Ezra Tock
Thesis Advisor: Prof. Yuhao Zhu
December 1st, 2025

Analysis of N-BVH Ray Tracing for Remote Use and for Extension to Dynamic Scenes

Contents

1. Introduction

- Introduction to ray tracing
- Bounding volume hierarchies

Ray Tracing

Backward Ray Tracing: simulate light transport by casting rays from the camera, through each pixel, onto the scene

- More realistic than rasterization
- Computing ray-geometry intersection tests is the key computational challenge

Applications: 3D rendering

- High-fidelity film and VFX, photo-realistic architectural visualization, scientific simulations
- 3D gaming, training simulations, social VR spaces, preview rendering

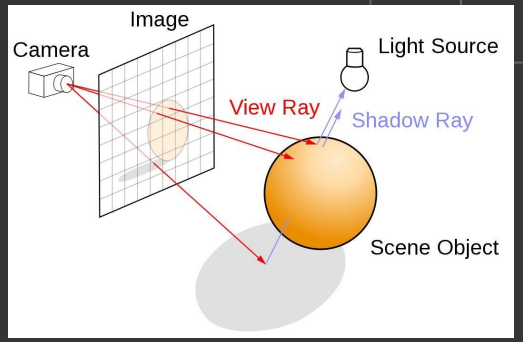


Image from NVIDIA Developer “Ray Tracing”

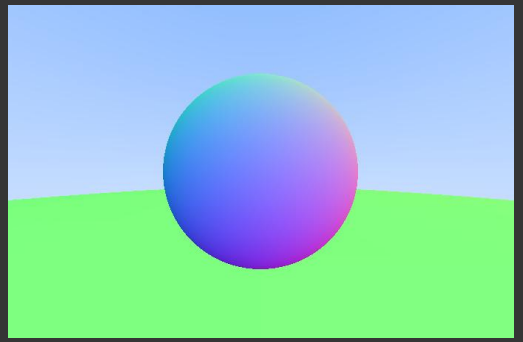


Image from Peter Shirley et. al “Ray Tracing in One Weekend”

Bounding Volume Hierarchies (BVHs)

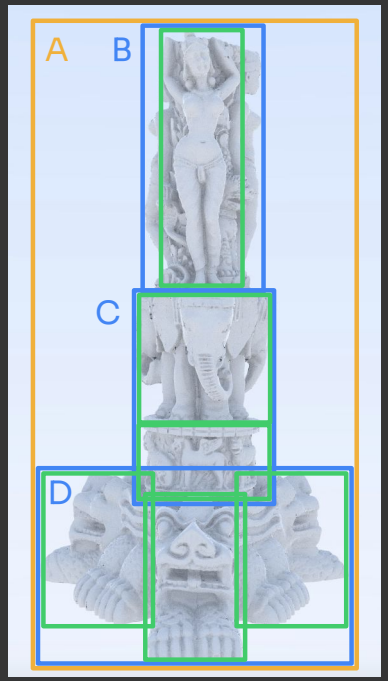
Geometry is typically represented by a mesh with millions of triangles

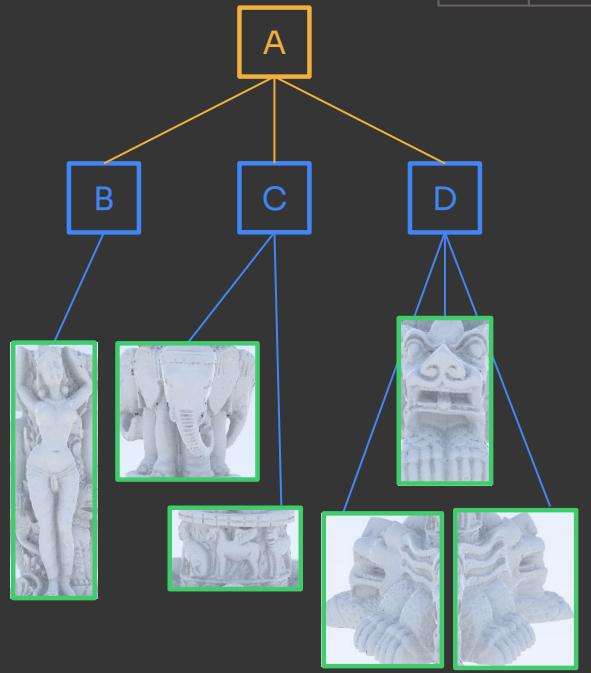
Main idea: organize the geometry into a tree of axis aligned bounding boxes (AABBs) to cull large portions of the scene for each ray query

