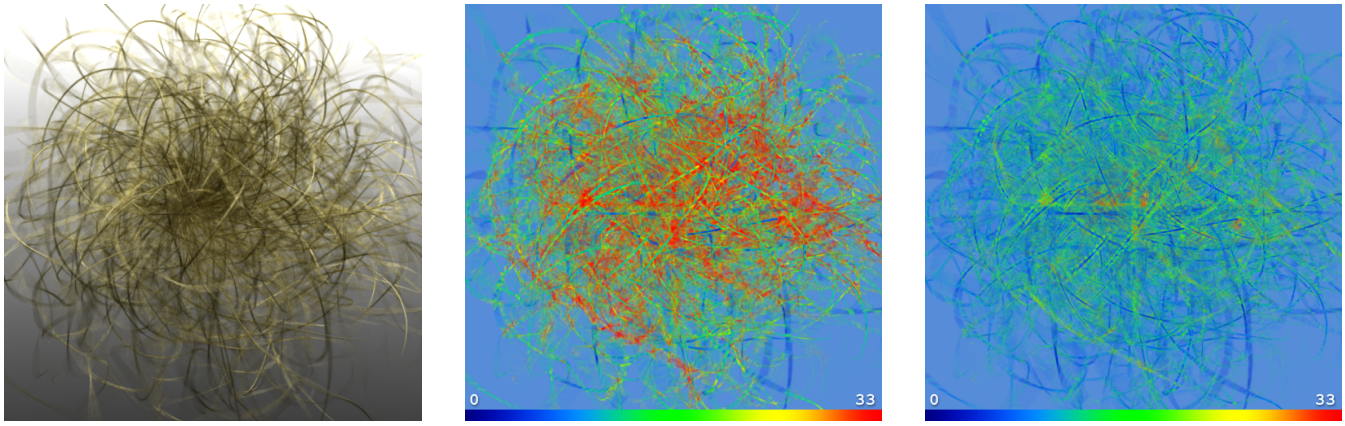
MSBVH: An Efficient Acceleration Data Structure for Ray Traced Motion Blur

Leonhard Grünschloß* NVIDIA / Weta Digital
Martin Stich† NVIDIA
Sehera Nawaz‡ NVIDIA / Weta Digital
Alexander Keller§ NVIDIA



(a) *Hairball rendered with motion blur.*

(b) *Number of intersections tests for the classic BVH with node interpolation.*

(c) *Number of intersections tests for the MSBVH (our approach).*

Figure 1: *The hairball consists of moving strands that are tessellated into many small triangles. The false color images depict the number of intersection tests for primary rays, where blue corresponds to small numbers and red to large numbers. As compared to the classic BVH with node interpolation, using an MSBVH, the number of intersection tests can be significantly reduced. This leads to an overall performance improvement.*

Abstract

When a bounding volume hierarchy is used for accelerating the intersection of rays and scene geometry, one common way to incorporate motion blur is to interpolate node bounding volumes according to the time of the ray. However, such hierarchies typically exhibit large overlap between bounding volumes, which results in an inefficient traversal. This work builds upon the concept of spatially partitioning nodes during tree construction in order to reduce overlap in the presence of moving objects. The resulting hierarchies are often significantly cheaper to traverse than those generated by classic approaches.

CR Categories: I.3.7 [Computing Methodologies]: Computer Graphics—Three-Dimensional Graphics and Realism

Keywords: Ray tracing, bounding volume hierarchies, motion blur.

1 Introduction

Motion blur occurs in photography whenever the camera or objects move while the shutter is open. The resulting blur in an image due to such motion is an important visual cue. Omitting the simulation of motion blur will exhibit strobing in a sequence of images and audiences will instantly recognize the imagery as artificial [Apodaca et al. 1995]. Thus, motion blur simulation is a crucial feature whenever rendering a movie.

The two major methods of specifying motion for rendering are transformation matrices and paths that depend on time. For example, these can be attributed to object instances or primitive vertices, respectively. Due to its simplicity, often only linear interpolation is applied as an approximation, which requires to sample both temporal transformation matrices and paths in time. The elements of the resulting linear splines are called motion segments. Hiding approximation artifacts requires a sufficient number of samples in time.

