Exploiting Local Orientation Similarity for Efficient Ray Traversal of Hair and Fur

Sven Woop, Carsten Benthin, Ingo Wald, Gregory S. Johnson
Intel Corporation

Eric Tabellion
DreamWorks Animation

Legal Disclaimer and Optimization Notice

Challenges of Hair Geometry

- Path Tracing hair requires high sampling rates to reduce noise and aliasing

➔ Our approach helps by improving traversal performance

- Long and thin structures are challenging to bound using AABBs

➔ Our approach uses oriented bounding boxes to produce much tighter bounds

- Many million hairs are common (in particular for furry animals)

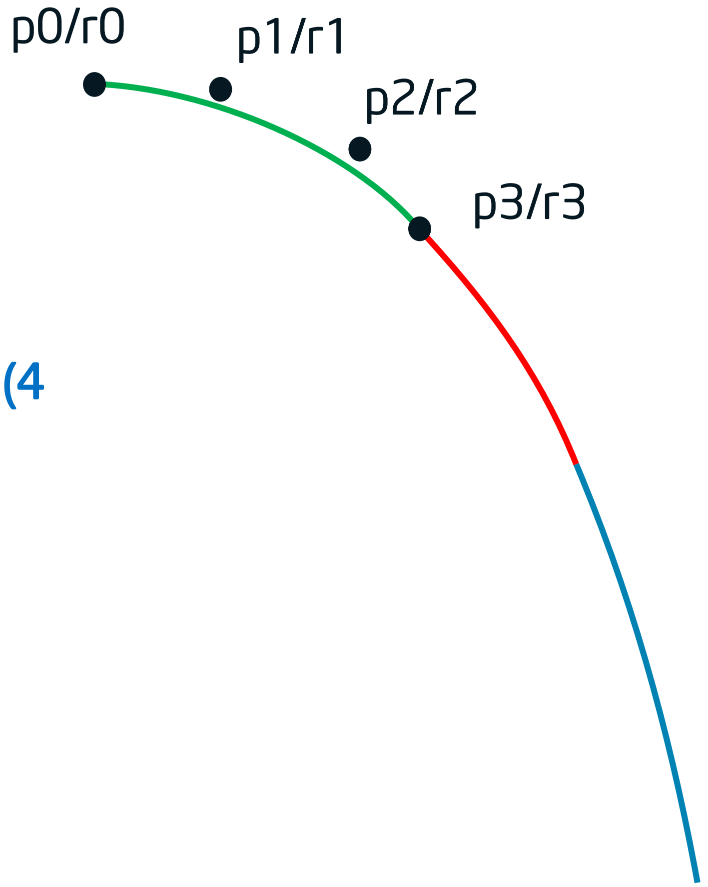
➔ We use direct ray/hair intersection to keep memory consumption low
(tesellation impractical because of high memory consumption)

Previous Work

- Path Tracing Hair
  - [Moon and Marschner 2006]: Simulating Multiple Scattering in Hair Using a Photon Mapping Approach
  - [Ou et. al. 2012]: ISHair: Importance Sampling for Hair Scattering
- Oriented Bounding Box (OBB) Hierarchies
  - [Gottschalk et. al. 1996]: OBB-Tree: A Hierarchical Structure for Rapid Interference Detection
  - [Lext and Akenine-Möller 2001]: Towards Rapid Reconstruction for Animated Ray Tracing
  - OBBs used in commercial renderers
- Ray/Curve Intersection
  - [Sederberg and Nishita 1990]: Curve Intersection using Bezier Clipping
  - [Nakamaru and Ohno 2002]: Ray Tracing for Curve Primitive

Hair Representation

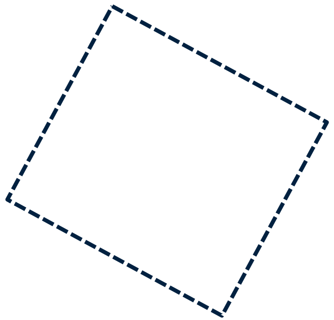
- Hair subdivided into individual hair segments (done in application)
- Hair segments represented as cubic bezier curves (4 control points) with interpolated radius (4 radii)



