HCCMeshes: Hierarchical-Culling oriented Compact Meshes

Tae-Joon Kim[1], Yongyoung Byun[1], Yongjin Kim[2], Bochang Moon[1], Seungyong Lee[2], and Sung-Eui Yoon[1]
[1] KAIST [2] POSTECH

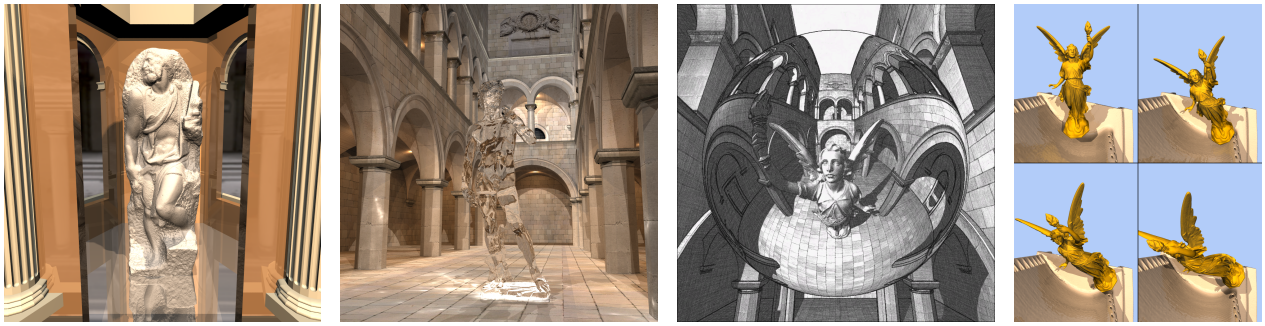

Figure 1: **Applications:** These figures show images of applications using our HCCMesh representations. From left, we show a Whitted-style ray tracing of the St. Matthew, photon mapping on a transparent David model in the Sponza scene, a line-art style rendering of the Lucy model reflected on a sphere, and collision detection between the Lucy and a CAD turbine model

1 Introduction

Ray tracing and collision detection are widely used for providing high-quality visualizations and user interactions. In these algorithms, we need to detect intersecting primitives between two input objects (e.g., a ray and a 3D object in ray tracing and two 3D objects in collision detection). In order to efficiently detect these intersecting primitives, hierarchical traversal and culling by using bounding volume hierarchies (BVHs) are commonly used.

Due to advances of model acquisition and computer-aided design techniques, massive models are easily generated these days. Such massive models can consist of hundreds of millions of triangles and thus use several gigabytes of memory. In addition, BVHs constructed from these massive models can use additional gigabytes of memory space. Although BVHs are intended to accelerate the performance of applications, the additional memory requirement of using BVHs can increase the working set size during the hierarchical traversal and can increase the data fetching time from the disk, which could negate the benefits of using BVHs. This high memory requirement of a BVH is likely to cause more serious performance issues in the coming years, given the well-known widening gap between the computational speed and the data access speed on current commodity hardware.

Only a few techniques have been proposed to design compact mesh and BVH representations in order to reduce the data access time and memory requirements during the hierarchical traversal. None of them supports various tree structures of BVHs, while providing efficient hierarchical culling and a low runtime access overhead. Furthermore, these prior techniques do not provide enough compression ratios to handle large-scale models consisting of hundreds of millions of triangles on commodity hardware [Kim et al. 2010].

2 Our Approach

To compute the HCCMesh representation, we first construct a BVH from a mesh and then decompose the BVH into a single high-level BVH and multiple low-level BVHs. In order to reduce the data fetching time from external drives and to lower the memory requirement of meshes and BVHs, we compute our HCCMesh representation for each low-level BVH. Our HCCMesh representation has in-core and out-of-core parts. The in-core HCCMesh representation, i-HCCMesh, tightly integrates the mesh and BVH representations. We compress i-HCCMeshes further to reduce the expensive data access time from external drives for applications that run in an out-of-core mode.

When a low-level BVH is requested at runtime, we fetch its corresponding o-HCCMesh from an external drive, decompress it, and store it in main memory as our i-HCCMesh representation which efficiently supports the random hierarchical traversal. In order to enable a high overall performance improvement, our methods support high compression ratios and fast decompression performance. If all the i-HCCMeshes fit into main memory, the o-HCCMeshes are sequentially fetched and decompressed into the i-HCCMeshes as the application begins. When all the o-HCCMeshes, but not all the i-HCCMeshes, fit into main memory, we load the o-HCCMeshes to main memory without any decompression and then decompress them into the i-HCCMeshes when necessary, in order to remove the expensive disk I/O access at runtime. Otherwise, the o-HCCMeshes are fetched on demand from the disk and decompressed into i-HCCMeshes while using the LRU-based memory management.

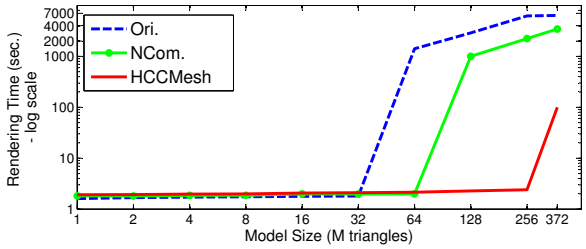

Figure 2: **Ray Tracing Time vs. Model Complexity:** This graph shows the rendering time with various model complexities of the St. Matthew model shown in Fig. 1. We measure the performance of ray tracing with our HCCMesh, the original (**Ori.**), and naively compressed (**NCom.**) representations.

3 Advantages of Our Approach

1. **Low memory requirement:** Our i-HCCMesh and o-HCCMesh has 3.6:1 and 10.4:1 compression ratios on average over a naively quantized representation. This low memory requirement reduces the data access time and the size of the working set during the hierarchical traversal.
2. **High performance improvement:** We test our method on ray tracing, photon mapping, non-photorealistic rendering, and collision detection (Fig. 1) and compare our method over the naively quantized representation. We can handle models ten times larger in these applications without the expensive disk I/O thrashing by using our representation (Fig. 2). In the case when we can avoid the disk I/O thrashing, we can improve the performance by up to two orders of magnitude.

References

KIM, T.-J., MOON, B., KIM, D., AND YOON, S.-E. 2010. RACBVHs: Random-accessible compressed bounding volume hierarchies. *IEEE Trans. on Visualization and Computer Graphics 16*, 2, 273–286.

SIGGRAPH 2010, Los Angeles, California, July 25 – 29, 2010.
ISBN 978-1-4503-0210-4/10/0007