Efficient Clustered BVH Update Algorithm for Highly-Dynamic Models

Kirill Garanzha

Department of Software for Computers
Bauman Moscow State Technical University, Russia
BVH summary

BVH is the tree that encloses scene objects

- Advantages:
  - Fast refitting
  - Predictable memory consumption
  - Fastest empty space passing for SAH trees

- Disadvantages:
  - Not ordered traversal of the ray in the space
  - Tree-quality degradation after refitting
  - Not very fast SAH-based build
Dynamic BVH related

- Fast BVH build assume some tree-quality degradation for ray tracing
  - BIH [Wächter EGSR06]
  - Build from hierarchy [Hunt RT07]
  - SAH binning [Wald RT07]
  - Selective restructuring [Yoon EGSR07]
Dynamic BVH related

- Fast BVH build assume some tree-quality degradation for ray tracing

- In general these approaches apart from “Selective restructuring” produce splits for every BVH-node in every frame – brute force
How to get rid of brute force?

- Lazy build
- Selective restructuring
- Avoid full reconstruction of nodes without strong dynamism under them
Idea

- To measure cheaply the dynamism under node while refitting the BVH.
- To apply the cheapest update technique for the node that saves good SAH cost:
  1. Leave it refitted
  2. Relocate BVH-cluster in proper position of the tree
  3. Rebuild the structure under node by exploiting SAH-binning, existing hierarchy, degree of dynamism.
Dynamism detection

The type of dynamism is detected in the bottom-up refitting process for every affected node:

- **Migrating node** represents a BVH-cluster of coherently moving triangles
- **Exploded node** represents a cluster of strong triangle explosions and growing overlaps
Detect migrating node

Example:

Nodes B and C are moving. They represent clusters of coherently moving triangles.
Example:

The SAH cost of the node (B+C) is improved. But nodes B and C should be united with others.
Detect migrating node

Example:

Like now…
Detect migrating node

Formula:

if $dM > dM_{\text{threshold}}$

then $N_L$ and $N_R$ represent migrating clusters and should be relocated

- $N_L$ and $N_R$ are children of $N$
- $dM$ is the predefined threshold
Detect exploded node

Example:

The overlap between \( N_L \) and \( N_R \) is likely to grow if SAH cost of \( N \) is increasing.
Detect exploded node

Example:

$N$ is worth to rebuild if there is a big % of broken nodes under it
Detect exploded node

Example:

Broken node – migrating or exploded. Broken count measures the dynamism under node...
Detect exploded node

Formula:

\[
\begin{align*}
\text{if} & \quad \begin{cases} 
\text{There is big overlap between } N_L \text{ and } N_R \\
\text{There is big } \% \text{ of broken nodes under } N 
\end{cases} \\
\text{then} & \quad \begin{cases} 
\text{The cluster with root } N \\
\text{should be restructured}
\end{cases}
\end{align*}
\]

- $N_L$ and $N_R$ are children of $N$
- $dE$ is the predefined threshold
The bottom-up refitting phase:

1. Updates BVs
2. Detects the dynamism
3. Accumulates broken counts

At the output it produces 2 arrays with migrating and exploded nodes
How to obtain a lot of smaller exploded clusters to perform independent rebuilds on them?
When exploded node $\mathcal{N}$ is detected:

- Lock it. $\mathcal{N}$ will be considered as atomic for rebuild process on a cluster above it.
- Unlock all exploded descendants of $\mathcal{N}$ that intersect the BV of the overlap between $\mathcal{N}_L$ and $\mathcal{N}_R$.
Locked exploded descendants will be rebuilt independently and will be considered as atomic for some rebuild process above them.
Independent clusters

But if some of them intersect the BV of the overlap within some ancestor $N$ then they create obstacles in the process eliminating the overlap.
Independent clusters

So they should be unlocked
Independent clusters

The set of independent clusters for rebuilding may form the hierarchy. Their roots are locked.
Hierarchical rebuild

1. New split should be created with SAH binning

2. Smaller-sized rows are considered while partitioning

3. In the next partition step the sub-row is refined if it has sufficient % of Broken count

4. Locked nodes are atomic

5. Other cluster, independent rebuild

This way saves some rebuild operations
Cluster migration

Migrating nodes are processed in 3 loops:

1. Unlink each node from the tree
2. Adjust the remaining tree and search reinsertion nodes
3. Insert each node back into the tree

At every loop all migrating nodes are passed in the “end-begin” order – i.e. a loop starts from nodes of higher levels
Cluster migration

Migrating nodes are processed in 3 loops:

1. Unlink each node from the tree
2. Adjust the remaining tree and search reinsertion nodes
3. Insert each node back into the tree

The formula that detects migrating nodes and these 3 loops automatically resolve the problem of the tree-thinning effect
Cluster migration

Where insert N when start from X?

The variant of decision is taken according to its SAH evaluation

1. Unite X and N

2. Descend into one of the children

The insertion is a recursive process of taking decisions

3. Decompose N

N is decomposed when its insertion produces severe overlap in X.
Problem: frequent updates of pointers in the proposed highly-dynamic structure result in cache-efficiency degradation.

Property of restructurings: in either rebuild or insert process a new BVH-node is allocated under some parent node.
Memory manager

Acceleration structure for pre-allocated array of BVH-nodes improves the situation:

Selector acceleration structure

BVH nodes array

It accelerates the allocation of a free BVH-node that is the nearest to the given one in the memory space (e.g. nearest to parent)
Memory manager

Acceleration structure for pre-allocated array of BVH-nodes improves the situation:

Selector acceleration structure

BVH nodes array

This strategy tries to keep the BVH-layout of reasonable cache-efficiency
Benchmarks

**BVH Updater** – Core 2 Quad 2.4 GHz
(1 core utilization)

**Ray Tracer** – GeForce 8800 GTX and CUDA, mono-ray tracer

**UNC Dynamic models:**

- Exploding dragon (252K triangles)
- Cloth simulation (92K triangles)
- Colliding balls (146K triangles)
Cloth timings

- Refit only
- Migrate only
- Render only
- Rebuild only
Balls timings

RT 08

Time, ms

Frame

Refit only
Migrate only
Render only
Total update
Rebuild only
Rebuild partitions

Relative number of rebuild partitions. Model specific 100% = number of partitions for the full-rebuild (i.e. number of inner nodes)
Independent clusters

The number of detected independent exploded clusters
Rendering performance

Relative rendering performance of the BVH in comparison to the one produced by full SAH binned-rebuild.
Cloth video
The algorithm adapts to the rigid motion associating every ball with a separate subtree and then exploits only migrating updates.
Conclusions

- The algorithm unites advantages of several updating techniques for the BVH.
- Cheapest update techniques are utilized when they can keep reasonable BVH quality.
- The algorithm is applicable to various types of dynamic models.
- Lots of produced independent clusters for rebuild will be useful in a future parallel version.