Vision, Modeling, and Visualization (2018)
F. Beck, C. Dachsbacher, and F. Sadlo (Eds.)

Compressed Bounding Volume Hierarchies for Efficient Ray Tracing of Disperse Hair

M. Martinek[1] and M. Stamminger[1] and N. Binder[2] and A. Keller[2]

[1]University of Erlangen-Nuremberg
[2]NVIDIA

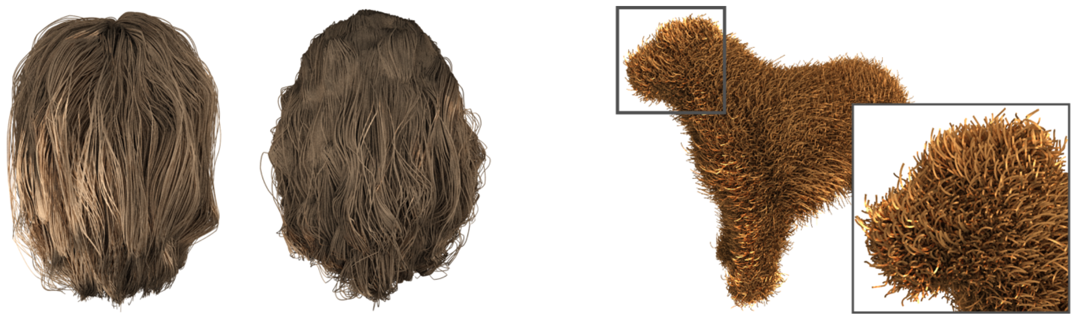


Figure 1: *Three different models for little to unstructured hair. From left to right: wavy and curly human hair as well as the dog model with 750.000 (wavy) and 1.000.000 (curly,dog) individual cubic Bézier curves, respectively. The wavy and curly hair models are courtesy of Cem Yuksel,* www.cemyuksel.com/research/hairmodels.

Abstract

Ray traced human hair is becoming more and more ubiquitous in photorealistic image synthesis. Despite hierarchical data structures for accelerated ray tracing, performance suffers from the bad separability inherent with ensembles of hair strands. We propose a compressed acceleration data structure that improves separability by adaptively subdividing hair fibers. Compression is achieved by storing quantized as well as oriented bounding boxes and an indexing scheme to specify curve segments instead of storing them.

We trade memory for speed, as our approach may use more memory, however, in cases of highly curved hair we can double the number of traversed rays per second over prior work. With equal memory we still achieve a speed-up of up to 30%, with equal performance we can reduce memory by up to 30%.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Ray tracing; Parametric curve and surface models;***

1. Introduction

Ray tracing human hair or fur is imperative when aiming for a realistic appearance of rendered characters. As for light transport simulation huge numbers of rays need to be traced through hair, efficiency is paramount. However, the nature of hair consisting of many thousands of individual strands of different lengths and directions is a challenge for efficient ray tracing: First, hair is usually modeled as a parametric curve with associated thickness, which leads to rather expensive intersection tests between a ray and a single hair. Second, the performance of a typical acceleration structure such as a volumetric kd-tree or a bounding volume hierarchy (BVH) is rather bad for hair due to its curved and thin shape.

In previous work [WBW*14], a hybrid hierarchy of axis-aligned and oriented bounding boxes has been proposed that leads to tighter bounding volumes and thus improved performance. In this paper, we further elaborate on this idea and propose a novel approach to construct a BVH for hair by contributing the following ideas:

1. Before building the hierarchy, we adaptively subdivide curvy hair into shorter, straighter segments, which can be better clustered in a bounding volume hierarchy.

DOI: 10.2312/vmv.20181258

2. We then build a BVH for these segments with axis-aligned or oriented bounding boxes, avoiding the memory-heavier oriented bounding boxes in the deep levels of the hierarchy.
3. Parameter intervals for hair segments and volume bounds are stored in a compressed manner, which significantly reduces overall memory consumption.

With our proposed hierarchy we can trade memory for performance: We achieve a speed-up of up to 50% in our test scenes at the price of increased memory. Using equal memory we still achieve a speed-up of up to 30% over prior work, whereas with equal performance we can reduce memory by up to 30%.

