Binned SAH Kd-Tree Construction on a GPU

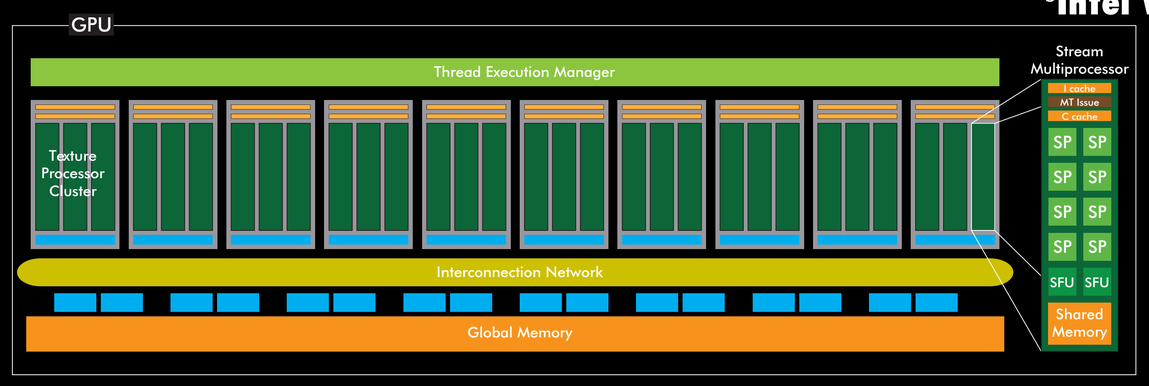
Piotr Danilewski[1], Stefan Popov[1], Philipp Slusallek[1,2,3]

UNIVERSITY OF SAARLAND
[1]Chair of Computer Graphics
[2]DFKI
[3]Intel Visual Computing Institute

