.

Scene	Visibility Net. 100 steps / total	Appearance Net. 100 steps / total	Total
Arcade	2.52s / 5.04m	1.08s/ 2.16m	**7m12s**
Bay Cedar	5.48s / 10.96m	4.93s/ 9.86m	**20m49s**
Rover	3.12s / 6.24m	1.66s/ 3.32m	**9m33s**
Pandanus	6.13s / 12.26 m	4.41s/ 8.82m	**21m4s**
Fractal	8.11s / 16.22m	12.03s / 24.06m	**40m16s**

ACM Trans. Graph., Vol. 42, No. 4, Article . Publication date: August 2023.

Table 2. Memory footprint of each network and the total size of the sparse voxel grid associated with each LoD. The original scene size is measured as binary file size containing both textures and geometry.

Scene	Visibility Network ($S = 2^{19}$) Feat./Lvl	Visibility Network ($S = 2^{19}$) Train. Params	Appearance Network ($S = 2^{19}$) Feat./Lvl	Appearance Network ($S = 2^{19}$) Train. Params	Voxel Grid Sparsity (LoD 0 / 6)	Voxel Grid Total Size	Total Memory Footprint LoDs	Total Memory Footprint Original Scene	Total Memory Footprint Space Saving
Arcade	2	3.81M (7.63 MB)	2	3.80M (7.60 MB)	99.43% / 71.48%	4.08 MB	19.31 MB	68.28 MB	**71.72%**
Bay Cedar	8	15.04M (30.09 MB)	4	7.56M (15.12 MB)	98.71% / 78.71%	8.93 MB	54.14 MB	752.92 MB	**92.81%**
Rover	4	7.54M (15.10 MB)	4	7.56M (15.12 MB)	98.62% / 72.07%	9.45 MB	39.67 MB	187.41 MB	**78.83%**
Pandanus	8	15.04M (30.09 MB)	4	7.56M (15.12 MB)	98.91% / 54.69%	8.60 MB	53.81 MB	1.40 GB	**96.16%**
Fractal	8	15.04M (30.09 MB)	4	7.56M (15.12 MB)	87.74% / 29.69%	86.23 MB	131.44 MB	1.92 GB	**93.15%**