2. Related Work

Usually, individual hair strands are modeled using a parametric representation such as cubic Bézier splines, which can be intersected with rays directly [SN90, NO02, Res17]. However, such direct ray-spline intersection tests are more expensive than recursively subdividing a curve and intersecting with its line segments as in Nakamaru et al. [NO02] at the price of an enlarged memory footprint.

Typical acceleration data structures for ray tracing perform badly when applied to hair: For spatial partitioning, the same hair has to be referenced multiple times from all voxels it overlaps, whereas in object partitioning schemes bounding volumes are largely overlapping. Performance thus suffers from either deep tree traversal or inefficient pruning and culling due to overlap. Although there are many approaches for the hierarchy generation on general purpose geometric primitives, only little work exists which tackles the problem of parametric curves such as found in hair or fur models. Based on the fact that hair strands are often oriented similarly, Woop et al. [WBW*14] propose to use a hybrid BVH consisting of axis aligned bounding boxes (AABB) as well as oriented bounding boxes (OBB). The locally oriented bounding volumes enclose the parametric curves more tightly and hence prevent much of the above mentioned overlap. Due to the resulting tighter bounds pruning performance is improved. We use a similar structure, but subdivide primitives before sorting them into the hierarchy. Further, we also avoid oriented bounding boxes in the deep hierarchy levels, where their performance-memory trade-off is bad.

Deep acceleration hierarchies also suffer from their large memory footprint. Selgrad et al. [SLM*16] use OBBs as well as compression and quantization for a more efficient acceleration data structure for ray tracing subdivision surfaces. Similar to Ernst et al. [EG07] and Dammertz et al. [DK08], where triangles are split before building the BVH, Selgrad et al. [SLM*16] use a flatness criterion in order to subdivide the parametric surface. They then locally orient the bounds to the surface and use those bounding volumes directly for ray traversal instead of intersecting the parametric surface itself. We use ideas from this paper to compress our hierarchy and significantly reduce its memory footprint.

3. Algorithm

Our new ray tracing acceleration data structure trades memory for performance. Based on curvature adaptive subdivision of the Bézier curves and a compact indexing scheme (Section 3.1), we sacrifice only little memory for the ability to create much tighter bounding volumes. We then build a bounding volume hierarchy similar to Woop et al. [WBW*14], however, with restrictions on the OBBs being built only for inner nodes, i.e. nodes where all children are non-leaf nodes (Section 3.2). Then, the hierarchy is further compressed by deciding which bounding volume can be represented at quantized precision (Section 3.3).

3.1. Bézier Segment Generation

In our system, hair is modeled by individual long B-Spline strands. In a first step, these B-Spline curves are converted to connected cubic Bézier curves. Each cubic Bézier curve consists of four control points, c_0, c_1, c_2, c_3 with a 3D position and associated width of the curve. The examples shown in Figure 1 are modeled using approximately 22 Bézier curves per hair strand for the two human hair models, whereas the dog model consists of 4 Bézier curves due to the short nature of its fur.

Instead of constructing the bounding volume hierarchy over these Bézier curves, we first adaptively subdivide all Bézier curves with high curvature into *Bézier segments* B_σ to better approximate them by a bounding volume. We therefore measure curvature as the maximum distance $d_{\max} = \max(d_1, d_2)$. As pictured in Figure 2 the distance d_1 is defined as the Euclidean distance between the inner control point c_1 and the line through start- and endpoints c_0, c_3 of the cubic Bézier curve. d_2 is computed in the same way as d_1 using control point c_2.

The curve is subdivided recursively, as long as $d_{\max}$ exceeds a given threshold θ or reaches a maximum number of segments per curve, see algorithm Algorithm 1.

For each resulting segment B_σ we generate a new primitive and store a pointer to the original Bézier curve as well as the current segment offset.