Bounding Representations

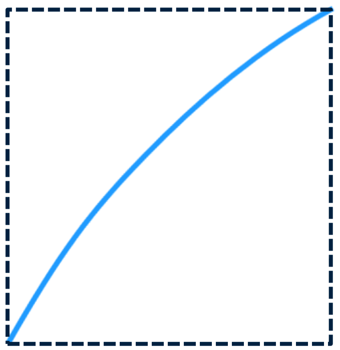
- **Axis Aligned Bounding Box (AABB):**
  lower and upper bounds in x,y,z in world space

- **Oriented Bounding Box (OBB):**
  lower and upper bounds in x,y,z in rotated space



Bounding Diagonal Hair Segment

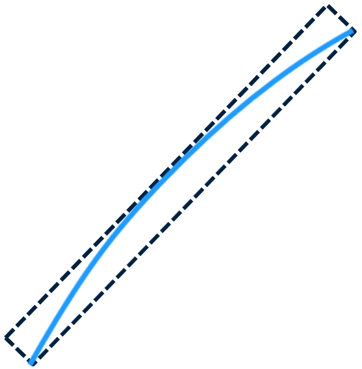
Axis aligned bounds



loose

➔ many false positives

Oriented bounds

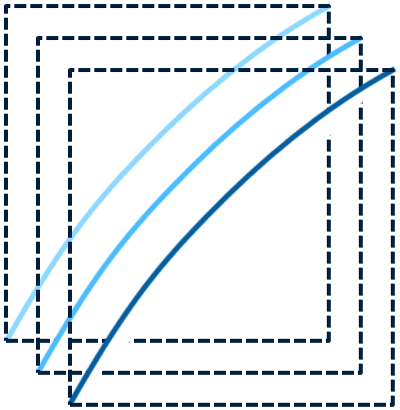


tight

➔ few false positives

Bounding Diagonal Hair Segments

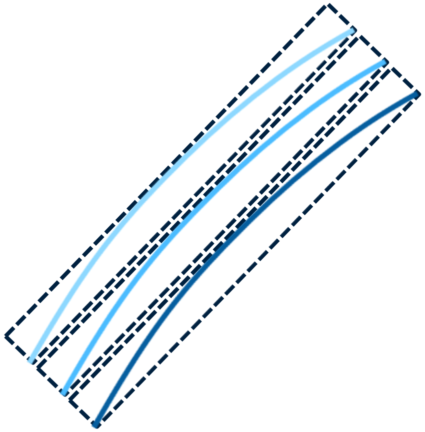
Axis aligned bounds



significant overlap

➔ many traversal steps

Oriented bounds



minimal overlap

➔ few traversal steps

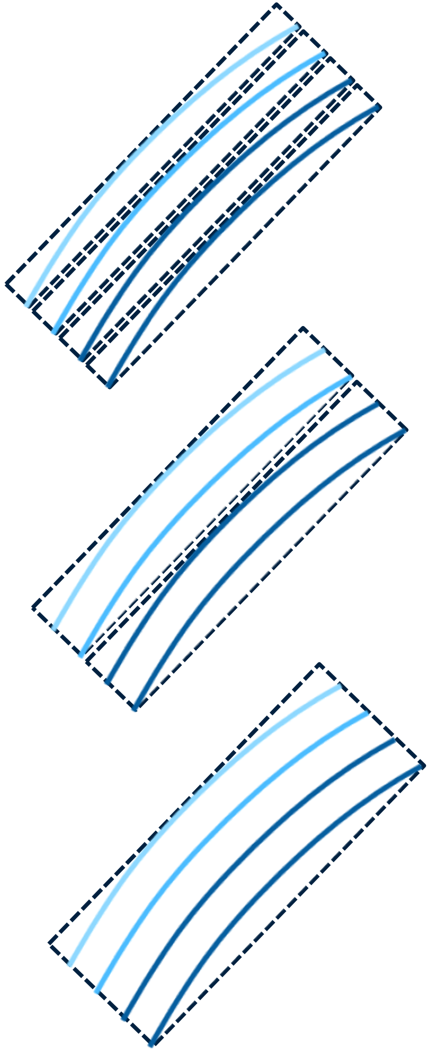
Local Orientation Similarity

- Neighboring hairs exhibit natural similarity in orientation
- For real hair, collisions cause similar orientation
- Synthetic hair mostly mimics real hair

Bounding Groups of Similarly Oriented Hairs

- Groups of equally oriented hair segments are effectively bounded by OBBs

➔ OBB hierarchy efficient for similarly oriented hair segments



Our Approach

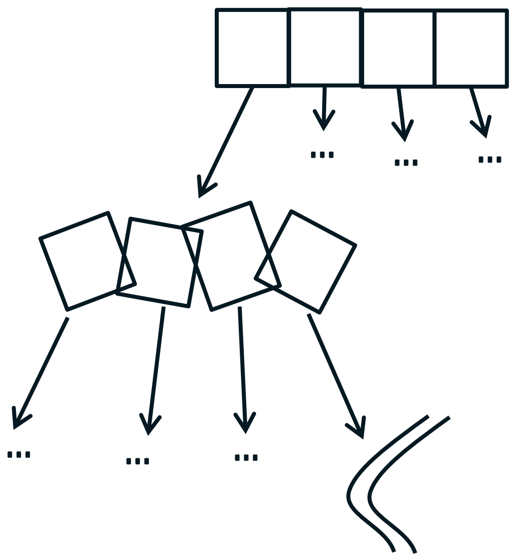
- Use mixed AABB/OBB hierarchy with fast direct ray/curve intersection