As we aim for photorealistic simulation, we consider ray traced motion blur. For this purpose, each ray is traced at a time within the camera's shutter interval. Accelerated ray tracing requires an auxiliary data structure, which must support adjusting to that time in a very efficient way. We introduce such a data structure that allows for faster ray traced motion blur as compared to previous methods.

2 Background

The two most common types of hierarchical acceleration data structures for ray tracing [Pharr and Humphreys 2010] are bounding volume hierarchies (BVH) and kd-trees, both of which can be extended to handle motion blur. [Olsson 2007] introduced 4D kd-trees with splits along the time axis in addition to the regular partitions in the spatial dimensions. The main drawback of this approach is the large amount of memory required for the hierarchies caused by the large number of object reference replications.

Most recent implementations of BVHs use axis-aligned bounding boxes as bounding volumes. BVHs are well suited for ren-

*e-mail: leonhard@gruenschloss.org
†e-mail: mstich@nvidia.com
‡e-mail: sehera.nawaz@googlemail.com
§e-mail: keller.alexander@gmail.com

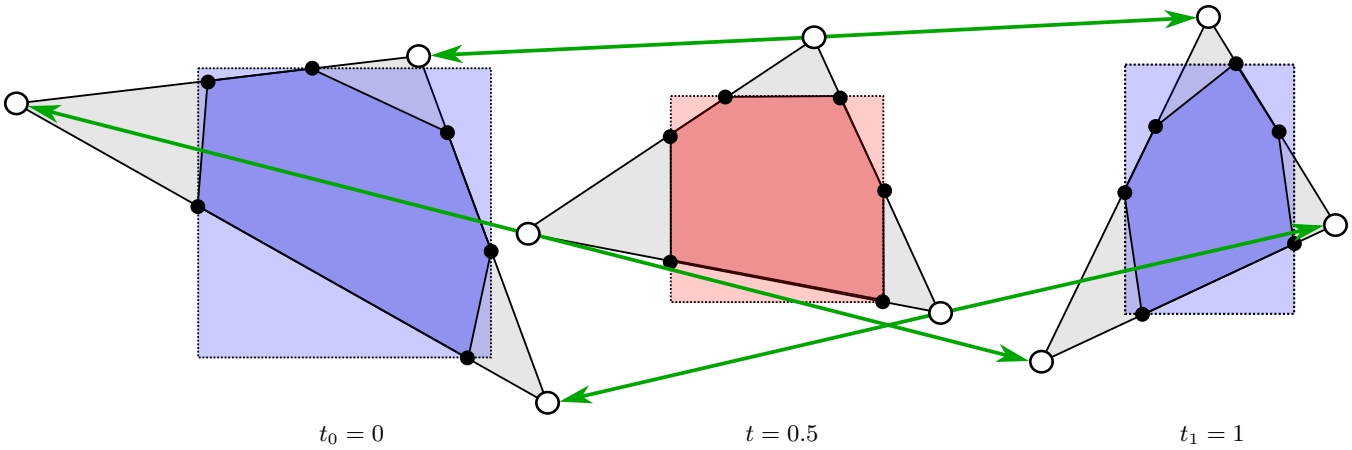


Figure 2: *Illustration for a triangle under linear motion (motion segments as green lines): At the center, the bounding box of a leaf node (red) together with a corresponding leaf triangle primitive is shown as it results from the SBVH hierarchy construction algorithm for $t = 0.5$. Due to spatial splits parts of the triangle primitive have been clipped, resulting in the solid polygon region inside the leaf bounding box. This clipped primitive is transformed to $t_0 = 0$ (shown on the left) and $t_1 = 1$ (shown on the right) and bounded to extend the leaf bounding boxes of the MSBVH (blue). The resulting leaf bounding boxes are interpolated during traversal.*

dering scenes with motion blur, since during traversal the bounding boxes of the hierarchy can be interpolated according to the ray time [Christensen et al. 2006; Hanika et al. 2010; Hou et al. 2010]. This results in much tighter bounds than intersecting with the bounding box of a moving primitive containing the complete motion path. However, depending on the scene geometry and its motion, classic interpolated BVHs are likely to suffer from overlapping bounding volumes, because each primitive is fully contained in a single leaf node. Overlap increases the total intersection cost since culling is not efficient in the affected regions.

A bounding volume hierarchy partitions the list of objects, while a kd-tree partitions space. [Stich et al. 2009] introduced the SBVH, which added spatial partitioning to BVHs at the cost of reference replication. These spatial splits significantly reduce the overlap of node bounding boxes for many scenes, which reliably increases rendering performance. Motion blur, however, has not been considered.