By limiting the number of segments per curve to 2^n, the n least significant bits of a pointer can be used to encode the segment identifier $\sigma \in \{0, \dots 2^n - 1\}$. In our case, we use 64-bit pointers and their lower $n = 3$ bits to represent the curve segment over the parameter interval $\left[\frac{\sigma}{2^n}, \frac{\sigma+1}{2^n}\right]$.

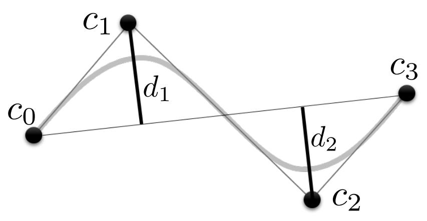


Figure 2: *A Bézier curve with control points c_0, c_1, c_2, c_3 is further subdivided if the value $d_{\max}$, defined as the maximum between d_0 and d_1, is larger than the given threshold θ.*

ALGORITHM 1
Adaptive Bézier Curve Subdivision

```
function FINDSUBD($B_\sigma$, s)
    $d\mathrm{max} \leftarrow \mathrm{CURVATURE}(B_\sigma)$
    if $d\mathrm{max} > \theta$ and $s < n$ then
        $left \leftarrow \mathrm{SUBDIVIDE}(B_\sigma)$
        $right \leftarrow \mathrm{SUBDIVIDE}(B_\sigma)$
        $s \leftarrow s+1$
        return max(FINDSUBD($left, s$), FINDSUBD($right, s$))
    else
        return s
    end if
end function
```

3.2. Bounding Volume Hierarchy Construction

We construct a hybrid bounding volume hierarchy with both axis-aligned and oriented bounding boxes [WBW*14, SLM*16]. The bounding volume hierarchy (as illustrated in Figure 3) is built top-down using the surface area heuristic (SAH). For each node, we decide whether an AABB or OBB is beneficial.

We first build a BVH similar to Wald et al. [Wal07], computing the expected SAH cost for traditional object partitioning in world space (creating AABBs) as well as in a local frame (creating OBBs). The local frame, which we call hair space, is determined by the dominant direction, i.e. the vector between the first and the last control point, of a randomly chosen B_σ^R and a randomized orientation around this axis.

We also employ the *similar orientation clustering* as proposed in Woop et al. [WBW*14] to better handle cases where two or more different hair strands intersect with each other. Object partitioning is not able to separate these strands into non-overlapping AABBs or OBBs. For this case we search for two Bézier segments B_σ^R and B_σ^M which define two clusters. All other Bézier segments are grouped depending on the Bézier segments' *similar alignment*. We define two curves to be more similar aligned if the angle spanned by the dominant curves' directions is smaller. B_σ^R is chosen randomly, whereas B_σ^M is the Bézier segment which is most unaligned with B_σ^R. The SAH cost is then computed from the bounds transformed into the hair space spanned of the hair cluster they belong to, i.e. either B_σ^R or B_σ^M.

In most cases the similar orientation clustering takes effect at almost leaf level and thus produces a massive amount of oriented bounds, which consume much more space than axis aligned bounds. For a 4-wide BVH, an AABB consumes 128 bytes (96 bytes for the bounds and 32 bytes for the four child pointers), whereas an OBB consumes almost twice as much, i.e. 224 bytes (192 bytes for the transformation matrices as well as 32 bytes for its children).

Since our tree structure is built upon much more elements than Woop's BVH [WBW*14], we decided to allow OBBs to be built only for *real* inner nodes, i.e. nodes where all four children are non-leaves. This has only low impact on performance but reduces memory requirements by over 50%.

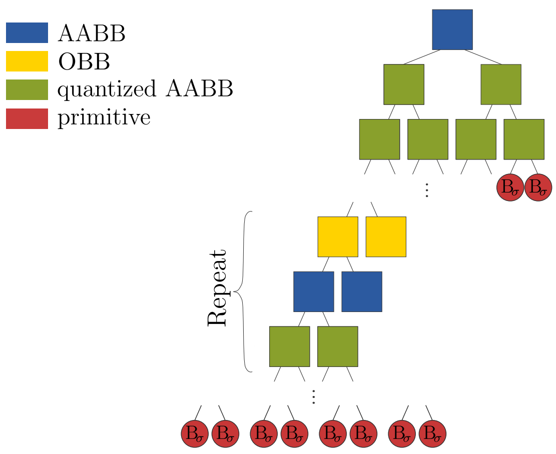


Figure 3: *Schematic of our BVH for hair rendering using subdivided curve segments B_σ (red) as primitives. Inner AABB nodes may either be stored using full precision (blue) or bounds quantized to 8 bits (green). Note, OBBs (yellow) are only created at inner nodes and the repeat section may be repeated throughout the hierarchy as long as all children of the OBB are inner nodes.*