O(log N) instead of O(N)

Further optimizations:

- Surface area heuristics (SAHs)
- TLAS and BLAS





"Statuette" scene from Weier et. al[9]

2. Literature Review

- Fujieda et al. “Neural Intersection function” (NIF)
- Weier et al. “N-BVH”

Fujieda et al. “Neural Intersection Function” (2023)

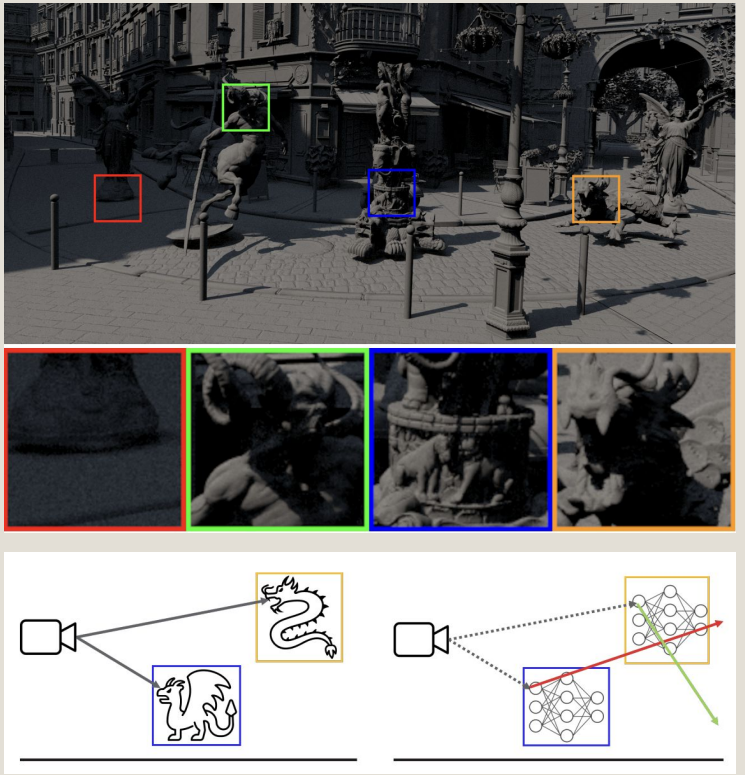
Key insight: Traditional BVHs are bulky and require irregular memory access patterns which are inefficient on GPUs

Design:

- Encode BLAS traversal queries with feature grids and feed them to small MLPs
- Only practical for secondary rays: primary rays produce unacceptable artifacts[3]

MLP Architecture:

- Dual-network strategy inspired by surface partitioning[4][7] and space partitioning[6][8] literature
- Removes the need for a signed distance parameter



NIF example render (upper) and rendering pipeline (lower) from Fujieda et al. “Neural Intersection Function”

Fujieda et al. “Neural Intersection Function” (2023)