3 Algorithm

We introduce the concept of the Motion SBVH (MSBVH), which supports the efficient ray tracing of motion blur by interpolating node bounding boxes, while at the same time reducing node overlap using spatial splits as introduced with the SBVH.

The construction of the topology of the MSBVH is identical to the SBVH built for $t = 0.5$ in the camera shutter interval $[0, 1]$. This exploits the observation that within a single frame, using a constant topology for the hierarchy works well in practice [Christensen et al. 2006].

In order to enable temporal interpolation, each hierarchy node must store $n + 1$ volumes. Each volume corresponds to a single time instant t_i for $i = 0, \ldots, n$. Thus, for the example of linear motion from $t_0 = 0$ to $t_1 = 1$, two bounding boxes need to be stored in each node, which bound the respective endpoints of the motion segments.

Key of the MSBVH construction is the computation of the $n + 1$ bounding boxes of each node. Given the SBVH bounding boxes, which correspond to time $t = 0.5$, we start by determining the bounding boxes of the leaf nodes: each primitive referenced in an SBVH leaf is intersected with the leaf's bounding box (referred to as "clipping box" in the following). The resulting clipped primitive then is transformed for each t_i as detailed in Section 3.1 and illustrated in Figure 2. The bounding box B_i of all of the leaf's transformed clipped primitives constitutes the MSBVH leaf bounding box at time t_i. Finally, the bounds are propagated up the hierarchy.

Besides linear interpolation, the construction allows one to use any convex combination to find the bounding box $B(t) = \sum_{i=0}^{n} w_i(t) B_i$ to be intersected with a ray at time t. Note that the operations are performed componentwise and that the weights $0 \leq w_i \leq 1$ need to sum up to one.

3.1 Clipping and Transformation

We now describe in detail how the primitives are clipped and how the results are transformed in order to determine the leaf bounding boxes of the MSBVH.

3.1.1 Triangles

Triangles are clipped against the clipping box using the Sutherland-Hodgman algorithm [Sutherland and Hodgman 1974]. For the vertices of the resulting polygon, we compute the barycentric coordinates (α, β, γ) with respect to their unclipped triangle $(\mathbf{a}, \mathbf{b}, \mathbf{c})$. These are found by repeatedly solving a 2×2 linear system for each vertex $\mathbf{p}$ of the clipped triangle polygon:

$$\alpha \cdot \mathbf{a} + \beta \cdot \mathbf{b} + \gamma \cdot \mathbf{c} = \mathbf{p},$$

or equivalently, since $\alpha = 1 - \beta - \gamma$,

$$\mathbf{a} + \beta \cdot (\mathbf{c} - \mathbf{a}) + \gamma \cdot (\mathbf{b} - \mathbf{a}) = \mathbf{p},$$

which is an overconstrained 3×3 linear system that can be reduced to a 2×2 system matrix as described in [Pharr and Humphreys 2010, Ch. 10].

The three triangle vertices are transformed according to time t_i. Then the barycentric coordinates are used to compute the transformed polygon vertices, which are used to extend the axis-aligned leaf bounding box for time t_i. After extending all $n + 1$ bounding boxes, the polygon data can be discarded. This procedure tightly bounds the transformed area of the clipped and transformed triangle as illustrated in Figure 2.

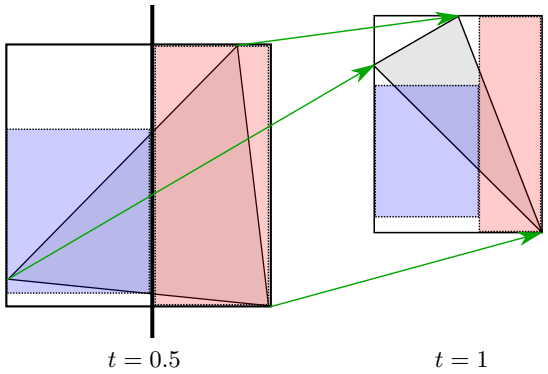


Figure 3: *On the left, two tight bounding boxes (in blue and red) are shown that result from splitting a triangle by a plane. On the right, the triangle has been moved according to its motion vector (in green). Since motion and clipping are not commutative, the transformation of the tight bounding boxes is not guaranteed to cover the transformed triangle (in gray).*

3.1.2 Axis-Aligned Bounding Boxes

Clipping may not be practical for some primitives. In such cases one can transform the bounding box of the primitive intersected with the clipping box. The resulting bounding boxes may not be as tight compared to clipping the primitive.