3.3. Bounding Volume Hierarchy Quantization

Once the tree is constructed, all AABBs are quantized relative to their parent bounding volume using 8-bit precision similar to Cline et al. [CSE06] resulting in 56 bytes per quantized AABB (24 bytes for the bounds and 32 bytes for the four child pointers) instead of 128 bytes for a regular AABB.

The recursive quantization algorithm is depicted in Algorithm 2: For each child we check wether it uses an AABB or OBB. In case of an AABB the child bounds may be quantized if its parent node is also an AABB, otherwise a regular AABB is generated. Note that any AABB that is a direct child of an OBB is stored at full precision, i.e. is not quantized, as illustrated in Figure 3. This simplifies the traversal of the bounding volume hierarchy.

ALGORITHM 2
Bounding Volume Hierarchy Quantization

```
 1: function COMPRESS(node, parentUsesOBB)
 2:     for all child$_i$ in node.children do
 3:         if node.usesAABB then                ▷ node is axis aligned
 4:             if parentUsesOBB then
 5:                 CREATEAABB(child$_i$)
 6:             else
 7:                 CREATEQUANTAABB(child$_i$,node.bounds)
 8:             end if
 9:         else                                 ▷ node is oriented
10:             CREATEOBB(child)
11:         end if
12:         COMPRESS(child$_i$, node.usesAABB == true)
13:     end for
14: end function
```

3.4. Bounding Volume Hierarchy Traversal

We use single ray traversal for hair rendering as in [WBW*14], which is heavily inspired by the kernels found in Intel's Embree [WWB*14], and added our leaf nodes containing the 64-bit pointers referencing the individual subdivided segments of the original Bézier curves.

The decompression of the quantized nodes is performed during ray traversal by transforming the quantized bounds stored in a child using the current bounding volume as a frame as by Selgrad et al. [SLM*16].

The Bézier curve intersection is performed as in [NO02], where each curve is approximated using line segments which are generated by recursively subdividing the curve using a constant subdivision level L. This intersection test is approximate and has been vectorized and extended by Woop et al. [WBW*14]. As we adaptively pre-subdivide our Bézier curves into at most 2^n line segments (see Section 3.1), the constant curve subdivision level has to be adopted to $L-n$ for each curve individually during ray traversal.

4. Results and Discussion

For comparison we use the two human hair models as well as the dog model shown in Figure 1, where from left (wavy) to right (dog) the hair dispersion increases noticeably. For ray-Bézier curve intersection we use a subdivision level $L = 3$, which corresponds to 8 line segments per Bézier curve. This is sufficient for all three models, the selected camera position, and resolution, as the original hair strands are already represented by a relative large number of Bézier curves. We compare a regular bounding volume hierarchy $\mathbf{BVH_{AABB}}(B)$ using the input Bézier curves with axis aligned bounding boxes only and the $\mathbf{BVH_{Woop}}(B)$ as proposed by Woop et al. [WBW*14] using axis aligned and oriented bounding boxes to three variants using our curvature adaptive and pre-subdivided Bézier segments B_σ as described in Section 3.1:

1. $\mathbf{BVH_{AABB}}(B_\sigma)$ uses only AABBs.
2. $\mathbf{BVH_{Woop}}(B_\sigma)$ uses the original hierarchy as described by Woop et al. [WBW*14].
3. $\mathbf{BVH_{Ours}}(B_\sigma)$ uses our compressed hierarchy, which is an evolution of Woop's BVH [WBW*14], where AABBs are quantized and OBBs are constructed for non-leaf nodes only as described in Section 3.