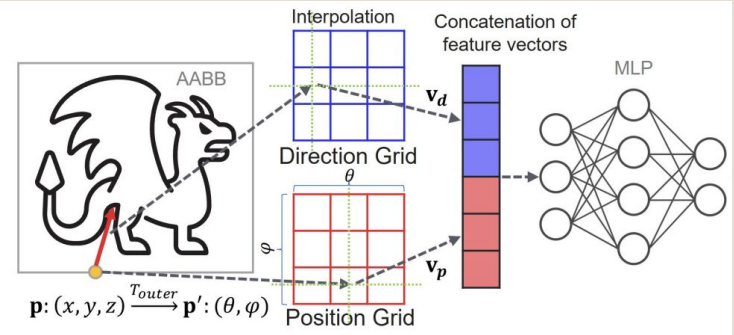
Ray aliasing: overlapping rays with different origins should produce the same intersection result, but neural networks were inconsistent

- **Origin normalization:** for the outer network, translate the origin to the bounding box intersection, and for the inner network, translate the origin to the surface

Ray encoding:

- High frequency geometry limits direct parameterization of position and direction, so use 128 or 256 res feature grids
- Convert to spherical coordinates to reduce the dimension

Results: 35% faster traversal, 256KB instead of multiple GB per object, and 31-39 dB reconstruction accuracy on the PSNR scale[3]



NIF outer network encoding scheme from Fujieda et al. “Neural Intersection Function” The inner network requires an additional distance feature in case the ray intersects an object at multiple locations, which gets looked up in an additional 1D grid, and gets concatenated along with the other feature vectors

Weier et al. “N-BVH: Neural Ray Queries with Bounding Volume Hierarchies” (2024)

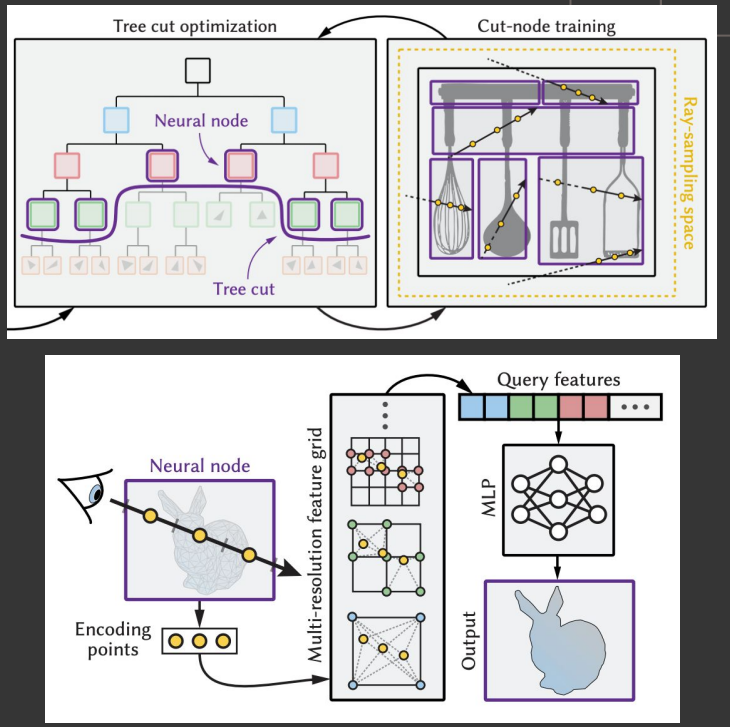
Key insight: NIF is viewpoint-dependent which limits its use in fully functional ray tracing pipelines

Design Strategies:

- Custom tree-cut structure for more strategic neural network replacement as opposed to NIF’s entire BLAS replacement
- Adaptive depth for different geometric complexities by training on prediction loss×ray-intersection probability

Ray Encoding:

- Encode 3-4 sampled points along the ray-box intersection interval for higher likelihood of a point close to the surface
- Use multi-resolution hash grids and trilinear interpolation



NBVH tree-cut structure and rendering pipeline[9]

Weier et al. “N-BVH: Neural Ray Queries with Bounding Volume Hierarchies” (2024)