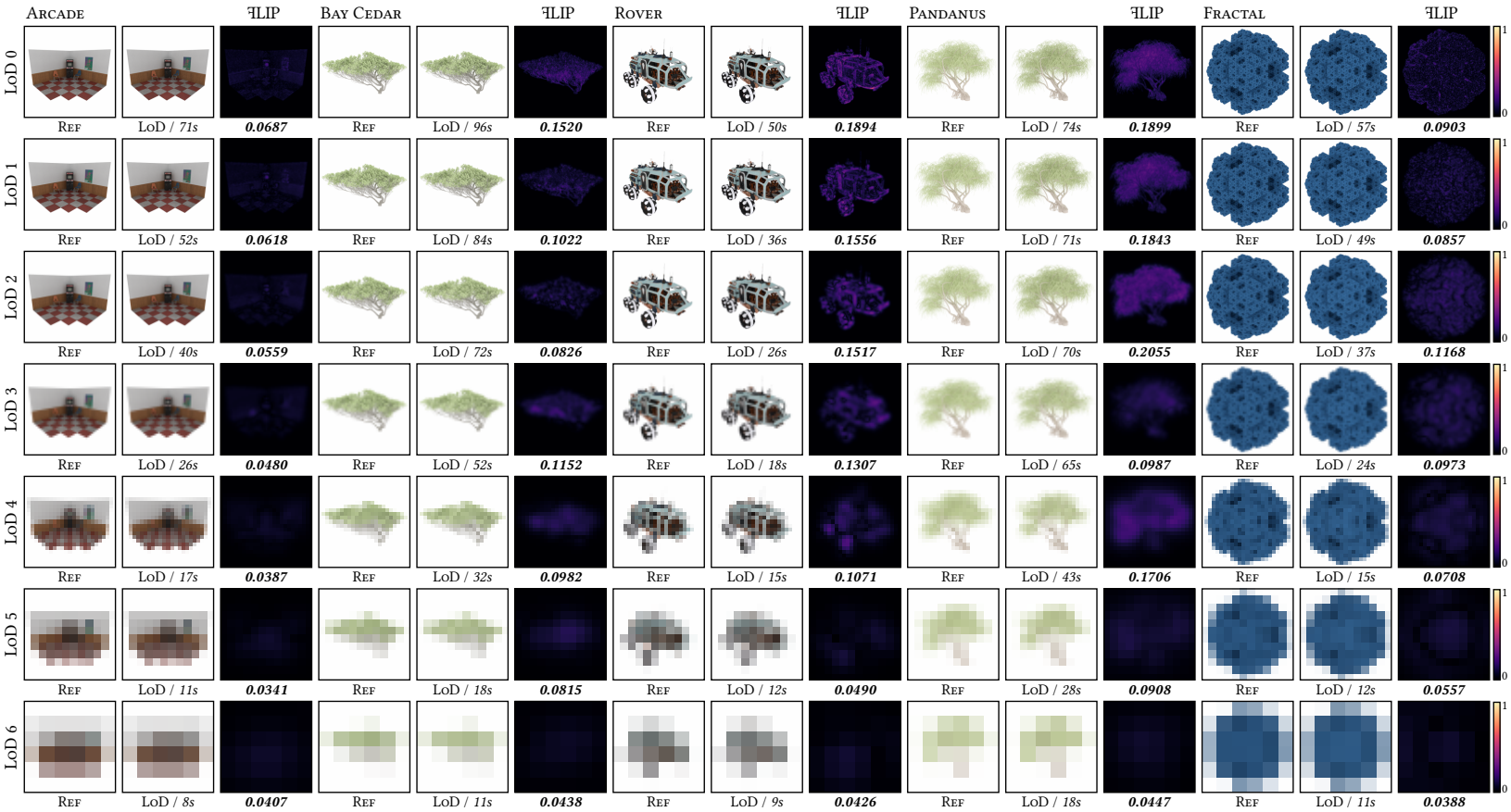

Fig. 16. Comparison of our method for a set of scenes with varying materials and geometric correlation. To compare to a standard path tracer at the same resolution, each scene is rendered at a different level of detail such that a voxel is roughly pixel-sized. The mean value of ꟻLIP error excludes empty background pixels. Reported timings correspond to fixed 512^2 images rendered at 1024 samples per pixel to ensure converged results.

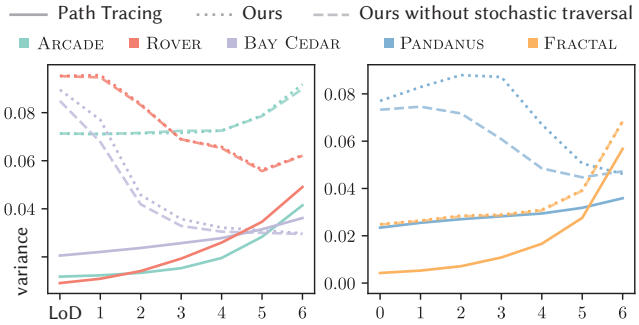

Fig. 17. Our approach does not apply any importance sampling and therefore has high per-pixel variance at higher detail levels. However, variance of our method is reduced as detail is reduced, while path tracing complexity stays constant due to tracing the same light paths. Our stochastic traversal incurs a minor addition of variance.

We expect this balance to change in the future by means of both more acceleration for neural representations and better sampling techniques. A preliminary analysis of variance reduction capabilities is provided later in this section.

Reconstruction Quality. We evaluate the quality of our approach on scenes of various complexity and discuss the individual challenges represented by each scene as displayed in Fig. 16. The memory footprint of each scene is shown in Table 2. All the scenes compared are rendered for an unlimited path depth, i.e., paths are only killed via Russian Roulette. This is the only viable comparison against a reference since our prefiltering technique compresses the light transport and the number of bounces in our generalised voxel path space does not correspond to the usual path depth of a path tracer.