- Exploits local orientation similarity to be efficient.
- No advantage for random hair distributions.

good

no advantage

Mixed AABB/OBB Hierarchy

- 4 wide Bounding Volume Hierarchy to make effective use of 4-wide SSE
- Node types
  - **AABB nodes** store 4 AABBs plus 4 child references
  - **OBB nodes** store 4 OBBs plus 4 child references
  - **Leaf nodes** store short lists of individual cubic bezier curves
- Triangles handled in separate BVH simplifies the implementation.



AABBs versus OBBs

- OBBs bound better, but more expensive ➔ tradeoff
  - Towards the root AABBs are best as hair segments are small relative to bounding box
  - Towards the leaves OBBs are best as oriented bounds can tightly enclose hair strands

➔Few nodes store AABBs and many OBBs

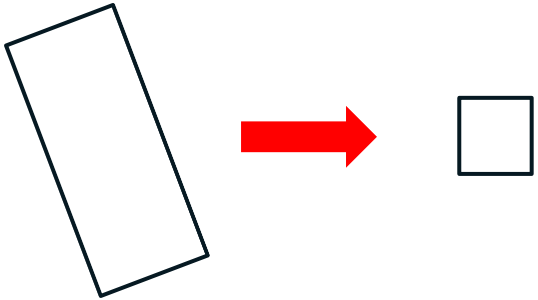
➔ Many AABB nodes and few OBB nodes get traversed

Performance	AABB only	OBB only	AABB+OBB
	100%	146%	186%

Uncompressed OBB Nodes

- Stores 4 OBBs in Struct of Array Layout for effective use of SSE
- OBB stored as affine transformation (3x4 matrices) that transforms OBB to unit AABB
- Fast ray/OBB intersection by first transforming ray and then intersecting with unit AABB
- Requires 224 bytes per node