Given the primitive bounding box $[\mathbf{b}_{\text{min}}, \mathbf{b}_{\text{max}}]$ at $t = 0.5$, its transformed counterpart $[\mathbf{f}_{\text{min}}, \mathbf{f}_{\text{max}}]$ at t_i, and the bounding box of the primitive intersected with the clipping box $[\mathbf{c}_{\text{min}}, \mathbf{c}_{\text{max}}]$,

$$\left[\mathbf{f}_{\text{min}} + \frac{\mathbf{c}_{\text{min}} - \mathbf{b}_{\text{min}}}{\mathbf{b}_{\text{max}} - \mathbf{b}_{\text{min}}} \cdot (\mathbf{f}_{\text{max}} - \mathbf{f}_{\text{min}}),\right.$$
$$\left.\mathbf{f}_{\text{min}} + \frac{\mathbf{c}_{\text{max}} - \mathbf{b}_{\text{min}}}{\mathbf{b}_{\text{max}} - \mathbf{b}_{\text{min}}} \cdot (\mathbf{f}_{\text{max}} - \mathbf{f}_{\text{min}})\right] \quad (1)$$

is the interpolated clipped bounding box at t_i, where all vector operations are performed elementwise.

The naïve approach of transforming the bounding box of the clipped primitive is ruled out due to the fact that transformation and clipping are not commutative as illustrated in Figure 3.

3.1.3 Complex Shapes

Resorting to using only bounding boxes (as described in the previous section) is not efficient for parametric surfaces and multi-resolution surfaces like, for example, displacement mapped subdivision surfaces. In these cases, the elements resulting from parametric or regular tessellation can be bounded, as illustrated in Figure 4 similar to [Hanika et al. 2010; Munkberg et al. 2010].

The resulting bounding boxes are handled as in the previous section. Finally, the bounding box of a node is determined by the bounds of the clipped boxes of all elements in that node. Alternatively, it is possible to avoid clipping the elements' bounding boxes. However, this generally does not result in an exact spatial split anymore and would consequently lead to larger bounding boxes.

Often, the bounding boxes obtained with refinement are much tighter than without and enable a more precise approximation of the SAH.

4 Instances

Instances are a common tool to reduce scene complexity and hierarchy construction times. Instead of a tree, a directed acyclic graph

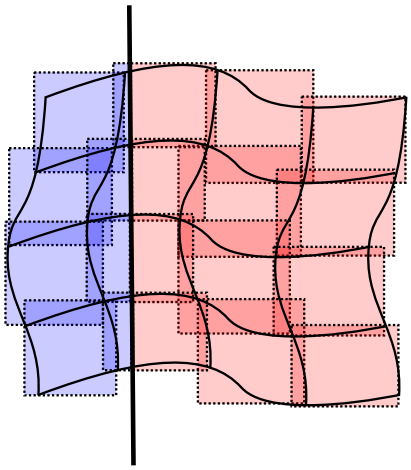


Figure 4: *Approximating a complicated patch by more than one bounding box allows to conservatively determine parts of the patch that belong to the left and right child as a consequence of a spatial split.*

is used, where parts of the hierarchy can be referenced multiple times. Such a reference contains a transformation matrix $A(t)$ that depends on the time t and is the frame of the referenced geometry. When a ray is intersected with such a node, it is transformed to the frame of the instance by applying the inverse transformation matrix.

In the context of animation, the important question is how to find a tight bounding box of an animated instance during hierarchy construction. For example, [Pharr and Humphreys 2010] sample $A(t)$ at discrete times t_i to determine a bounding box, but apply the temporally continuous mapping $A^{-1}(t)$ to transform the ray. Their sampling does not guarantee a conservative bounding box and consequently geometry can be missed during ray intersection.