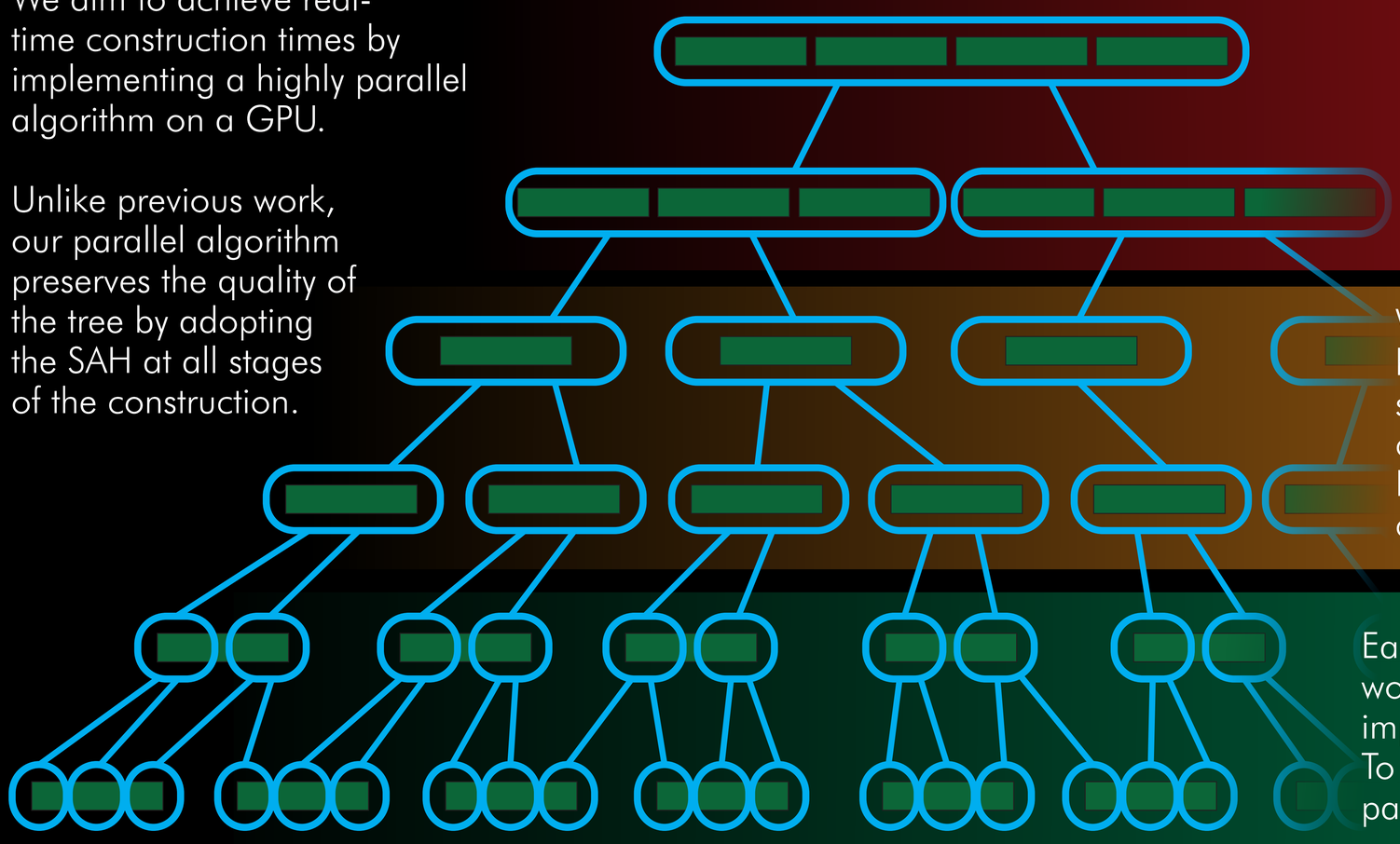
Introduction

In ray tracing, kd-trees are often regarded as the best acceleration structures, yet are too slow to build to be used for dynamic scenes.
We aim to achieve real-time construction times by implementing a highly parallel algorithm on a GPU.

Unlike previous work, our parallel algorithm preserves the quality of the tree by adopting the SAH at all stages of the construction.



Program stages



We are using approximative binned SAH. To count the number of primitives in each bin, every thread block works independently on a portion of a node, and only one synchronisation at the end is needed to sum the partial results. Relocation of triangles to the children is synchronised between blocks using atomic operations.

We are still using binned SAH, with each thread block handling a node exclusively. Binning is performed in shared memory using atomic operations, but multiple counters are used to avoid conflicts from the same warp. Destination addresses for relocation of triangles are computed using parallel prefix sum.

Each block handles several nodes at a time, each thread working on one of the triangles. We use a simple parallel implementation of an exact SAH algorithm.
To compute destination addresses for triangle relocation, a parallel segmented sum algorithm is adopted.

Results

As number of independent blocks running in parallel on a GPU is likely to increase in the future, parallel efficiency with this respect is an important measure of an algorithm.

Efficiency is defined as a weighted ratio of run time on one multiprocessor and x multiprocessors. Our implementation achieves unprecedented scalability both on GPU and CPU.

$$Eff_x = \frac{t_1}{t_x \cdot x}$$

Traverse and intersection cost parameters balance the tree construction and its quality. In this setting we allow only minimal performance drop of the RTFact raytracer compared to a full-quality SAH kd-tree.