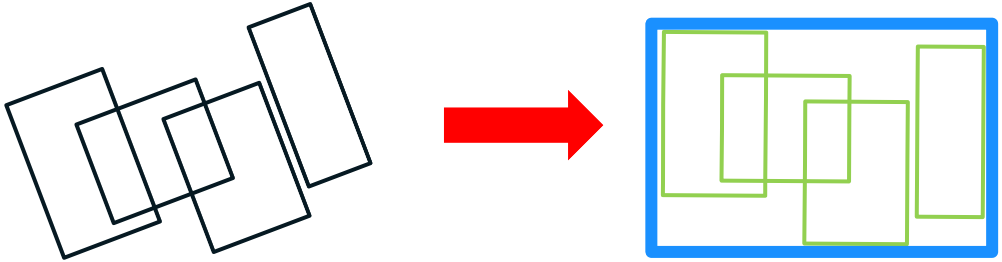
➔ about 2x the size of an AABB node



```
struct UncompressedOBBNode
{
  float[4] matrix[3][4];
  Node* children[4];
}
```

Compressed OBB Nodes

- Stores one shared quantized (signed chars) rotation that transforms the OBBs to AABBs
- Stores merged AABBs (after rotation) of all 4 children using floating point
- Stores quantisized (unsigned chars) AABBs of each child relative to merged AABB
- Requires only 96 bytes per node (less than half of uncompressed)



```
struct CompressedOBBNode
{
  char matrix[3][4];
  float min_x,min_y,min_z;
  float max_x,max_y,max_y;
  uchar cmin_x[4],cmin_y[4],cmin_z[4];
  uchar cmax_x[4],cmax_y[4],cmax_z[4];
  Node* children[4];
}
```

AABB/OBB Hierarchy Construction

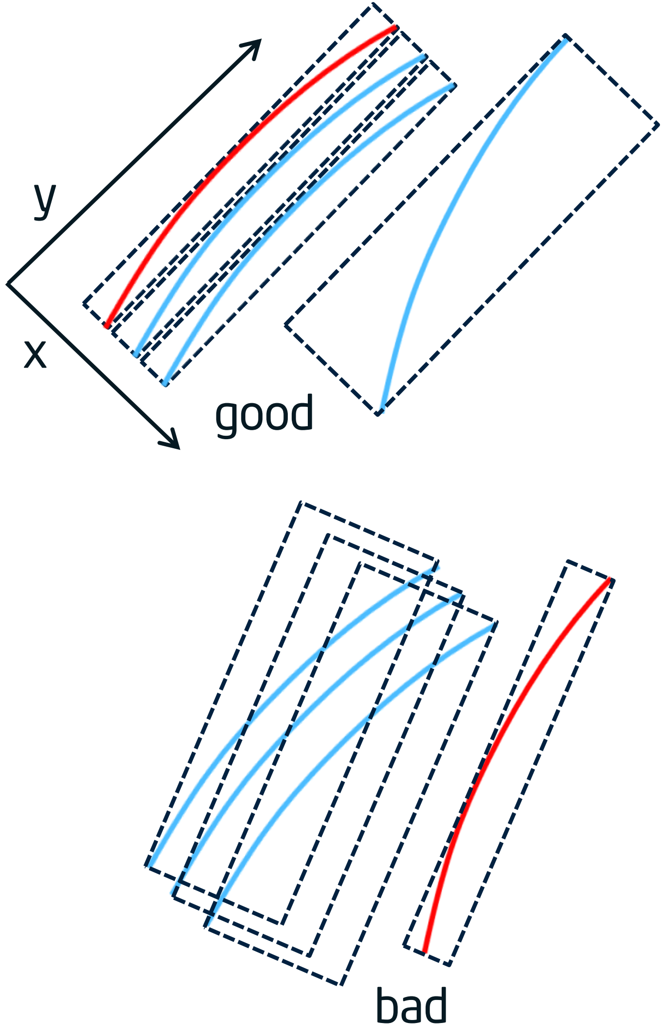
- Traditional top down build using SAH heuristic [Wald 2007]
- Handling lists of bezier curves (not lists of bounding boxes)
  ➔ control points needed for spatial splits
  ➔ control points allow to compute precise bounds in different spaces
- Use lowest SAH split from multiple splitting heuristics
- Some splitting heuristics operate in a special **hair space**
- Spatial splits [Stich et. al.; Popov et. al.] can make the approach more robust by handling challenging cases