The Arcade in Fig. 16 has relatively simple geometry, but features large flat surface areas with multiple high-resolution textures including varying roughness parameters. Our method reconstructs the geometry with high accuracy, without any special handling for large-scale surface-like geometry. The poster on the wall is challenging to prefilter because of the rapidly changing reflectance across different voxels. Due to the high sparsity of the scene (99.43 % of the voxels at the highest resolution are empty), we can still obtain good results across all scales with relatively small networks (see Table 2).

The Bay Cedar is a prominent example of a mixture of structured and unstructured geometry. The trunk, the middle branches, and the upper leaves all result in different kinds of correlation. Our method accurately captures this continuum, even for thin branches with low fractional voxel coverage at low resolutions. At high resolutions, these features are naturally captured by the sparse encoding.

The Rover features a more complex mix of surface-like structures and exhibits a variety of multi-scale details, such as a thin antenna on top of the car and many small details with a mix of glossy and diffuse materials. This makes it a particularly challenging scene for the appearance network, particularly visible at LoD 0. As the resolution decreases, high-frequency components in the filtered appearance diminish, and we observe reduced reconstruction errors.

The Pandanus the most challenging scenes for our method. The leaves correspond to a multitude of thin hair-like structures that lead to high-frequency appearance changes across voxels. In some of the detail levels, we see difficulty in robustly learning visibility and appearance such that the overall transported energy is fully conserved. Accuracy fluctuates across multiple training reruns, and we observe these difficulties to affect different levels of detail every time, while the error for other levels is then well reduced. This hints at an interesting problem for follow-up investigations on optimal training strategies that balance energy loss across all levels of detail.

The Fractal contains numerous intricate details that result in an extremely dense geometry, as can be seen by the reported sparsity for LoD 0 and LoD 6 in Table 2. Our method can nonetheless achieve high-quality reconstructions for all levels of detail. Note that while the scene uses a single diffuse material, all the shading is due to multiple scattering, masked by high-frequency occlusions that need to be learnt by the appearance network at lower resolutions.

Variance Analysis. Compared to unfiltered path tracing where variance increases as pixels cover an increasing amount of details aggregating as viewing distance increases, the prefiltered appearance representation in our method reduces variance as expected from such approaches. Fig. 17 compares variance of detail levels for multiple scenes, rendered at resolutions matching voxels to pixel footprints. Our prefiltering technique manages to reduce the variance even for assets with complex visibility.

Comparisons to Previous Work. We first compare our neural prefiltering approach to the state-of-the-art Hybrid LoD technique from Loubet et. al [2017] for the Oak Tree and the Bare Tree in Fig. 18. Our method noticeably reduces the error compared to their method, particularly as more details are aggregated and thus transition from correlated to uncorrelated geometry. At the highest resolution, their Oak Tree reconstruction still closely matches the reference, except for slight edge blur. However, as the resolution decreases, their blurring becomes a clear defect in the ꟻLIP error, with noticeable undesired brightening. Even for the leaves where the assumption of uncorrelated geometry can be adequate, we notice a loss of accuracy in the high-frequency details that should remain at lower resolutions. For the Bare Tree, which is composed only of a trunk and branches with a simple surface-like structure, their segmentation process turns to volume approximations too quickly. Thus the branches suffer typical overblurring of volumetric representations. Our approach retains low error across scales and geometric configurations in comparison. In terms of memory footprint, our neural approach improves compression rates by 27% over their method. Hybrid LoD stores the Oak Tree with 3.72MB of combined textures and mesh and an additional 79.22MB of sparse volumetric data, totalling **82.94**MB. In comparison, our approach stores 22.64MB for the voxel grid and 37.72MB for the two networks, totalling **60.36**MB.

We also compare our approach to the deep appearance prefiltering method introduced by Bako et al. [2023] in Fig. 19. Their current implementation supports direct lighting only which we emulate by limiting the path depth to two during appearance training. They also require an expensive 0.5-2 days of training on a cluster while our entire pipeline can be trained in under one hour on a single GPU at better reconstruction quality overall.