Pipeline Phase	Estimated Time for Statuette	Output
Geometry Loading	~8 seconds	Triangle mesh in memory
BVH Construction	~25 seconds	Full traditional BVH
Tree Cut Initialization	<1 second	Initial NBVH structure
Training Ray Generation	~15 seconds	Ray query/response pairs
Neural Network Training	~90 seconds	Trained NBVH weights
Tree Cut Optimization	~10 seconds	Final NBVH structure
NBVH Loading	~2 seconds	NBVH in GPU memory
Rendering	16-33ms	Rendered frames

Component	Size	Size for Statuette
Original Geometry Representation	~64B × T	642MB
Feature Grids	$4B \times O \times 56R^2$	~10MB
MLP	4B × L × W	~300KB
Tree-Cut BVH	64B × N	~700KB
Total NBVH	Sum of the above	11.2MB (57x compression)

Time estimates for the different phases of the NBVH pipeline for Weier’s Statuette scene on his NVIDIA RTX 3090 GPU based on various implementation hints (left), and Space requirements for different components of the NBVH structure (right).

- Neural Network training dominates training time due to over 2 million parameters
- Feature grids dominate memory footprint
- 57x compression atones 2-4x slower rendering

3. Remote Ray Tracing

- Fully remote pipelines
- Split pipelines

Fully Remote Pipelines

My setup: TurboVNC and VirtualGL

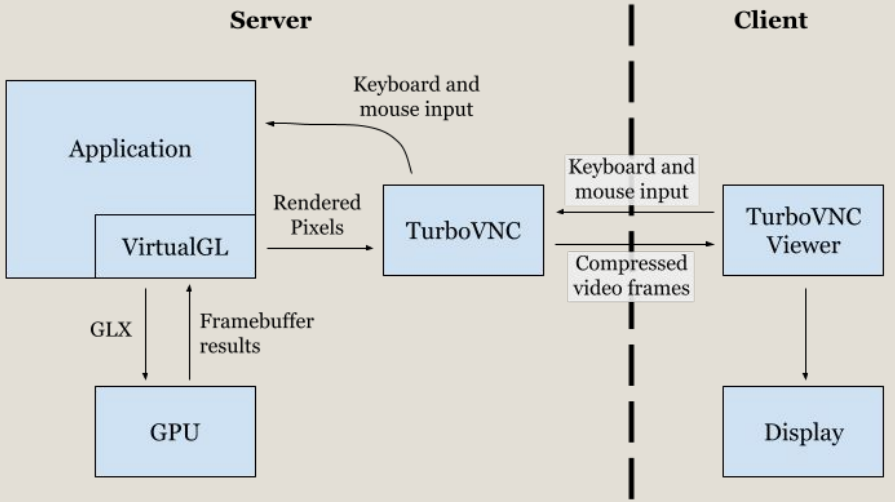
- TurboVNC creates a virtual X server
- VirtualGL intercepts GLX calls and redirects them to the physical GPU
- Poor encoding efficiency

Traditional setup: Parsec, NICE DVC

- Hardware-accelerated encoding
- About 70% of typical end-to-end latency is network traversal: render-time is trivial enough

Applications: Cloud gaming

- 2-4x slower render time, but having multiple concurrent users improves GPU utilization