Split Heuristics

- AABB Split Heuristics
  - Object Binning (16 bins) in world space
  - Spatial Splits (16 bins) in world space
- OBB Split Heuristics
  - Object Binning (16 bins) in hair space (most important)
  - Spatial Splits (16 bins) in hair space
  - Similar Orientation Clustering

Hair Space

- Hair space used for binning and calculating OBBs of nodes
- Hair space is a coordinate space with one axis well aligned with a set of hair curves
- Only rotations used to be area preserving
- Calculation
  - calculate candidate spaces (4 in the paper) aligned with main direction (start to end point) of random hairs
  - pick space where sum of surface areas of bounding boxes of hair is smallest

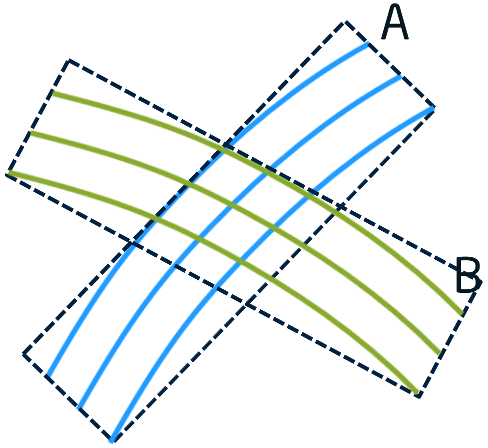


Similar Orientation Clustering

- Can separate two crossing hair strands
  ➔ No single hair space will work well

- Calculation
  - pick random hair A
  - pick hair B that is maximally misaligned with hair A
  - cluster according to main direction of hairs A and B
  - bound clusters according to space aligned with main direction of A and B

- Gives about 5% higher rendering performance



4-wide AABB/OBB Hierarchy Construction

- Split multiple times to fill up all 4 children
  (pick largest node or node with highest SAH gain)
- If only „AABB heuristic“ splits create AABB node
- If one split was an „OBB heuristic“ split create OBB node and store OBB aligned with hair space computed for each child

➔ SAH decides where to use which node type

AABB/OBB Hierarchy Traversal

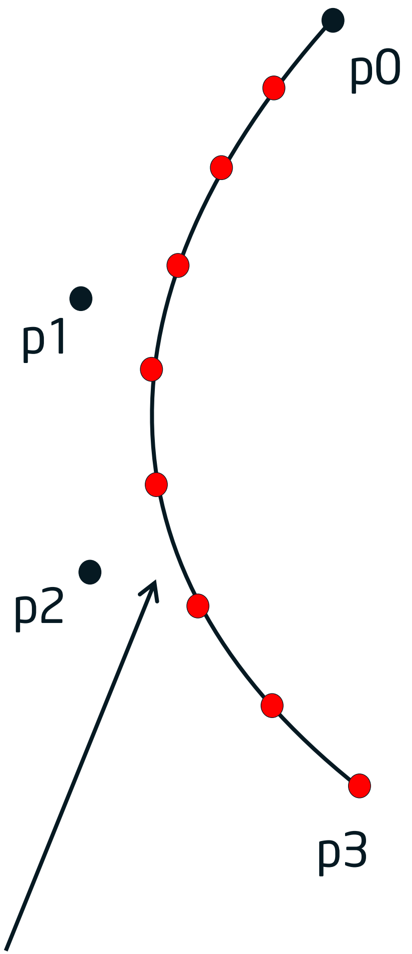
- Modified highly optimized BVH4 single ray traversal kernel of Embree
- Kept fast path for AABB node handling
- Added slow path for OBB node handling
- Added fast ray/hair segment intersection