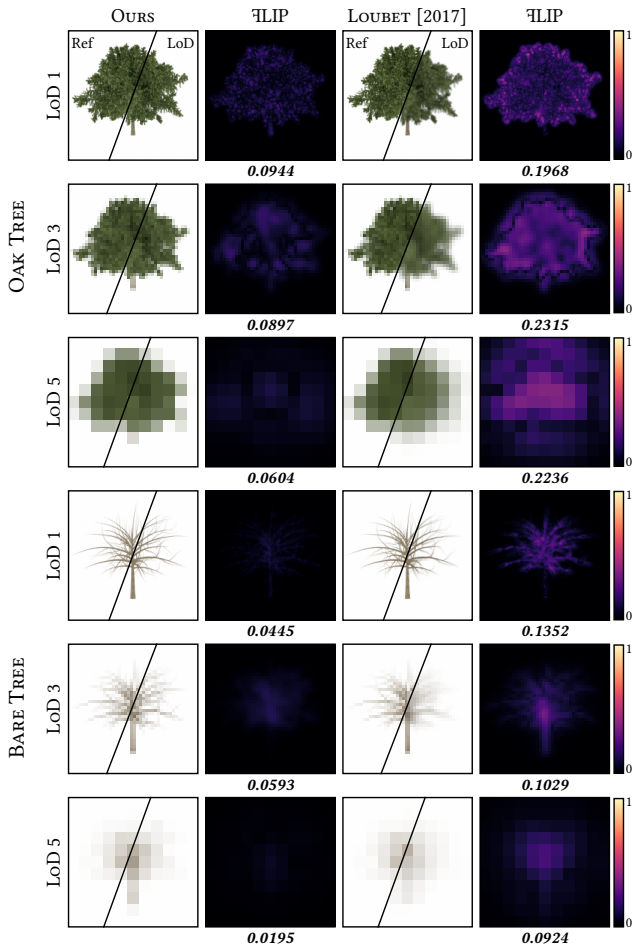

Fig. 18. Our neural prefiltering pipeline compared to the state-of-the-art Hybrid LoD technique for a low, medium, and high-resolution level of detail for two different scenes. For the Oak Tree, our visibility and appearance networks were trained using 8 and 4 features per level, respectively. For the second asset, both networks only use 4 features per level.

Correlated Visibility. In Sec. 4.4, we proposed a method to optimise per-voxel thresholds for increased accuracy in visibility. We compare our approach of optimal thresholding for binary correlation-aware visibility tracking with simple stochastic visibility, as shown for two different hash table sizes in Fig. 20. In the stochastic alternative, the output of the visibility network is directly interpreted as a probability. For volume-like details, it can sometimes outperform binary classification, while eliminating the need for any thresholding. Note that, even if the point-to-point visibility becomes stochastic some learned correlation is still preserved. The stochastic approach is effective if the hash table size can be sufficiently increased and the scene has strong volume-like properties. For general scenes, however, such a simplified stochastic approach is susceptible to light leaks, and we find the fully correlation-preserving thresholds to be more robust overall.

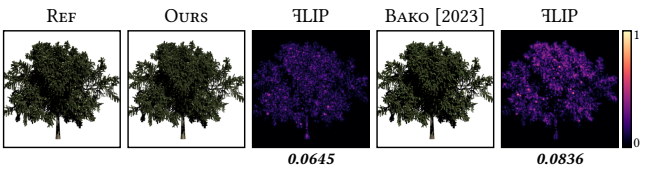

Fig. 19. Our method evaluated against Bako et al. [2023] at LoD 1. Due to the glossy tree material, the scene illuminated by a directional light source reveals many challenging specular highlights. Our two networks use eight features per level for the hash grid to account for this.

Limitations and Future Work. We showed that in some remaining challenging cases, further investigation into optimised appearance learning strategies that deterministically control error across all levels of detail is needed, particularly with a priority on robust energy conservation where accuracy has to be traded for compression. Regarding general capacity for complex appearance, the use of spherical harmonics-based directional encodings, while efficient, may not fully capture complex high-frequency appearances. Where filtered appearance has to retain sharp glossy highlights, directional adaptations of all-frequency (potentially parametric) encodings are likely needed (e.g. an optimised hash grid, tri-planar, or octahedral maps). Furthermore, tailored importance sampling for robust training of such phenomena will then be required to capture all relevant angular configurations reliably. Automatically adapting training distributions in the spirit of active exploration [Diolatzis et al. 2022] could be a promising avenue for future work. As discussed in our variance analysis, developing effective importance sampling strategies for sampling complex aggregate distributions at runtime is an important next step to further improve the practical efficiency of neural representations in rendering applications. Effective compression of dynamic content is actively researched for neural representations [Park et al. 2021; Pumarola et al. 2021; Wang et al. 2022b], with interesting recent research avenues [Song et al. 2023] for the type of parametric encodings we use. As to performance, optimising the traversal logic in a way that fuses visibility inferences and voxel stepping in one kernel is a good opportunity as the software ecosystem of neural components in graphics matures.