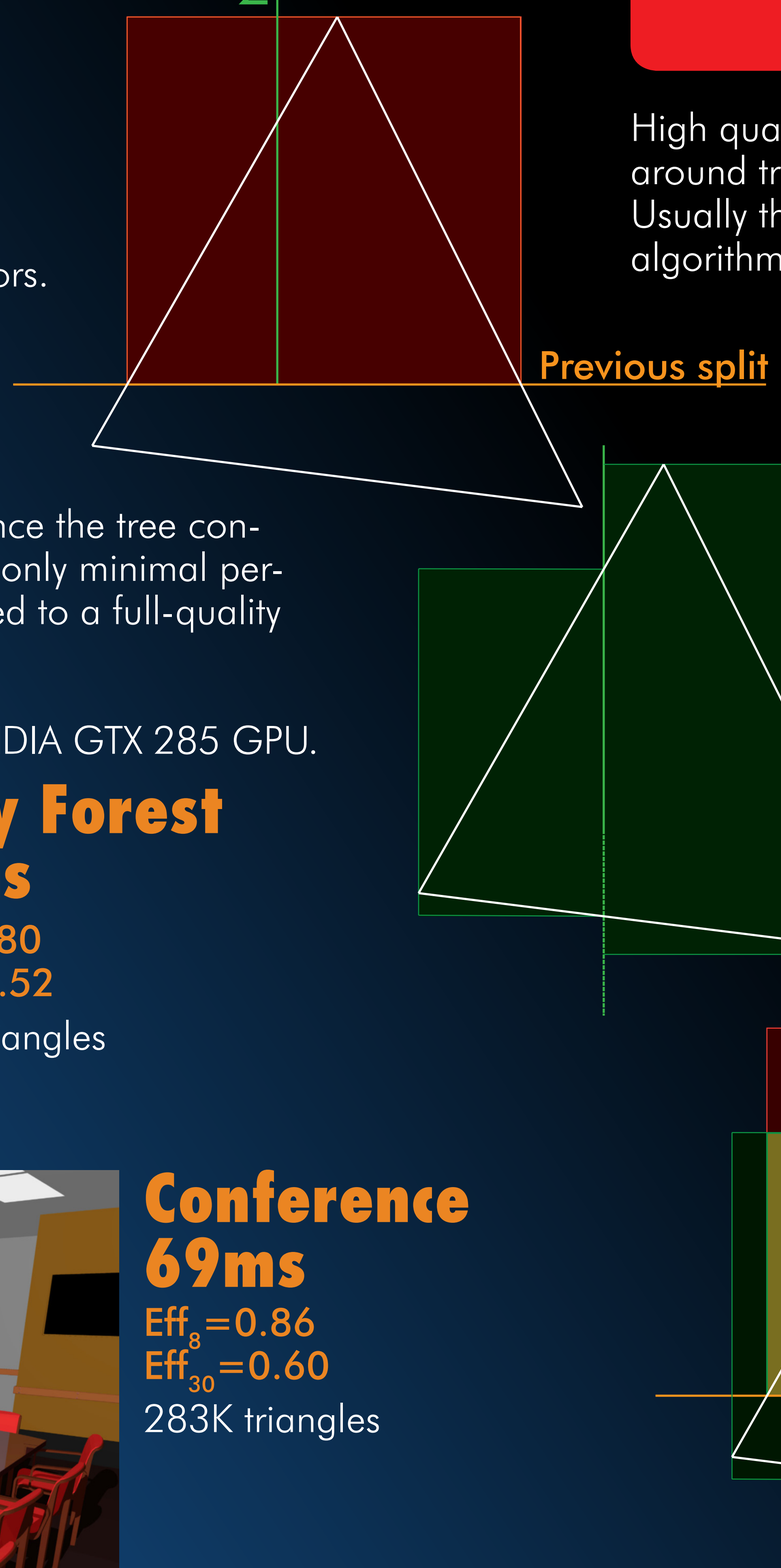
The construction times were mesured using NVIDIA GTX 285 GPU.

Fairy Forest
38ms
$Eff_8 = 0.80$
$Eff_{30} = 0.52$
174K triangles

Conference
69ms
$Eff_8 = 0.86$
$Eff_{30} = 0.60$
283K triangles

Triangle intersection

High quality kd-tree builders need to tighten the bounding boxes around triangles which are intersected by the split plane.
Usually there are only few triangles to be split but precise clipping algorithms may noticeably slow down the whole construction.



In our approach, we initially ignore previous splits, significantly simplifying our task. Instead, we intersect whole triangle with the plane and compute the resulting bounding boxes.

Only later we intersect the boxes with the previous one. In some cases the resulting boxes may be bigger than necessary but such situation is rare and does not degrade the kd-tree quality much.

contact: danilewski@cs.uni-saarland.de

I. Georgiev, P. Slusallek: RTFact: Generic Concepts for Flexible and High Performance Ray Tracing, Eurographics Symposium on Interactive Ray Tracing, 2008
B. Choi, R. Komuravelli, V. Lu, H. Sung, R. Bocchino, S. Adve, J. Hart: Parallel SAH k-D Tree Construction, High Performance Graphics 2010
Q. Hou, X. Sun, K. Zhou, C. Lauterbach, D. Manocha, B. Guo: Memory-Scalable GPU Spatial Hierarchy Construction, 2009
S. Popov, J. Gunther, H.-P. Seidel, P. Slusallek: Experiences with streaming construction of SAH KD-trees, Symposium on Interactive Ray Tracing, 2006

This work was supported by International Max Planck Research School for Computer Science and Saarland University