Ray-Hair Segment Intersection

- Use 8-wide AVX to generate 8 points on curve in parallel using precalculated Bezier coefficients a,b,c,d:

  avxf p = a*p0 + b*p1 + c*p2 + d*p3

- Intersect ray using 8-wide AVX in parallel with 8 line segments using test by [Nakamaru and Ohno 2002]
- 8 segments work well for our models
  ➔ rarely very curved hair segments need pre-subdivision

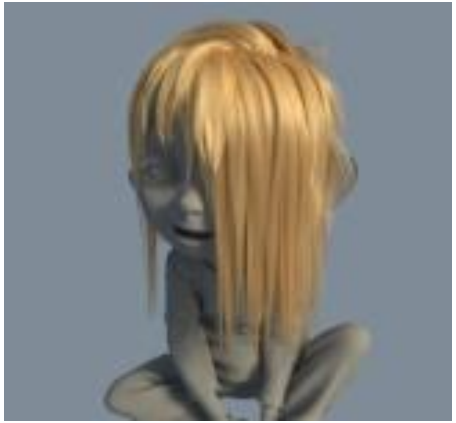


Benchmark Settings

- Dual Socket Intel® Xeon® E5-2697 (AVX2, 2x 12 cores @ 2.7 GHz, 64GB memory)
- 1M pixel resolution, path traced including shading (50% shading, 50% tracing)
- Representative movie content from Dreamworks

Tighten
420k triangles
2.2M curves

Tiger
83k triangles
6.5M curves



Sophie
75k triangles
13.3M curves

Yeti
82k triangles
153M curves

Results

		AABBs triangles	AABBs curves	AABB/OBBs curves	+ spatial splits	+ compression
	Perf.	3.5fps	3.7fps	6.6fps	7.5fps	7.3fps
	Mem.	1.1GB	257MB	387MB	633MB	404MB
	Perf.	1.44fps	1.0fps	2.1fps	2.7fps	2.5fps
	Mem.	3.5GB	0.8GB	1.1GB	1.8GB	1.1GB
	Perf.	4.2fps	3.5fps	7.1fps	7.3fps	7.1fps
	Mem.	6.8GB	1.6GB	2.1GB	3.3GB	2.7GB
	Perf.	-	1.8fps	2.6fps	3.1fps	3.2fps
	Mem.	-	18.6GB	21.7GB	34.4GB	24.9GB

Measured on Dual Socket Intel® Xeon® E5-2697, 12 cores @ 2.7 GHz

Results: Using Ray/Curve Intersector

		AABBs triangles	AABBs curves	AABB/OBBs curves	+ spatial splits	+ compression
	Perf.	3.5fps	3.7fps	6.6fps	7.5fps	7.3fps
	Mem.	1.1GB	257MB	387MB	633MB	404MB
	Perf.	1.44fps	1.0fps	2.1fps	2.7fps	2.5fps
	Mem.	3.5GB	0.8GB	1.1GB	1.8GB	1.1GB
	Perf.	4.2fps	3.5fps	7.1fps	7.3fps	7.1fps
	Mem.	6.8GB	1.6GB	2.1GB	3.3GB	2.7GB
	Perf.	-	1.8fps	[illegible]	[illegible]	3.2fps
	Mem.	-	18.6GB	[illegible]	[illegible]	24.9GB

→Using our ray/curve intersector reduces performance by 15% at 1/4th the memory consumption

Measured on Dual Socket Intel® Xeon® E5-2697, 12 cores @ 2.7 GHz

Results: Triangles Consume too much Memory

		AABBs triangles	AABBs curves	AABB/OBBs curves	+ spatial splits	+ compression
	Perf.	3.5fps	3.7fps	6.6fps	7.5fps	7.3fps
	Mem.	1.1GB	257MB	387MB	633MB	404MB
	Perf.	1.44fps	1.0fps	2.1fps	2.7fps	2.5fps
	Mem.	3.5GB	0.8GB	1.1GB	1.8GB	1.1GB
	Perf.	4.2fps			7.3fps	7.1fps
	Mem.	6.8GB			3.3GB	2.7GB
	Perf.	-			3.1fps	3.2fps
	Mem.	-	18.6GB	21.7GB	34.4GB	24.9GB

➔ Out of memory, even with 64GB of memory and tessellation into only 8 triangles.

