

Double Bounded Neural Ray Queries

Alexander Nikolaev¹, Nikolay Mozokhin¹, Roman Rodionov¹ and Vladimir Frolov²

¹Lomonosov Moscow State University

²Institute for Artificial Intelligence, Lomonosov Moscow State University

{alexander.nikolaev, nikolay.mozokhin, roman.rodionov, vladimir.frolov}@graphics.cs.msu.ru

Abstract

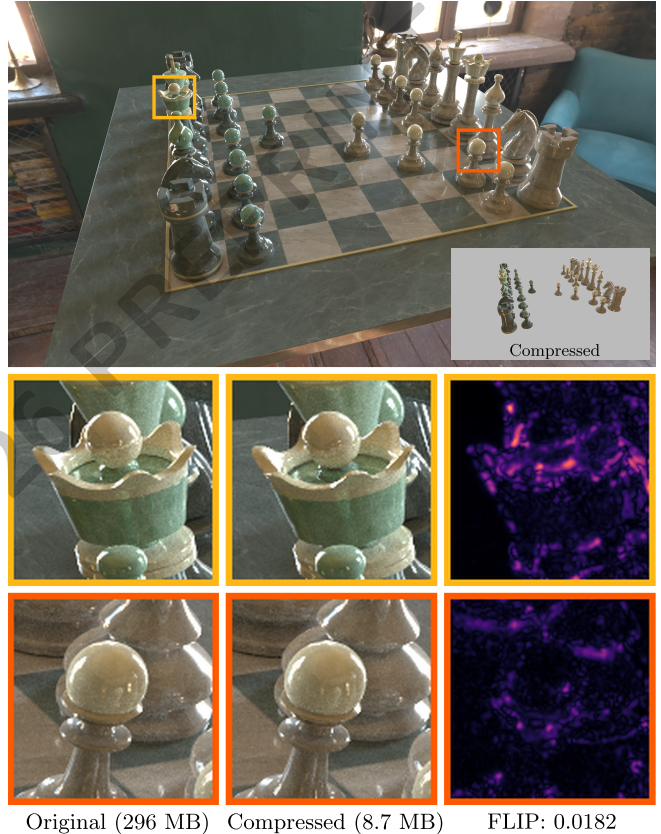
We introduce a novel neural ray tracing method designed for compact scene representations and real-time rendering. To compress the scene with minimal fidelity losses, we address the issue of limiting the search space for intersections. For each scene we first construct two lightweight proxy shells that tightly bound the original surface from inside and outside. While executing ray queries, we intersect the rays with the shells and extract the regions that potentially contain the intersection with the original surface. Extracted regions are passed to the small neural network to retrieve the exact intersection location. We implement our method as part of a GPU-accelerated hybrid path tracing pipeline. We demonstrate it running real-time rendering on a variety of scenes, achieving up to 300× memory reduction and surpassing existing compressed ray tracing techniques in memory-quality trade-off.

1 Introduction

Modern rendering applications demand the storage and real-time rendering of highly detailed geometry. Dominant industry approaches rely on triangle meshes, which are relatively cheap to evaluate but memory intensive [Maggiordomo *et al.*, 2023; Haydel *et al.*, 2023]. Recent neural compression methods achieve favorable quality-to-size trade-offs but remain computationally expensive for real-time rendering [Takikawa *et al.*, 2021; Sivaram *et al.*, 2024; Zhang *et al.*, 2025; Chabra *et al.*, 2020]. Other works introduce compact spatial encoding and enable relatively efficient sampling and queries [Kim *et al.*, 2024; Zakharov *et al.*, 2024; Müller *et al.*, 2022; Takikawa *et al.*, 2022; Takikawa *et al.*, 2023], yet they are still costly for ray tracing, as multiple neural network evaluations are required per ray query. Neural ray tracing techniques reduce computational cost by limiting network evaluations, operating directly on rays [Weier *et al.*, 2024; Fujieda *et al.*, 2025].

