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

Tero Karras

NVIDIA Research
Trees
Trees

Path tracing

Pharr & Humphreys

Real-time ray tracing

NVIDIA

Better

Faster

Collision detection

Ageia

Particle simulation

NVIDIA

Photon mapping

Uchida

Surface reconstruction

Amenta et al.

Voxel-based global illumination

Crassin et al.
Fastest existing methods are sequential
- Parallelize within each hierarchy level
- But not between levels
Fastest existing methods are sequential
  - Parallelize within each hierarchy level
  - But not between levels

Lack of parallelism
  - Small workloads bottlenecked by top levels
  - Sub-linear scaling of performance
Outline

- Novel way to build the entire tree in parallel
  - Two algorithmic “building blocks”
  - Fast, scalable
Outline

- Novel way to build the entire tree in parallel
  - Two algorithmic “building blocks”
  - Fast, scalable

- Main focus: BVHs
  - Point-based octrees and k-d trees also covered in the paper
Bounding volume hierarchy
Bounding volume hierarchy
1. Assign Morton codes
2. Sort primitives
3. Generate hierarchy
4. Fit bounding boxes

\[ p_x = 0.1010 \]
\[ p_y = 0.0111 \]
\[ p_z = 0.1100 \]
LBVH - Lauterbach et al. [2009]

1. Assign Morton codes
2. Sort primitives
3. Generate hierarchy
4. Fit bounding boxes

\[ p_x = 0.101100 \]
\[ p_y = 0.010111 \]
\[ p_z = 0.110100 \]
1. Assign Morton codes
2. Sort primitives
3. Generate hierarchy
4. Fit bounding boxes

$p_x = 0.101011110010$
$p_y = 101011110010$
$p_z = 0.11100$
LBVH - Lauterbach et al. [2009]

1. Assign Morton codes
2. Sort primitives
3. Generate hierarchy
4. Fit bounding boxes
LBVH - Lauterbach et al. [2009]

1. Assign Morton codes
2. Sort primitives
3. Generate hierarchy
4. Fit bounding boxes
Binary radix tree
Binary radix tree
Binary radix tree
Binary radix tree
Binary radix tree
Longest common prefix
Longest common prefix

δ(5,6) = 4

δ(5,6) = 4
Longest common prefix

\[ \delta(0,3) = 2 \]

\[ \delta(5,6) = 4 \]
Garanzha et al. [2011]
Our method
Our method

- Define a numbering scheme for the nodes
  - Gain some knowledge of their identity
  - Establish a connection with the keys
Our method

- Define a numbering scheme for the nodes
  - Gain some knowledge of their identity
  - Establish a connection with the keys

- Find the children of a given node
  - Only look at node index and nearby keys
Our method

- Define a numbering scheme for the nodes
  - Gain some knowledge of their identity
  - Establish a connection with the keys

- Find the children of a given node
  - Only look at node index and nearby keys

- Do this for all nodes in parallel
Numbering scheme
Numbering scheme
Numbering scheme
Algorithm
Algorithm

\[\delta(2,3) = 4\]  \(\checkmark\)  \[\delta(2,3) = 4\]  \[\delta(3,4) = 0\]
Algorithm

\[ \delta(0,3) = 4 \quad \checkmark \quad \delta(3,4) = 0 \]
$\delta(0,3) = 2$

$\delta(2,3) = 4$
Algorithm

\[ \delta(0,3) = 2 \]
Algorithm

For each node i=0..n-2 in parallel:

1. Determine direction of the range
2. Expand the range as far as possible
3. Find where to split the range
4. Identify children

$O(n \log h)$
Duplicate keys

- The algorithm only works with unique keys
  - Duplicates are common in practice
Duplicate keys

- The algorithm only works with unique keys
  - Duplicates are common in practice

- Trick: Augment each key with its index
  - Distinguishes between duplicates
  - Keys are still in lexicographical order
Duplicate keys

- The algorithm only works with unique keys
  - Duplicates are common in practice

- Trick: Augment each key with its index
  - Distinguishes between duplicates
  - Keys are still in lexicographical order

- Tie-break when evaluating $\delta(i,j)$
LBVH

1. Assign Morton codes
2. Sort primitives
3. Generate hierarchy
4. Fit bounding boxes
Lauterbach et al. [2009]
Our method

- Need a different approach
  - How many levels are there?
  - Which nodes are located on a given level?
Our method

- Need a different approach
  - How many levels are there?
  - Which nodes are located on a given level?

- Traverse paths in the tree in parallel
  - Start from leaves, advance toward the root
  - Terminate threads using per-node atomic flags
Our method
Results

- Evaluate performance on GTX 480 (Fermi)
  - CUDA, 30-bit Morton codes
Results

- Evaluate performance on GTX 480 (Fermi)
  - CUDA, 30-bit Morton codes

- Compare against Garanzha et al. [2011]
  - Identical tree (top-level SAH splits disabled)
Results

- Evaluate performance on GTX 480 (Fermi)
  - CUDA, 30-bit Morton codes

- Compare against Garanzha et al. [2011]
  - Identical tree (top-level SAH splits disabled)

- Simulate large GPUs
  - N times as many cores
  - N times the memory bandwidth
Results

Fairy Forest
174K triangles

milliseconds

2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0

Morton
Sort
Build
AABB

Our method

Morton
Sort
Build
AABB

Garanzha et al.

1 × cores
2 ×
4 ×
Our method

Garanzha et al.

Fairy Forest
174K triangles

Results

millisecons

2

1.8

1.6

1.4

1.2

1

0.8

0.6

0.4

0.2

0

Morton
Sort
Build
AABB

Morton
Sort
Build
AABB

1.7 ×
1.1 ×
1.3 ×
2.4 ×
12.5 ×
33.6 ×
Results

Turbine Blade
1.77M triangles

milliseonds

<table>
<thead>
<tr>
<th>Method</th>
<th>Morton</th>
<th>Sort</th>
<th>Build</th>
<th>AABB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Garanzha et al.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.7 ×

7.0 ×

0.8 ×

1.0 ×
Results

Stanford Dragon
871K triangles

Our method
Garanzha et al.
Acknowledgements

- Timo Aila
- Samuli Laine
- David Luebke
- Jacopo Pantaleoni
- Jaakko Lehtinen

For helpful suggestions and proofreading.
Thank You

- Questions