7 CONCLUSION

As photorealistic rendering approaches real-time efficiency, particularly on GPUs with limited storage capacity, level of detail techniques preserving complex appearance are more relevant than ever. We demonstrated that neural representations for local appearance are promising building blocks, achieving more uniform rendering convergence and high compression rates, while attaining high accuracy. Our method is designed to integrate well with existing physically-based renderers. It efficiently decomposes the problem into visibility and appearance representations. We showed the merits of representing visibility with classifying neural fields, handling visibility correlations from opaque to aggregate surfaces. We also demonstrated the effectiveness of increasing their efficiency by use of optimal classifier theory, allowing direct control over light leaks. For high-dimensional appearance, we showed the effectiveness of

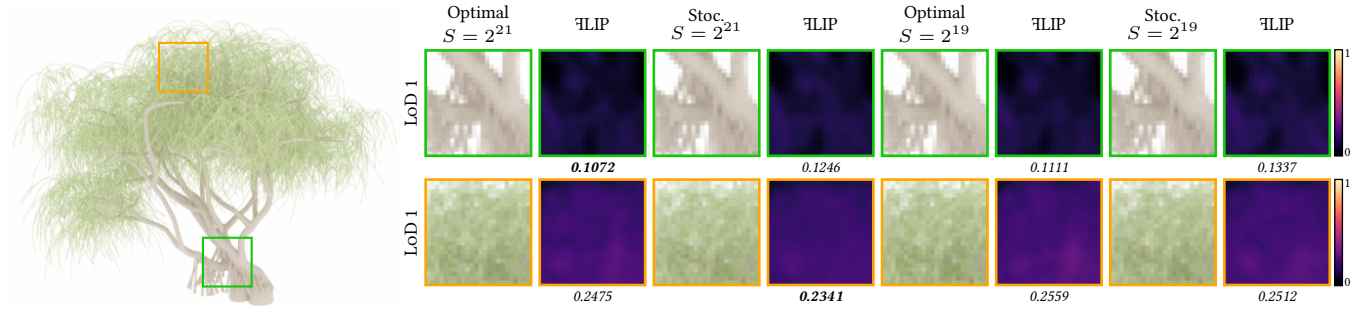

Fig. 20. Our optimal thresholds compared to an alternative approach of stochastic thresholding. Increasing the hash table size of the visibility network to $S = 2^{21}$ improves the stochastic thresholding for volumetric features at the cost of increased error for surface-like features and potential light leaks.

implicit neural representations for storing the corresponding high-dimensional neural fields. With our method, we cover the essential aspects of prefiltering with improvements over previous work for a variety of assets. Further development of robust multi-scale learning strategies and effective importance sampling strategies for complex aggregate distributions are noteworthy examples of many new research directions for neural representations in graphics.

ACKNOWLEDGMENTS

We thank Steve Bako for giving early access to their Deep Appearance Prefiltering technique, Johannes Meng and Sebastian Herholz for proofreading, Christoph Peters for his early mathematical insights and the anonymous reviewers for their helpful remarks and suggestions to improve the paper. We also thank the artists for generously providing the assets used in our test scenes: PolyHaven (environment maps), Walt Disney Animation Studio (Bay Cedar and Pandanus from the Moana Island Scene) and vajrablue (Rover). This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 956585 (https://prime-itn.eu).

REFERENCES

Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. 2020. ꟻLIP: A Difference Evaluator for Alternating Images. *ACM Comp. Graph. and Interactive Techn.* 3, 2 (2020), 15:1–15:23.