In this paper, we introduce a neural ray tracing method¹ that achieves real-time performance while surpassing exist-

¹To access the code and additional materials, please see <https://github.com/Alehandreus/double-bounded-rt>



Original (296 MB) Compressed (8.7 MB) FLIP: 0.0182

Figure 1: Designed for ray tracing, our method achieves high fidelity and massive memory footprint reduction for selected geometry.

ing approaches in compression-to-quality trade-off. Our contributions are as follows:

- We propose a novel neural geometry compression method designed for ray tracing that achieves up to 300× memory reduction with minimal quality loss;
- We devise an algorithm that mitigates discontinuities in the target function by splitting rays into multiple segments and evaluating them separately;
- We implement an efficient hybrid inference pipeline, demonstrating real-time path tracing on complex scenes.

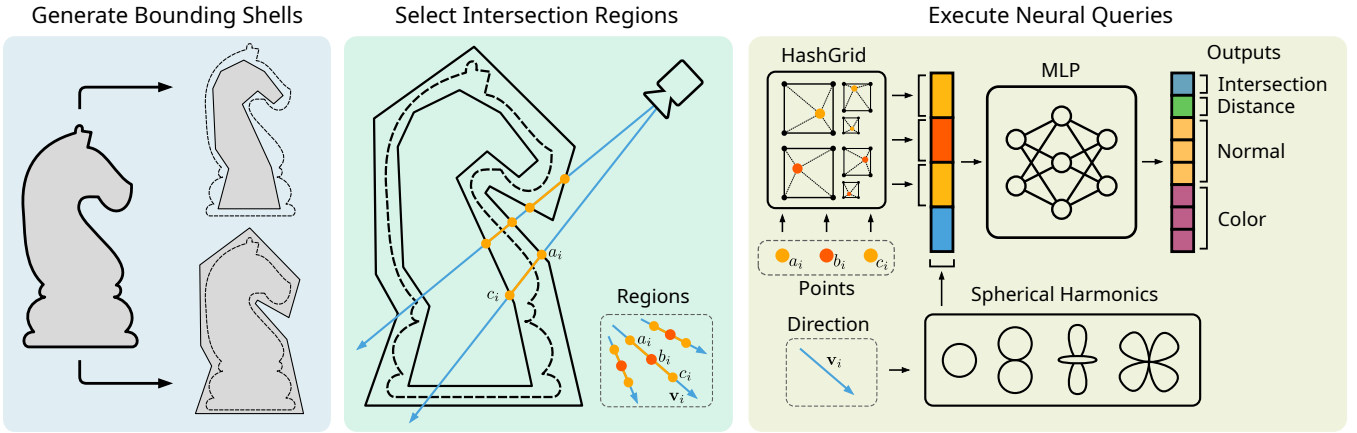


Figure 2: Given the original surface, we generate two coarse bounding meshes. During ray tracing we intersect the rays with both shells, extracting ray segments (a_i, c_i) and directions $\mathbf{v}_i$ for further processing. a_i, c_i , and the midpoint b_i are encoded with a multi-resolution hash grid encoding. The direction vector $\mathbf{v}_i$ is encoded with spherical harmonics. All embeddings are concatenated and passed to a small MLP that provides the color, normal and distance to the intersection point, as well as a binary classification output indicating whether an intersection exists within the segment. If there is no intersection, we consider the next segment on a ray.

2 Related Work

Neural ray tracing relies on extracting the intersection point given both the origin and the direction of a ray [Fujieda *et al.*, 2023]. Unlike a signed distance field, this function suffers from discontinuities: for non-convex objects, a slight movement of ray position can lead to a sharp change of the hit point coordinates. Various encodings and auxiliary losses were proposed to ensure consistency [Aumentado-Armstrong *et al.*, 2022; Feng *et al.*, 2022; Liu *et al.*, 2023].

Prior work has shown that the largest accuracy improvements come from limiting the search space. [Fujieda *et al.*, 2025] encloses the original surface in a sparse voxel grid, while [Weier *et al.*, 2024] constructs a shallow BVH tree. The network is then provided with the narrow region that potentially contains the intersection point.

However, both voxels and BVH nodes form an uneven surface; furthermore, axis-aligned bounding boxes cannot ensure tight bounding for most geometries, exacerbating the discontinuity issue.

3 Proposed Pipeline

To mitigate the problem of frequent bumps in the target function, we propose to enclose the original surface in two coarse bounding meshes. For each target object, we construct an inner and an outer shell using the Bounding Mesh algorithm [Gaschler *et al.*, 2015].

