High Performance Graphics (2012)
C. Dachsbacher, J. Munkberg, and J. Pantaleoni (Editors)

Maximizing Parallelism in the Construction of BVHs, Octrees, and k-d Trees

Tero Karras

NVIDIA Research

Abstract

A number of methods for constructing bounding volume hierarchies and point-based octrees on the GPU are based on the idea of ordering primitives along a space-filling curve. A major shortcoming with these methods is that they construct levels of the tree sequentially, which limits the amount of parallelism that they can achieve. We present a novel approach that improves scalability by constructing the entire tree in parallel. Our main contribution is an in-place algorithm for constructing binary radix trees, which we use as a building block for other types of trees.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types

1. Introduction

In the recent years, general-purpose GPU computing has given rise to a number of methods for constructing bounding volume hierarchies (BVHs), octrees, and k-d trees for millions of primitives in real-time. Some methods aim to maximize the quality of the resulting tree using the surface area heuristic [DPS10], while others choose to trade tree quality for increased construction speed [LGS*09, PL10, GPM11].

The right quality vs. speed tradeoff depends heavily on the application. Tree quality is usually preferable in ray tracing [AL09] where the same acceleration structure is often reused for millions of rays. Broad-phase collision detection [Eri04] and particle interaction [YB11] in real-time physics represent the other extreme, where construction speed is of primary importance—the acceleration structure has to be reconstructed on every time step, and the number of queries is usually fairly small. Furthermore, certain applications, such as voxel-based global illumination [CNS*11] and surface reconstruction [ZGHG11], specifically rely on regular octrees and k-d trees, where tree quality is fixed.

The main shortcoming with existing methods that aim to maximize construction speed [GPM11, ZGHG11] is that they generate the node hierarchy in a sequential fashion, usually one level at a time. This limits the amount of parallelism that they can achieve at the top levels of the tree, and can lead to serious underutilization of the parallel cores. The sequential processing is already a bottleneck with small workloads on current GPUs, which require tens of thousands of independent parallel threads to fully utilize their computing power. The problem can be expected to become even more significant in the future as the number of parallel cores keeps increasing. Another implication of sequential processing is that the existing methods output the hierarchy in a breadth-first order, even though a depth-first order would usually be preferable considering data locality and cache hit rates.

In this paper, we introduce a fast method for constructing BVHs, octrees, and k-d trees so that the overall performance scales linearly with the number of available cores (Figure 1) and the resulting data structure is always in a strict depth-first order. We start by presenting a novel in-place algorithm for constructing binary radix trees in a fully data-parallel fashion, and then show how the algorithm can be used as a building block for efficiently constructing other types of trees.

2. Background

Lauterbach et al. [LGS*09] were the first to present a parallel method for constructing so-called *linear BVHs* by ordering the input primitives along a space-filling curve. The idea is to assign a Morton code for each primitive, sort the Morton codes, and generate a node hierarchy where each subtree corresponds to a linear range of sorted primitives. The sorting effectively groups the primitives so that ones close to each other in 3D end up close to each other in the resulting tree.

The Morton code for a given point contained within the 3D unit cube is defined by the bit string $X_0Y_0Z_0X_1Y_1Z_1\ldots$

DOI: 10.2312/EGGH/HPG12/033-037