Hendrik Baatz, Jonathan Granskog, Marios Papas, Fabrice Rousselle, and Jan Novák. 2021. NeRF-Tex: Neural Reflectance Field Textures. In *Eurographics Symp. Rend.*

Steve Bako, Pradeep Sen, and Anton Kaplanyan. 2023. Deep Appearance Prefiltering. *ACM Trans. Graph.* 42, 2, Article 23 (2023).

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. 2021. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. *ICCV* (2021).

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. *CVPR* (2022).

Markus Billeter, Ola Olsson, and Ulf Assarsson. 2009. Efficient Stream Compaction on Wide SIMD Many-Core Architectures. In *Proceedings of the Conference on High Performance Graphics 2009* (New Orleans, Louisiana) *(HPG '09)*. Association for Computing Machinery, New York, NY, USA, 159–166. https://doi.org/10.1145/1572769.1572795

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag.

Adrian Blumer, Jan Novák, Ralf Habel, Derek Nowrouzezahrai, and Wojciech Jarosz. 2016. Reduced Aggregate Scattering Operators for Path Tracing. *Comp. Graph. Forum* 35, 7 (2016), 461–473.

Paula Branco, Luís Torgo, and Rita P. Ribeiro. 2016. A Survey of Predictive Modeling on Imbalanced Domains. *ACM Comput. Surv.* 49, 2, Article 31 (2016).

Eric Bruneton and Fabrice Neyret. 2011. A survey of nonlinear prefiltering methods for efficient and accurate surface shading. *Trans. Vis. and Comp. Graph.* 18, 2 (2011), 242–260.

Eric Bruneton and Fabrice Neyret. 2012. Real-time Realistic Rendering and Lighting of Forests. *Comp. Graph. Forum* 31, 2pt1 (2012), 373–382.

Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. 2021. Efficient Geometry-aware 3D Generative Adversarial Networks. In *arXiv*.

Robert L. Cook, John Halstead, Maxwell Planck, and David Ryu. 2007. Stochastic Simplification of Aggregate Detail. *ACM Trans. Graph.* 26, 3 (2007), 79–es.

Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. 2011. Interactive indirect illumination using voxel cone tracing. In *Comp. Graph. Forum*, Vol. 30. 1921–1930.

Hong Deng, Yang Liu, Beibei Wang, Jian Yang, Lei Ma, Nicolas Holzschuch, and Ling-Qi Yan. 2022. Constant-Cost Spatio-Angular Prefiltering of Glinty Appearance Using Tensor Decomposition. *ACM Trans. Graph.* 41, 2, Article 22 (2022).

Stavros Diolatzis, Julien Philip, and George Drettakis. 2022. Active Exploration for Neural Global Illumination of Variable Scenes. *ACM Trans. Graph.* 41, 5, Article 171 (2022).

Michael Garland and Paul S. Heckbert. 1997. Surface Simplification Using Quadric Error Metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., USA, 209–216. https://doi.org/10.1145/258734.258849

David Harbecke, Yuxuan Chen, Leonhard Hennig, and Christoph Alt. 2022. Why only Micro-F1? Class Weighting of Measures for Relation Classification. In *Proc. of NLP Power!Efficient Benchmarking in NLP*. 32–41.

Jon Hasselgren, Jacob Munkberg, Jaakko Lehtinen, Miika Aittala, and Samuli Laine. 2021. Appearance-Driven Automatic 3D Model Simplification. In *Eurographics Symp. Rend.* 85–97.

Eric Heitz, Jonathan Dupuy, Cyril Crassin, and Carsten Dachsbacher. 2015. The SGGX Microflake Distribution. *ACM Trans. Graph.* 34, 4, Article 48 (2015).

Eric Heitz and Fabrice Neyret. 2012. Representing Appearance and Pre-filtering Subpixel Data in Sparse Voxel Octrees. In *High Perf. Graph.*, Carsten Dachsbacher, Jacob Munkberg, and Jacopo Pantaleoni (Eds.).

Wenzel Jakob. 2013. *Light transport on path-space manifolds*. Ph. D. Dissertation. Cornell University.