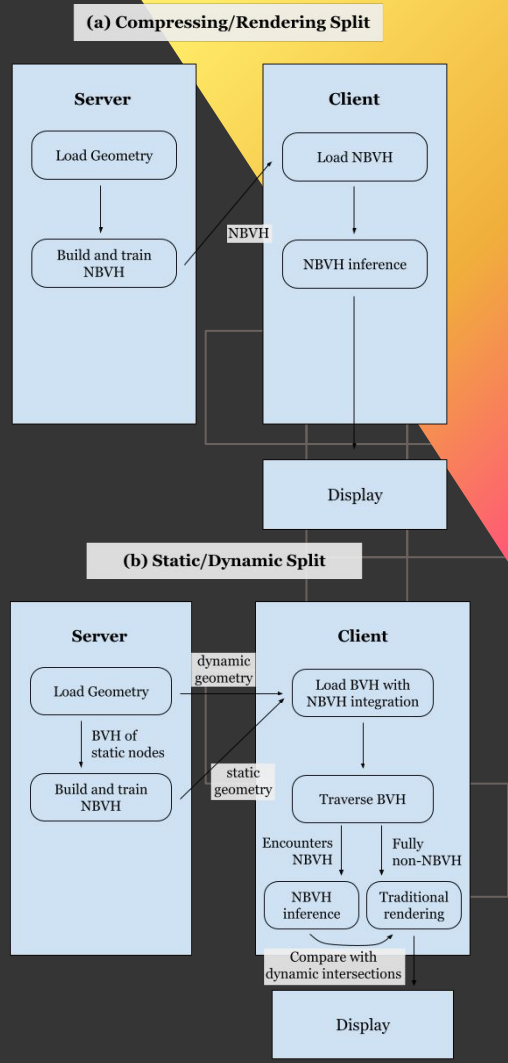
Split Pipelines

Compressing/Rendering Split: full NBVH compression on the server, transfer the NBVH for local rendering

- 11MB Statuette scene transfers in seconds over 50-100Mbps broadband
- **Applications:** Compress app size of predetermined static scenes, with potential manual tuning for certain objects' feature grids

Static/Dynamic Split: Static geometry is compressed on the server-side once, and dynamic geometry remains as a triangle mesh for traditional rendering

- Joint BVH and NBVH traversal requires both traditional ray-mesh intersection tests and MLP inference when encountering an NBVH node
- **Applications:** Games with large static backgrounds and few dynamic characters



Splits at Intermediate Stages of NBVH

Pipeline Phase	Estimated Time for Statuette	Output
1. Geometry Loading	~8 seconds	Triangle mesh in memory
2. BVH Construction	~25 seconds	Full traditional BVH
3. Tree Cut Initialization	<1 second	Initial NBVH structure
4. Training Ray Generation	~15 seconds	Ray query/response pairs
5. Neural Network Training	~90 seconds	Trained NBVH weights
6. Tree Cut Optimization	~10 seconds	Final NBVH structure
7. NBVH Loading	~2 seconds	NBVH in GPU memory
8. Rendering	16-33ms/frame	Rendered frames

Example Split: Steps 1-3 remote, steps 4-8 local

- Neural Network Training requires the full scene geometry—defeats the purpose of NBVH
- Theoretically saves marginal CPU time, but it doesn't make sense to do NBVH locally cloud GPU resources are readily available

4. N-BVH for Dynamic Scenes

- Complete reconstruction and delta compression
- Static/dynamic split

Complete Reconstruction and Delta Compression

Complete reconstruction:

- Pros: no accumulated error, simple update logic
- Compressing Statuette is 9,000× slower than 60fps which is unacceptable for both continuous reconstruction and temporal caching
- MLP training already saturates GPU capacity

Delta compression:

- Freezing the MLP only saves 30-40% of training time because the forward pass of the MLP is still required, and accuracy degrades
- Only retraining updated hash entries also causes retraining distant unchanged regions
- Optimistic speedup of ~10×—still unacceptable