3.1 Rendering

During inference, only lightweight shells and the neural network are available. For each ray, we first extract a region possibly containing the intersection with the original surface. The first point a_i of a region is the entry point to the outer shell. The second point c_i is either the exit point from the outer shell or the entry point to the inner shell, whichever is closer to a_i . Each new outer shell entry forms a new region (Figure 2).

To improve sensitivity to small details, we spatially encode the regions. The start a_i , end c_i , and middle b_i points of a region are encoded with a multi-resolution hash grid encoding [Müller *et al.*, 2022]. The hash grid contains 16 levels of 2^n 4-dimensional features, n is left as a hyperparameter. The direction of a ray $\mathbf{v}_i$ is encoded with 16 spherical harmonics. All embeddings are concatenated and passed to an MLP containing 4 hidden layers of 128 neurons. The MLP outputs the color, normal, distance to the intersection point, and a binary classification result for the existence of an intersection point between the ray and the original surface inside the segment. To minimize the number of neural network evaluations, we process all ray segments in order: we first traverse the full BVH of coarse shells to extract the ray segment closest to the ray origin. If the neural query classifies the segment as not containing the intersection, we do another traversal pass to extract the next segment.

The renderer is a wavefront path tracer implemented in CUDA. To improve performance, we run neural inference in half precision, perform ray compaction after each bounce and neural query, and employ CUDA-OpenGL interoperation to reduce transfers between host and device. Unlike [Weier *et al.*, 2024], our implementation does not require injecting neural inference amid BVH traversal and takes advantage of hardware-accelerated ray tracing for shell intersection.

Analogous to [Weier *et al.*, 2024], our rendering framework supports hybrid rendering of scenes with partial compression. Each ray is traced against both uncompressed parts of the scene and neural objects.

3.2 Training

We train in batches of 10^5 rays sampled dynamically. Directions are generated as random unit vectors and origins are sampled on the outer mesh surface to ensure good coverage of the scene. We use Adam [Kingma and Ba, 2014] optimizer with a learning rate of 10^{-3} and train for 200k iterations with EMA [Polyak and Juditsky, 1992]. We employ binary cross-

