Object Partitioning Considered Harmful: Space Subdivision for BVHs

Stefan Popov, Iliyan Georgiev, Rossen Dimov, and Philipp Slusallek
Motivation

- Classical BVH construction is not perfect
  - Looks only at primitive’s centroids

- How much more performance is there?
Background

- **SAH:** \(\text{cost}(N) = C_T + \frac{SA(N_L)|N_L| + SA(N_R)|N_R|}{SA(N)}\)

- **Cost based BVH construction: Top-down**
  - Partition set of \(N\)'s primitives into \(N_L\) and \(N_R\)
  - Take partition with minimal cost

  ![Diagram illustrating partitioning](attachment:diagram.png)

  - Exhaustive search: \(O(2^N)\)
Classical BVH Construction

- Assumes finely tessellated geometry
  - Primitive → point
Can We Do Better?

- CBVH split
  - Cost ≈ 700

- Optimal partition
  - Cost ≈ 100
Geometric Partitioning

- Regular approach: Partition $N$’s primitives
  - Evaluate $AABB$s, and use to compute cost
  - $O(2^N)$ partitions to test

- Geometric partitioning:
  - Fix child $AABB$s and put primitives according to SAH
  - Some configurations are infeasible
  - Child $AABB$ boundaries $\equiv$ boundaries of primitives
  - $O(N^{12})$ configurations to test
Geometric Partitioning Example

- Boundaries of $N_L$ or $N_R$ incident with dotted lines
- $P_4$ shared $\rightarrow$ put into node with smaller SA
Geometric Partitioning Example

- Configuration infeasible
  - $P_2$ is not covered
Practical Considerations

- \( O(N^{12}) \) is actually \( O(N^6) \)
  - Each side of the parent \( AABB \) is inherited by a child
- Select child \( AABBs \) on a regular grid
  - Run-time: \( O(G^6N^{0.5}) \) including cost calculation
    - Choosing \( G = RN^{1/6} \) yields \( O(N^{1.5}) \)
  - Look at CBVH configurations as well
Results: FPS Random Rays

- Bunny: 0.7 0.7
- Fairy Forest: 0.7 0.7
- Sponza: 0.4 0.3 0.4
- Venice: 0.5
- Soda Hall: 0.5

Higher is better

Classical BVH | Our Method (R=64) | Our Method (R=4K)
Results: Surface Area Cost

Classical BVH  |  Our Method (R=64)  |  Our Method (R=4K)

Bunny: 66  |  58  |  142  |  166
Fairy Forest: 68  |  68  |  167  |  167
Sponza: 80  |  150  |  108  |  167
Venice: 95  |  108  |  166
Soda Hall: 167  |  167

Lower is better
Result Analysis

- Suspect: SAH
  - Overlap + locally minimizing SAH has adverse effect
- Experiment: Use recursive cost evaluation
  - Tree cost better than CBVH but slower FPS!
- Hypothesis: SA model needs space partitioning
  - Intuition: Early ray termination
- New algorithm
  - Penalize overlap in cost function
  - Refine search space by allowing primitive splitting
Splitting Primitives

- Feasible and infeasible configurations

- Two possible ways to split a primitive

- SAH cost is the same
Search Spaces

- Child AABBs ∈ continuum inside parent’s AABB
  - Not limited to boundary of primitives anymore
- Limit search to a grid for practical purposes
- Augment with search space of other algorithms
  - CBVH & KD-tree construction search spaces
Penalizing Overlap

- Bias SAH to account for overlap

\[
\text{cost}(N) = C_T + \left(1 + C_O \frac{V(N_L \cap N_R)}{V(N)}\right) \left(\frac{SA(N_L)|N_L|}{SA(N)} + \frac{SA(N_R)|N_R|}{SA(N)}\right)
\]

- \(C_O\) – the overlap penalty
  - Standard SAH: \(C_O = 0\)
  - Standard SAH with space partitioning: \(C_O \to \infty\)
The Generic Algorithm

- **Parameters:**
  - Search space
  - Overlap penalty

- **Algorithm**
  - Take configuration $\in$ search space with lowest cost

- **Interesting parameters**
  - CBVH: BVH, $C_O = 0$
  - Full: Grid + KD tree + BVH, $C_O \to \infty$
  - KDBVH: KD tree, $C_O$ irrelevant
Results: FPS Random Rays

CBVH  | Full Search | KDBVH

Bunny  | 0.7  | 0.8  | 0.8
Fairy Forest  | 1.1  | 1.2  | 1.2
Sponza  | 0.4  | 0.8  | 0.8
Venice  | 1.1  | 1.2  | 1.2
Soda Hall  | 0.8  | 1.2  | 1.2

Higher is better
Results: FPS Frustum Traversal

<table>
<thead>
<tr>
<th>Scene</th>
<th>CBVH</th>
<th>Full Search</th>
<th>KDBVH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bunny</td>
<td>9.3</td>
<td>5.1</td>
<td>5.1</td>
</tr>
<tr>
<td>Fairy Forest</td>
<td>5.1</td>
<td>4.8</td>
<td>5.9</td>
</tr>
<tr>
<td>Sponza</td>
<td>5.8</td>
<td>3.3</td>
<td>3.7</td>
</tr>
<tr>
<td>Venice</td>
<td>8.7</td>
<td>5.4</td>
<td>8.7</td>
</tr>
<tr>
<td>Soda Hall</td>
<td>9.3</td>
<td>5.1</td>
<td>5.1</td>
</tr>
</tbody>
</table>

Higher is better
Comparison to Pre-Splitting

Higher is better

<table>
<thead>
<tr>
<th>Method</th>
<th>Sponza Rotated</th>
<th>Sponza</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBVH</td>
<td>0.11</td>
<td>0.57</td>
</tr>
<tr>
<td>EVH (t=14)</td>
<td>0.25</td>
<td>0.96</td>
</tr>
<tr>
<td>EVH (t=16)</td>
<td>0.28</td>
<td>0.95</td>
</tr>
<tr>
<td>ESC (e=80)</td>
<td>0.29</td>
<td>0.59</td>
</tr>
<tr>
<td>ESC (e=200)</td>
<td>0.26</td>
<td>0.55</td>
</tr>
<tr>
<td>Full Search</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>KDBVH</td>
<td>0.55</td>
<td></td>
</tr>
</tbody>
</table>
Role of Overlap Penalty

- Ray Cost
- Size in MB

Graph showing the relationship between Overlap Penalty and Ray Cost, as well as Size in MB.
Spatial Build Algorithm

- Implement KDBVH using sweep plane
- Extensions:
  - Combine with CBVH to control size using $C_O$
  - Sampling of cost function
- Issues: Might miss cost minimum
  - Cost is quadratic between split plane positions
Conclusion & Future Work

- SAH inadequate without space partitioning!
- Generic framework to study BVH construction
  - Can explore full $2^N$ search space
- Spatial build algorithm
  - Fast with near optimal results
- Research early termination aware cost function
Thank you!