We therefore introduce an approach that determines conservative bounding boxes for hierarchy construction. In order to enable interpolation, the $n + 1$ bounding boxes of an MSBVH node are determined by bounding the instance using the frame $A_i := A(t_i)$. Tighter bounding boxes are achieved by transforming the instance geometry, while transforming the bounding box of the instance is faster, but not as tight in general. The resulting instance bounding boxes are processed as described in Section 3.1.2. Note that the primitives of individual instance hierarchies are still clipped and transformed as described in Section 3.1,

Conservative ray intersection then requires to approximate the transformation: During hierarchy traversal, the bounding boxes are interpolated in exactly the same way as described in the algorithm overview. When reaching the leaf that holds the instance, the matrices are interpolated analogously to the bounding boxes: $A'(t) = \sum_{i=0}^{n} w_i(t) A_i$.

The ray is transformed using the inverse of this interpolated transformation matrix before traversing the instance-level MSBVH. Note that we are not interpolating individual factored matrices resulting from a matrix decomposition into scale, rotation, and translation components [Pharr and Humphreys 2010]. Instead, we interpolate the individual matrix elements, resulting in splines that match the interpolation of the instance bounding boxes. This guarantees that no geometry of the instance can be missed.

For the example of linear interpolation with equidistant times $t_i := \frac{i}{n}$, first the fraction $\lambda = n \cdot t - i$ inside the motion segment $i = \lfloor n \cdot t \rfloor$ is determined. Then the ray is transformed using the inverse

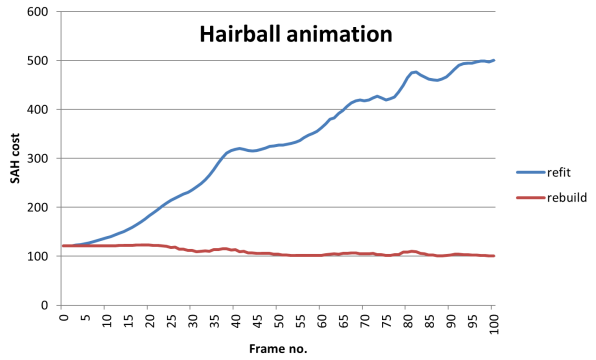


Figure 5: *Evolution of the SAH cost function applied to the MSBVH for the hairball scene from Figure 1: as expected, the cost difference between refitting and rebuilding the hierarchy each frame gradually increases over the course of the animation.*

of $A'(\lambda) := (1-\lambda)\cdot A_i + \lambda \cdot A_{i+1}$. For arbitrary t_i a binary search could be used to locate the corresponding motion segment.

Mail-boxing [Amanatides and Woo 1987] can be applied in order to avoid intersecting a ray more than once with the same instance. Alternatively, it is possible to queue a fixed number of instance intersections along the ray [Hanika et al. 2010]. This queue then is sorted to remove duplicates before performing further intersections. In this context it is recommended to account for the mail-boxing when evaluating the SAH, similar to [Hunt 2008].

While instancing can substantially reduce memory requirements and hierarchy construction time, it cannot reduce the overlap among instances. Thus, there are cases where replacing the directed acyclic graph by its corresponding tree results in superior performance.

5 Refitting

Bounding volume hierarchy construction cost can be amortized across frames by keeping the hierarchy topology, updating leaf bounding boxes, and propagating the results up the hierarchy [Wald et al. 2007]. Using the SAH cost function as a measure, refitting can result in a degradation of the tree quality, as illustrated in Figure 5. Therefore, refitting usually is complemented by a heuristic [Lauterbach et al. 2006] which triggers rebuilds for degenerated parts of the hierarchy.

In order to refit a leaf bounding box for the MSBVH, we take into account the clipped primitives. For a subsequent frame, we re-clip each primitive against the corresponding leaf bounding box of the hierarchy at the previous frame's shutter closing time. This way, no additional geometry or hierarchy data needs to be stored. Due to the re-clipping of primitives, the MSBVH refitting pass is slightly more expensive compared to classic BVH refitting.

6 Implementation Details

Following the SBVH implementation [Stich et al. 2009], an implementation of the MSBVH as described in the previous sections is straightforward. In the following, complementary aspects are detailed.

6.1 Updating the Reference Sorting Order

One common way to speed up acceleration hierarchy construction is to pre-sort the primitive index array, for example according to primitive centroids [Wald and Havran 2006]. However, when primitives are split spatially, the centroids of the resulting references need to be recomputed.