Static/Dynamic Partitioning

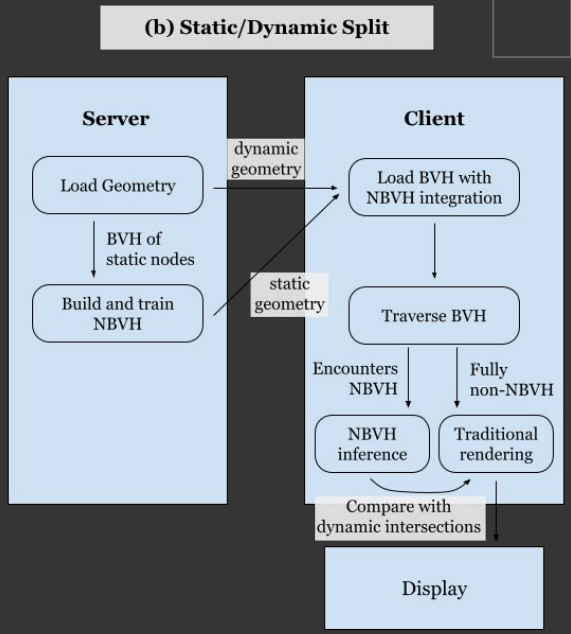
Joint BVH and NBVH traversal:

- Mark nodes with MLP approximations as NBVH nodes
- When encountering an NBVH node in traversal, ray intersection must also be tested with the dynamic child nodes
- Additional worst case slowdown, but still acceptable

Separate traversal with depth records:

- Additional slowdown for all NBVH ray queries, and repetition of shallow BVH traversal
- Useful for fully remote rendering, server transmits frames with additional 24-32 bit depth information

Useful for applications like cloud gaming, especially when large or repeated static components are involved, and progressive refinement can be used for dynamically determined static content



5. Conclusion

Current Progress and Next Steps

Reflections about effective research:

- Learn by doing
- Have flexible pacing to stay focused on the main goal

Current Progress:

- NBVH can fit into remote ray tracing applications, and could enable 5-10× user density in applications like cloud gaming
- NBVH is fundamentally infeasible for dynamic scenes
- Static/dynamic partitioning is the key feasible approach and allows significant memory compression

Next Steps:

- Verify bottleneck predictions with empirical results
- Implement efficient static/dynamic split for practical applications

Thank you

References

[1] NVIDIA Corporation. "NVIDIA Video Codec SDK: NVENC Programming Guide," NVIDIA Developer Documentation, 2020. [Online]. Available: https://developer.nvidia.com/video-codec-sdk.

[2] Cai, Wei. “A Message from the New IEEE Access Editor-in-Chief.” *IEEE Access*, 2 Oct. 2025, ieeeaccess.ieee.org/featured-articles/survey-cloud-gaming-future-computer-games/.

[3] Fujieda, Shin, et al. “Neural Intersection Function.” *arXiv.Org*, 12 June 2023, arxiv.org/abs/2306.07191.

[4] Genova, Kyle, et al. “Learning Shape Templates with Structured Implicit Functions.” *arXiv.Org*, 12 Apr. 2019, arxiv.org/abs/1904.06447.

[5] MacDonald, J. David, and Kellogg S. Booth. “Heuristics for Ray Tracing Using Space Subdivision - the Visual Computer.” *SpringerLink*, Springer-Verlag, 10 July 2018, link.springer.com/article/10.1007/BF01911006.

[6] Martel, Julien N. P., et al. “Acorn: Adaptive Coordinate Networks for Neural Scene Representation.” *arXiv.Org*, 6 May 2021, arxiv.org/abs/2105.02788.

[7] Tretschk, Edgar, et al. “Patchnets: Patch-Based Generalizable Deep Implicit 3D Shape Representations.” *arXiv.Org*, 5 Feb. 2021, arxiv.org/abs/2008.01639.

[8] Wang, Peng, et al. “Neus: Learning Neural Implicit Surfaces by Volume Rendering for Multi-View Reconstruction.” *arXiv.Org*, 1 Feb. 2023, arxiv.org/abs/2106.10689.

[9] Weier, Philippe, et al. “N-BVH: Neural Ray Queries with Bounding Volume Hierarchies.” *arXiv.Org*, 25 May 2024, arxiv.org/abs/2405.16237.

[10] Yu, Zheng. “Ray Tracing in Computer Graphics.” *Highlights in Science, Engineering and Technology*, 27 Dec. 2022, www.researchgate.net/publication/366663130_Ray_Tracing_in_Computer_Graphics.