Measured on Dual Socket Intel® Xeon® E5-2697, 12 cores @ 2.7 GHz

Results: Adding OBBs

		AABBs triangles	AABBs curves	AABB/OBBs curves	+ spatial splits	+ compression
	Perf.	3.5fps	3.7fps	6.6fps	7.5fps	7.3fps
	Mem.	1.1GB	257MB	387MB	633MB	404MB
	Perf.	1.44fps	1.0fps	2.1fps	2.7fps	2.5fps
	Mem.	3.5GB	0.8GB	1.1GB	1.8GB	1.1GB
	Perf.	4.2fps	3.5fps	7.1fps	7.3fps	7.1fps
	Mem.	6.8GB	1.6GB	2.1GB	3.3GB	2.7GB
	Perf.	-	1.8fps	2.6fps	[illegible]	[illegible]
	Mem.	-	18.6GB	21.7GB	[illegible]	[illegible]

→ adding OBBs gives 80% speedup for 30% higher memory consumption

Measured on Dual Socket Intel® Xeon® E5-2697, 12 cores @ 2.7 GHz

Results: Adding Spatial Splits

		AABBs triangles	AABBs curves	AABB/OBBs curves	+ spatial splits	+ compression
	Perf.	3.5fps	3.7fps	6.6fps	7.5fps	7.3fps
	Mem.	1.1GB	257MB	387MB	633MB	404MB
	Perf.	1.44fps	1.0fps	2.1fps	2.7fps	2.5fps
	Mem.	3.5GB	0.8GB	1.1GB	1.8GB	1.1GB
	Perf.	4.2fps	3.5fps	7.1fps	7.3fps	7.1fps
	Mem.			2.1GB	3.3GB	2.7GB
	Perf.			2.6fps	3.1fps	3.2fps
	Mem.			21.7GB	34.4GB	24.9GB

➔spatial splits give 15% speedup for 60% higher memory consumption

Measured on Dual Socket Intel® Xeon® E5-2697, 12 cores @ 2.7 GHz

Results: Adding Compression

		AABBs triangles	AABBs curves	AABB/OBBs curves	+ spatial splits	+ compression
	Perf.	3.5fps	3.7fps	6.6fps	7.5fps	7.3fps
	Mem.	1.1GB	257MB	387MB	633MB	404MB
	Perf.	1.44fps	1.0fps	2.1fps	2.7fps	2.5fps
	Mem.	3.5GB	0.8GB	1.1GB	1.8GB	1.1GB
	Perf.	4.2fps	3.5fps	7.1fps	7.3fps	7.1fps
	Mem.			2.1GB	3.3GB	2.7GB
	Perf.			2.6fps	3.1fps	3.2fps
	Mem.			21.7GB	34.4GB	24.9GB

➔ spatial splits and compression give 13% speedup for similar memory consumption

Measured on Dual Socket Intel® Xeon® E5-2697, 12 cores @ 2.7 GHz

Video

- Path tracing with up to 10 bounces @ about 1M pixels
- 2x Intel(R) Xeon(R) CPU E5-2687W @ 3.10GHz (16 cores total)

Conclusion and Future Work

- AABB/OBB hierarchy gives almost 2x speedup for hair geometry
- Need to improve build performance currently 20x slower than building standard BVH over curve segments
- Handling triangles in same BVH could give additional benefit.
- Support for Motion Blur is important for movie rendering.

Questions?

Source code for Xeon and Xeon Phi
available as part of Embree 2.3.1,
https://embree.github.com