Thus, care must be taken to not invalidate the sorting order of the reference index array due to spatial splits. Consequently, for each spatial split, we keep a list of those primitives in the node that are duplicated due to a spatial split. We recompute centroids for those primitives, sort this array, and then use a merge-sort step to combine this array with the array of unsplit primitives, similar to [Wald and Havran 2006].

6.2 Hierarchy Traversal Order

Relying on one spatial split at a single time instant for the whole shutter interval requires coherent motion. Even with this assumption fulfilled, primitives in one node can move into opposite directions with respect to a spatial split plane and change bounding boxes. Thus, instead of using the split plane for determining the hierarchy traversal order, it can be more efficient to determine the traversal order according to the parametric distance of the ray-bounding box intersection.

6.3 Multiple Motion Segments

Using a single motion segment is not a sufficient approximation for non-linear motion like, for example, a rotating propeller. In such cases multiple motion segments need to be used for a single frame.

Restricting the number of motion segments to 2^m with equidistant times $t_i = \frac{i}{2^m}$ allows for the efficient determination of bounding boxes in case children in the hierarchy use different numbers of time samples. The power-of-two restriction guarantees that the existing motion segment endpoints for the children coincide, as we only need to take the maximum number of segments of the children and resample the children with a smaller number of motion segments.

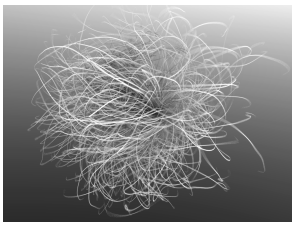
This avoids increasing the number of motion segments arbitrarily when propagating bounding boxes from children to parents. For example, if arbitrary numbers of motion segments were allowed, the parent node of four children with 2, 3, 5, and 7 motion segments would need $2 \cdot 3 \cdot 5 \cdot 7 = 210$ motion segments to accurately represent the children's transformation interpolations without approximations. In general, this number is equal to the least common multiple, which motivates restricting the number of motion segments to powers of two.

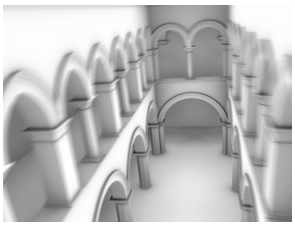
7 Results and Discussion

We implemented the MSBVH in the NVIDIA OptiX ray tracing framework [Parker et al. 2010] and compared the efficiency of the resulting trees to regular BVHs with node interpolation. Table 1 shows various performance statistics obtained by rendering a number of different test scenes with ambient occlusion shading. Throughout all test cases, a reduction of overall intersection and/or traversal steps can be observed. As in [Stich et al. 2009], the SAH costs of the hierarchies in all test cases is reduced by introducing spatial splits.

Our implementation focuses exclusively on tree quality rather than optimizing build performance. Thus, construction times of the MSBVH exceed those of a regular BVH by a fair amount. An optimized implementation of the MSBVH would likely narrow the gap significantly, but due to the inherent additional computation, its construction speed cannot reach that of a BVH.

Nevertheless, as soon as enough rays are traversed through the hierarchy, the additional overhead to build a hierarchy of higher quality





Build Method	References	Nodes	Spatial Splits	SAH Cost	Intersections (Max/Avg)	Traversals (Max/Avg)	Render Time
Hairball, 348K tris							
BVH	348,000	182,450	–	111	100% / 100%	100% / 100%	100%
MSBVH	871,321	599,276	51,689	106	32% / 43%	119% / 105%	85%
Rotated Sponza, 76K tris							
BVH	76,081	63,830	–	124	100% / 100%	100% / 100%	100%
MSBVH	140,525	121,486	9,138	102	41% / 47%	76% / 77%	72%
Bubs, 1888K tris							
BVH	1,888,229	1,994,490	–	40	100% / 100%	100% / 100%	100%
MSBVH	1,940,520	2,046,742	2,141	29	105% / 89%	105% / 93%	94%
Croc, 1237K tris							
BVH	1,292,986	1,356,650	–	139	100% / 100%	100% / 100%	100%
MSBVH	1,625,919	1,762,402	18,748	133	99% / 96%	97% / 95%	90%