Simon Kallweit, Thomas Müller, Brian Mcwilliams, Markus Gross, and Jan Novák. 2017. Deep Scattering: Rendering Atmospheric Clouds with Radiance-Predicting Neural Networks. *ACM Trans. Graph.* 36, 6, Article 231 (2017).

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Leif Kobbelt, Swen Campagna, and Hans-Peter Seidel. 1998. A General Framework for Mesh Decimation. In *Proc. Graph. Interface*. 43–50.

Alexandr Kuznetsov, Krishna Mullia, Zexiang Xu, Miloš Hašan, and Ravi Ramamoorthi. 2021. NeuMIP: Multi-Resolution Neural Materials. *ACM Trans. Graph.* 40, 4, Article 175 (jul 2021), 13 pages. https://doi.org/10.1145/3450626.3459795

Alexandr Kuznetsov, Xuezheng Wang, Krishna Mullia, Fujun Luan, Zexiang Xu, Milos Hasan, and Ravi Ramamoorthi. 2022. Rendering Neural Materials on Curved Surfaces. In *ACM SIGGRAPH 2022 Conference Proceedings* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. Association for Computing Machinery, New York, NY, USA, Article 9, 9 pages. https://doi.org/10.1145/3528233.3530721

Jaakko Lehtinen. 2007. A Framework for Precomputed and Captured Light Transport. *ACM Trans. Graph.* 26, 4 (2007), 13–es.

Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. 2018. Noise2Noise: Learning Image Restoration without Clean Data. In *Int. Conf. Machine Learning*. 2971–2980.

Guillaume Loubet and Fabrice Neyret. 2017. Hybrid mesh-volume LoDs for all-scale pre-filtering of complex 3D assets. *Computer Graphics Forum* 36, 2 (2017), 431–442.
Guillaume Loubet and Fabrice Neyret. 2018. A new microflake model with microscopic self-shadowing for accurate volume downsampling. *Computer Graphics Forum* 37, 2 (2018), 111–121.
Linjie Lyu, Ayush Tewari, Thomas Leimkuehler, Marc Habermann, and Christian Theobalt. 2022. Neural Radiance Transfer Fields for Relightable Novel-view Synthesis with Global Illumination. In *ECCV*.
Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval.* Cambridge University Press.
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 Müller. 2021. *tiny-cuda-nn.* https://github.com/NVlabs/tiny-cuda-nn
Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *CoRR* abs/2201.05989 (2022).
Ken Museth. 2021. NanoVDB: A GPU-Friendly and Portable VDB Data Structure For Real-Time Rendering And Simulation. In *ACM SIGGRAPH Talks (SIGGRAPH)*. Article 1, 2 pages.
Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. 2021. HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields. *ACM Trans. Graph.* 40, 6, Article 238 (dec 2021).
Rolandos Potamias, Stylianos Ploumpis, and Stefanos Zafeiriou. 2022. Neural Mesh Simplification. 18562–18571.
Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2021. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
Enrico Puppo and Roberto Scopigno. 1997. Simplification, LOD and Multiresolution Principles and Applications. In *Eurographics - Tutorials*. Eurographics Association.
Gilles Rainer, Adrien Bousseau, Tobias Ritschel, and George Drettakis. 2022. Neural Precomputed Radiance Transfer. *Comp. Graph. Forum (Proc. Eurographics)* 41, 2 (2022).
Peiran Ren, Jiaping Wang, Minmin Gong, Stephen Lin, Xin Tong, and Baining Guo. 2013. Global Illumination with Radiance Regression Functions. *ACM Trans. Graph.* 32, 4, Article 130 (2013).
Tobias Ritschel, Carsten Dachsbacher, Thorsten Grosch, and Jan Kautz. 2012. The state of the art in interactive global illumination. In *Comp. Graph. Forum*, Vol. 31. 160–188.
P.Y. Simard, D. Steinkraus, and J.C. Platt. 2003. Best practices for convolutional neural networks applied to visual document analysis. In *Doc. Anal. and Recogn.* 958–963.
Peter-Pike Sloan. 2008. Stupid Spherical Harmonics (SH) Tricks. *Game Developers Conference* (01 2008).
Peter-Pike Sloan, Jan Kautz, and John Snyder. 2002. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. *ACM Trans. Graph.* 21, 3 (2002), 527—-536.
Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. 2023. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics* (2023).
Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. 2022. Variable Bitrate Neural Fields. In *ACM Trans. Graph.* Article 41.
Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, W Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. 2022. Advances in neural rendering. In *Comp. Graph. Forum*, Vol. 41. 703–735.
Eric Veach. 1997. *Robust Monte Carlo Methods for Light Transport Simulation.* Ph. D. Dissertation. Stanford University.
Delio Vicini, Wenzel Jakob, and Anton Kaplanyan. 2021. A Non-Exponential Transmittance Model for Volumetric Scene Representations. *ACM Trans. Graph.* 40, 4, Article 136 (2021).
Beibei Wang, Wenhua Jin, Miloš Hašan, and Ling-Qi Yan. 2022a. SpongeCake: A Layered Microflake Surface Appearance Model. *ACM Trans. Graph.* 42, 1, Article 8 (2022).
Liao Wang, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, Yingliang Zhang, Minye Wu, Jingyi Yu, and Lan Xu. 2022b. Fourier PlenOctrees for Dynamic Radiance Field Rendering in Real-Time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 13524–13534.
E Woodcock. 1965. Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. *Conf. App. Comp. Methods to Reactor Problems* 557 (1965).
Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. 2022. Neural fields in visual computing and beyond. In *Comp. Graph. Forum*, Vol. 41. 641–676.
Zilin Xu, Zheng Zeng, Lifan Wu, Lu Wang, and Ling-Qi Yan. 2022. Lightweight Neural Basis Functions for All-Frequency Shading. In *SIGGRAPH Asia Conf. Papers (SA)*. Article 14, 9 pages.
Ling-Qi Yan, Miloš Hašan, Wenzel Jakob, Jason Lawrence, Steve Marschner, and Ravi Ramamoorthi. 2014. Rendering Glints on High-Resolution Normal-Mapped Specular Surfaces. *ACM Trans. Graph.* 33, 4, Article 116 (2014).
Junqiu Zhu, Sizhe Zhao, Lu Wang, Yanning Xu, and Ling-Qi Yan. 2022. Practical Level-of-Detail Aggregation of Fur Appearance. *ACM Trans. Graph.* 41, 4, Article 47 (2022).

