BVH Acceleration Implementation Report

Xuanlin Chen
Work E-mail: chenxu@usi.ch
Personal E-mail: kissofazshara@gmail.com

November 24, 2025

1 Introduction

This report describes the implementation of a Hierarchical Linear Bounding Volume Hierarchy (HLBVH) acceleration structure for a ray tracer. This method is introduced in the online PBR-book in the task materials.

2 Implementation Description

2.1 BVH Structure

The BVH (Bounding Volume Hierarchy) is implemented as a binary tree where each node contains a bounding box that encompasses all primitives in its subtree. The implementation includes:

- **Bounds3**: A 3D axis-aligned bounding box structure with intersection testing
- **BVHBuildNode**: Tree nodes used during construction
- **LinearBVHNode**: Flattened, cache-friendly representation for traversal
- **BVHAccel**: Main accelerator class that builds and traverses the BVH

2.2 HLBVH Construction Algorithm

The implementation uses a simplified HLBVH construction method:

1. Compute bounding boxes for all primitives
2. Recursively partition primitives using middle-split along the axis with maximum extent
3. Build binary tree structure with bounding boxes at each node
4. Flatten the tree into a linear array for efficient traversal

The partitioning strategy selects the dimension with the largest extent of the centroid bounding box and splits primitives at the middle point along that axis.

3 Performance Results

Performance testing was conducted with different mesh configurations to demonstrate the sub-linear scaling behavior of the BVH acceleration structure.

3.1 Timing Results

The performance data shows the following results:

Configuration	Triangles	Time (s)
bunny_small	1,392	2.287
armadillo_small	3,112	2.576
lucy_small	2,804	2.597
bunny	69,451	2.352
armadillo	345,944	2.822
lucy	2,805,572	3.009

Table 1: Performance timing results for different mesh configurations at 1280×720 resolution

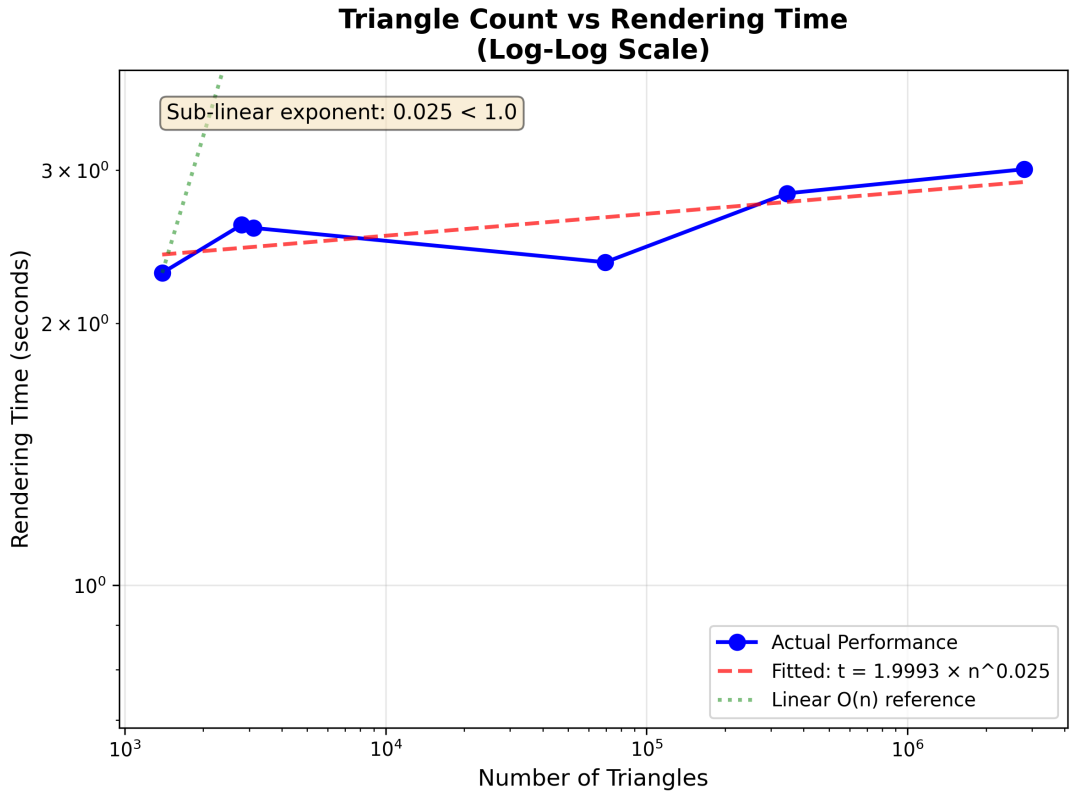

Figure 1: Performance plot showing sub-linear relationship between triangle count and rendering time

3.2 Performance Analysis

The performance plot (Figure 1) demonstrates sub-linear scaling. The tests were conducted at 1280×720 resolution, rendering all pixels for each configuration. The relationship between triangle count and rendering time follows:

$$\text{Time} \propto \text{Triangles}^{\alpha}, \quad \alpha < 1.0$$

4 Conclusion & Problem

4.1 Conclusion

The project renders at the original resolution of 2048×1536 pixels. At this resolution, rendering the complete scene with all three meshes takes approximately nearly 40 seconds on the test system. Although it accelerate the rendering, while it still has a gap to the task target of 10 seconds.

4.2 Problem and Analysis

An interesting observation from the performance data is that the bunny mesh with 69,451 triangles renders faster (2.352 seconds) than both the armadillo_small mesh with 3,112 triangles (2.576 seconds) and the lucy_small mesh with 2,804 triangles (2.597 seconds). This counter-intuitive result is puzzling, as one would expect that more triangles would require more rendering time. This phenomenon is likely related to the internal geometric structure of the models.

5 References

The implementation is based on the following references:

1. Pharr, M., Jakob, W., & Humphreys, G. (2018). *Physically Based Rendering: From Theory to Implementation* (3rd ed.). Bounding Volume Hierarchies. Retrieved from `https://pbr-book.org/3ed-2018/Primitives_and_Intersection_Acceleration/Bounding_Volume_Hierarchies#BVHAccel`

2. pbrt-v3: Physically Based Rendering. GitHub repository. Retrieved from `https://github.com/mmp/pbrt-v3`