Table 1: *Comparison of a regular bounding volume hierarchy (BVH) without spatial splits using interpolated bounding boxes and the MSBVH. The **References** column shows the number of primitive references in the final data structure. For the regular BVH, this corresponds to the original number of primitives in the scene. The **Nodes** column contains the number of tree nodes (internal nodes and leaves), giving an indication of the increased memory requirements for an MSBVH. **Spatial Splits** is the number of successful spatial splits executed during the MSBVH build. The theoretical efficiency of the resulting trees, according to the surface area heuristic, is shown in the **SAH Cost** column. **Intersections** and **Traversals** compare the number of ray-primitive intersections and number of acceleration structure traversal steps required to render the scene with one primary and one ambient occlusion ray. The average and maximum values across all the rays in the image are used for the comparison. **Render Time** compares frame render times (excluding acceleration structure build times) of the test scenes in our implementation in OptiX.*

will pay off. Thus for typical production scenes, where a lot of samples are drawn per frame, we expect decreased total image synthesis times when using the MSBVH.

Acknowledgements

The authors would like to thank Samuli Laine for the original hairball generator code, Ryan Vance for the static Bubs scene, and Knud Dollereder for the Croc model. The hairball was animated using the Bullet Physics library.

References

AMANATIDES, J., AND WOO, A. 1987. A fast voxel traversal algorithm for ray tracing. In *In Eurographics 87*, 3–10.

APODACA, T., GRITZ, L., PORTER, T., JACOB, O., TURNER, M., LETTERI, J., POON, E., AND ZARGARPOUR, H. 1995. Using RenderMan in animation production. In *SIGGRAPH '95: ACM SIGGRAPH 1995 Courses*, ACM Press.

CHRISTENSEN, P., FONG, J., LAUR, D., AND BATALI, D. 2006. Ray tracing for the movie 'Cars'. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing 2006*, 1–6.

HANIKA, J., KELLER, A., AND LENSCH, H. 2010. Two-level ray tracing with reordering for highly complex scenes. In *Proceedings of Graphics Interface 2010*, Canadian Information Processing Society, Toronto, Ont., Canada, Canada, GI '10, 145–152.

HOU, Q., QIN, H., LI, W., GUO, B., AND ZHOU, K. 2010. Micropolygon ray tracing with defocus and motion blur. In *ACM SIGGRAPH 2010 papers*, ACM, New York, NY, USA, SIGGRAPH '10, 64:1–64:10.

HUNT, W. 2008. Corrections to the surface area metric with respect to mail-boxing. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing 2008*, 77–80.

LAUTERBACH, C., YOON, S., TUFT, D., AND MANOCHA, D. 2006. RT-DEFORM: Interactive ray tracing of dynamic scenes using BVHs. *Symposium on Interactive Ray Tracing*, 39–46.

MUNKBERG, J., , HASSELGREN, J., TOTH, R., AND AKENINE-MÖLLER, T. 2010. Efficient bounding of displaced Bézier patches. In *Proceedings of the Conference on High Performance Graphics*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, HPG '10, 153–162.

OLSSON, J. 2007. *Ray-Tracing Time-Continuous Animations using 4D KD-Trees*. Master's thesis, Lund University.

PARKER, S., BIGLER, J., DIETRICH, A., FRIEDRICH, H., HOBEROCK, J., LUEBKE, D., MCALLISTER, D., MCGUIRE, M., MORLEY, K., ROBISON, A., AND STICH, M. 2010. Optix: A general purpose ray tracing engine. *ACM Transactions on Graphics* (August).

PHARR, M., AND HUMPHREYS, G. 2010. *Physically Based Rendering: From Theory to Implementation, 2nd edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

STICH, M., FRIEDRICH, H., AND DIETRICH, A. 2009. Spatial splits in bounding volume hierarchies. In *Proceedings of the Conference on High Performance Graphics 2009*, ACM, New York, NY, USA, HPG '09, 7–13.

SUTHERLAND, I., AND HODGMAN, G. 1974. Reentrant polygon clipping. *Commun. ACM 17* (January), 32–42.

WALD, I., AND HAVRAN, V. 2006. On building fast kd-trees for ray tracing, and on doing that in $O(N \log N)$. In *Proceedings of the IEEE Symposium on Interactive Ray Tracing 2006*, 61–69.

WALD, I., MARK, W., GÜNTHER, J., BOULOS, S., IZE, T., HUNT, W., PARKER, S., AND SHIRLEY, P. 2007. State of the art in ray tracing animated scenes. In *Eurographics 2007 State of the Art Reports*.