A GENERALISED VOXEL PATH SPACE MEASURES

In Section 3.1, we introduced our generalised voxel path space $\mathcal{X}$, here we show how it can be constructed using the configuration vector $\bar{\boldsymbol{c}} = (\boldsymbol{c}_0, ..., \boldsymbol{c}_{l-1}) \in \{0,1\}^l$. We recall that $\boldsymbol{x}_j = \boldsymbol{\omega}_j$ if $\boldsymbol{c}_j = 0$ and $\boldsymbol{x}_j = (\boldsymbol{p}'_j, \boldsymbol{\omega}_j)$ if $\boldsymbol{c}_j = 1$. Therefore, our generalised voxel path space is defined as:

$$\mathcal{X} = \bigcup_{l=1}^{\infty} \bigcup_{\bar{\boldsymbol{c}} \in \{0,1\}^l} \mathcal{X}_{\bar{\boldsymbol{c}}}^{(l)}, \quad \mathcal{X}_{\bar{\boldsymbol{c}}}^{(l)} = \times_{i=0}^{l-1} \begin{cases} \mathcal{M}' \times \Omega & \text{if } \boldsymbol{c}_i = 1, \\ \Omega & \text{otherwise.} \end{cases},$$

with corresponding measure definitions

$$\mu'(X) = \sum_{l=1}^{\infty} \sum_{\bar{\boldsymbol{c}} \in \{0,1\}^l} \mu'^{(l)}\left(X \cap \mathcal{X}_{\bar{\boldsymbol{c}}}^{(l)}\right).$$

$$d\mu'^{(l)}(\bar{\boldsymbol{x}}) = \prod_{j=0}^{l-1} \begin{cases} dA(\boldsymbol{p}'_j) dS(\boldsymbol{\omega}_j) & \text{if } \boldsymbol{c}_j = 1, \\ dS(\boldsymbol{\omega}_j) & \text{otherwise.} \end{cases}$$