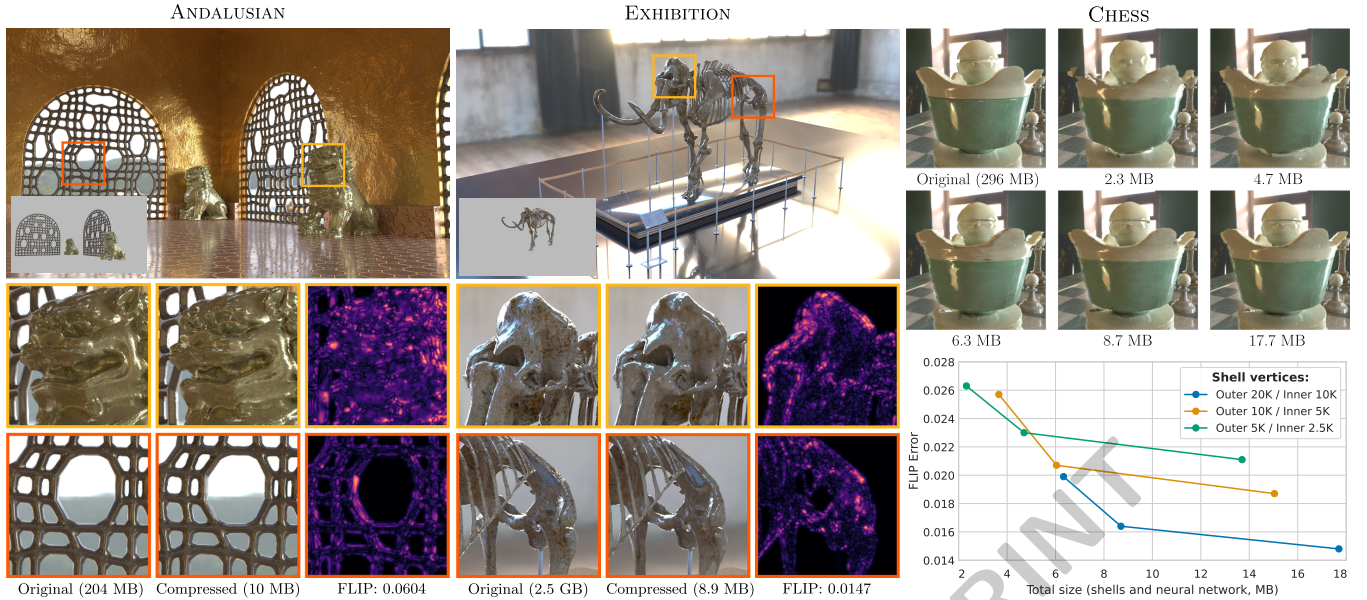


Figure 3: Left and middle: ANDALUSIAN and EXHIBITION scenes at 1920×1080 resolution. Corner images indicate compressed parts. Right: CHESS scene at different compression rates. For each graph line, shells are fixed and hashgrid size iterates over 2^{14} , 2^{16} , and 2^{18} .

Scene	Uncompressed		Ours			N-BVH		
	Size (MB)	FPS	Size (MB)	FPS ↑	FLIP ↓	Size (MB)	FPS ↑	FLIP ↓
CHESS	296	594	8.70	115	0.0165	69		0.0278
ANDALUSIAN	204	267	9.97	61	0.0604	11.23	27	0.0722
EXHIBITION	2521	545	8.88	124	0.0147	38		0.0274
STATUETTE	642	970	8.94	155	0.0049	97		0.0126

Table 1: Compared to N-BVH, our method demonstrates better quality at the same compression level and $1.6 - 3.2\times$ better performance. For our method, we use hash grid of size 2^{16} (3.4 MB), outer shell of approximately 20k vertices and inner shell of 10k vertices (5-6 MB). For N-BVH, we use hash grid of size 2^{16} (2.9 MB) and 275k nodes (8.33 MB). All performance metrics are reported for NVIDIA RTX 5090. Original sizes are reported only for selected geometry.

entropy loss for the classification output, L_2 loss for the color and L_1 loss for the distance and the normal. The final loss is the weighted sum of all components:

$$L = L_{cls} \cdot 0.1 + L_{dist} \cdot 0.01 + L_{normal} + L_{color} \cdot 10.$$

The training pipeline is implemented in PyTorch, with a custom CUDA library for BVH traversal. On a single NVIDIA RTX 5090, training takes between 30 and 60 minutes depending on the scene.

4 Results

We compute the original memory cost as the sum of buffers allocated for vertex positions, face indices, UV coordinates and textures, as well as BVH nodes. After compression, we consider inner and outer shells, which require only vertex positions and face indices, and the neural network parameters.

We evaluate our method on four scenes introduced in [Weier *et al.*, 2024] (Figure 1 for the CHESS scene and Figure 3 for others) and compare with N-BVH [Weier *et al.*, 2024]. We configure our renderer to match the reference renderer of N-BVH. For all scenes, we use hash grid of size 2^{16} for both

methods. We also define space-limiting structures of both methods to be approximately the same size: for our method, we use outer shells of approximately 20k vertices and inner shells of 10k vertices (5-6 MB), while for N-BVH we use 275k nodes (8.33 MB). Due to tighter bounding, our method achieves significantly better quality at a slightly better compression level. With our method being designed for hardware RT, we achieve $1.6 - 3.2\times$ better performance compared to N-BVH (Table 1).

Towards higher precision, the shell size becomes the primary factor (Figure 3, right), suggesting the hashgrid is most efficient at heavy compression configurations.

5 Conclusion

We propose a novel compressed ray tracing method focused on reducing the search space for intersection. Compared to existing approaches, our method supports hardware-accelerated ray tracing and is superior in both quality and performance. The hybrid path tracing pipeline is efficient and ready to be integrated into existing physically based rendering engines.

Acknowledgments

The work of Alexander Nikolaev and Roman Rodionov was supported by the The Ministry of Economic Development of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000C313925P4H0002; grant No 139-15-2025-012).

References

- [Aumentado-Armstrong *et al.*, 2022] Tristan Aumentado-Armstrong, Stavros Tsogkas, Sven Dickinson, and Allan Jepson. Representing 3d shapes with probabilistic directed distance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19343–19354, 2022.
- [Chabra *et al.*, 2020] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX*, page 608–625, Berlin, Heidelberg, 2020. Springer-Verlag.
- [Feng *et al.*, 2022] Brandon Feng, Yinda Zhang, Danhang Tang, Ruofei Du, and Amitabh Varshney. Prif: Primary ray-based implicit function. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*, page 138–155, Berlin, Heidelberg, 2022. Springer-Verlag.
- [Fujieda *et al.*, 2023] Shin Fujieda, Chih Chen Kao, and Takahiro Harada. Neural Intersection Function. In Jacco Bikker and Christiaan Gribble, editors, *High-Performance Graphics - Symposium Papers*. The Eurographics Association, 2023.
- [Fujieda *et al.*, 2025] Shin Fujieda, Chih-Chen Kao, and Takahiro Harada. Lsnif: Locally-subdivided neural intersection function. volume 8, New York, NY, USA, May 2025.
- [Gaschler *et al.*, 2015] Andre Gaschler, Quirin Fischer, Alois Knoll, and Technische Universität München Lehrstuhl für Rechnerarchitektur und Rechnerorganisation. *The Bounding Mesh Algorithm*. TUM-I. TUM Technische Universität München, Institut für Informatik, 2015.
- [Haydel *et al.*, 2023] Jacob Haydel, Cem Yuksel, and Larry Seiler. Locally-adaptive level-of-detail for hardware-accelerated ray tracing. *ACM Trans. Graph.*, 42(6), December 2023.
- [Kim *et al.*, 2024] Doyub Kim, Minjae Lee, and Ken Museth. Neuralvdb: High-resolution sparse volume representation using hierarchical neural networks. *ACM Trans. Graph.*, 43(2), 2024.
- [Kingma and Ba, 2014] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2014.
- [Liu *et al.*, 2023] Zhuoman Liu, Bo Yang, Yan Luximon, Ajay Kumar, and Jinxi Li. Raydf: Neural ray-surface distance fields with multi-view consistency. In *Advances in Neural Information Processing Systems*, volume 36, pages 25238–25267, 2023.
- [Maggiordomo *et al.*, 2023] Andrea Maggiordomo, Henry Moreton, and Marco Tarini. Micro-mesh construction. *ACM Trans. Graph.*, 42(4), July 2023.
- [Müller *et al.*, 2022] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4), July 2022.
- [Polyak and Juditsky, 1992] Boris Polyak and Anatoli Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.
- [Sivaram *et al.*, 2024] Venkataram Sivaram, Tzu-Mao Li, and Ravi Ramamoorthi. Neural geometry fields for meshes. In *ACM SIGGRAPH 2024 Conference Papers*, SIGGRAPH ’24, 2024.
- [Takikawa *et al.*, 2021] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11353–11362, 2021.
- [Takikawa *et al.*, 2022] Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. Variable bitrate neural fields. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings*, SIGGRAPH ’22, page 1–9. ACM, August 2022.
- [Takikawa *et al.*, 2023] Towaki Takikawa, Thomas Müller, Merlin Nimier-David, Alex Evans, Sanja Fidler, Alec Jacobson, and Alexander Keller. Compact neural graphics primitives with learned hash probing. In *SIGGRAPH Asia 2023 Conference Papers*, SA ’23, New York, NY, USA, 2023. Association for Computing Machinery.
- [Weier *et al.*, 2024] Philippe Weier, Alexander Rath, Élie Michel, Iliyan Georgiev, Philipp Slusallek, and Tamy Boubekeur. N-bvh: Neural ray queries with bounding volume hierarchies. In *ACM SIGGRAPH 2024 Conference Papers*, SIGGRAPH ’24, 2024.
- [Zakharov *et al.*, 2024] Sergey Zakharov, Katherine Liu, Adrien Gaidon, and Rares Ambrus. Refine: Recursive field networks for cross-modal multi-scene representation. In *ACM SIGGRAPH 2024 Conference Papers*, SIGGRAPH ’24, New York, NY, USA, 2024. Association for Computing Machinery.
- [Zhang *et al.*, 2025] Congyi Zhang, Jinfan Yang, Eric Hedlin, Suzuran Takikawa, Nicholas Vining, Kwang Moo Yi, Wenping Wang, and Alla Sheffer. Nesi: Neural explicit-shape-intersection-based geometry representation. *ACM Trans. Graph.*, 44(5), July 2025.