We evaluate the impact of the threshold parameter θ, which controls the number of generated Bézier segments (see Section 3.1). In all results we use manually set thresholds, which differ for the different hair models: for the two human hair models $\theta = [0.075, 0.05, 0.025, 0.015]$, for the dog model $\theta = [0.15, 0.125, 0.1, 0.075]$. A more sophisticated approach to find a good theta, however, would be to determine θ based on the given hair curvature statistics. In addition, we analyze the different uses of OBBs with respect to memory consumption and ray tracing efficiency and show results for equal memory as well as equal performance.

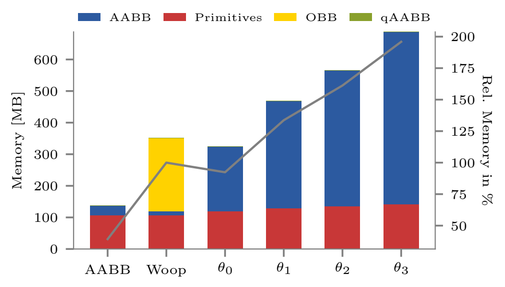
Our implementation has been integrated into Intel's Embree ray tracing framework [WWB*14].

4.1. Construction Time

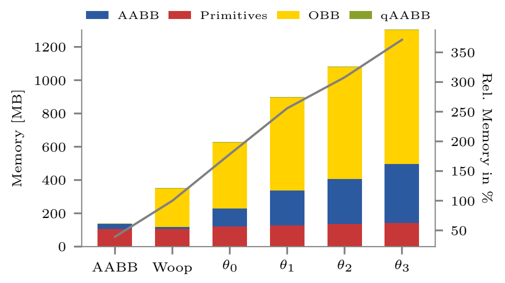
As expected, construction time increases linearly with the number of subdivided Bézier segments. For the example of the curly human hair model, $\mathbf{BVH_{Woop}}(B)$ needs 0.88 sec. for constructing the hierarchy over 100.000 primitives, whereas building $\mathbf{BVH_{Ours}}(B_\sigma)$ needs 2.7 sec. for approx. 350.000 Bézier segments including 0.4 sec. for primitive generation and 0.3 sec. for compression.

4.2. Memory Footprint

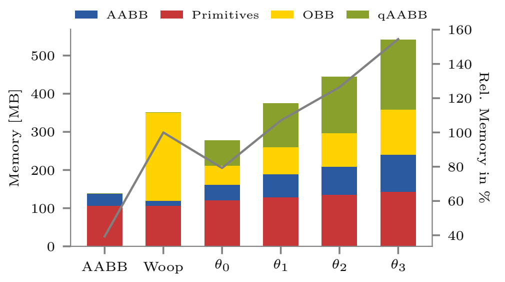
Figure 4 shows the memory consumption of the curly hair model for different θ. We show only the curly hair model as the other two models show similar memory behavior throughout the different hierarchies and only differ from the curly model by a constant factor.



(a) *$BVH_{AABB}(B_\sigma)$ for different θ*



(b) *$BVH_{Woop}(B_\sigma)$ for different θ*



(c) *$BVH_{Ours}(B_\sigma)$ for different θ*

Figure 4: *Absolute memory consumption in MB for different BVHs for the curly hair model. The black curve indicates the relative memory consumption of each BVH as compared to the $BVH_{Woop}(B)$. Note that $BVH_{AABB}(B)$ as well as $BVH_{Woop}(B)$ do not depend on θ, thus, their memory consumption does not change throughout Figures 4a to 4c.*

The absolute memory consumption in MB is shown in Figure 4 for axis aligned (blue), oriented bounding boxes (yellow), and quantized bounds (green), as well as primitives (red). Primitives in this case contain the original Bézier curves as well as our compact indexing for Bézier segments B_σ for all BVHs but $\mathbf{BVH_{AABB}}(B)$ and $\mathbf{BVH_{Woop}}(B)$, which contain the original Bézier curves only. The amount of primitives does not change for the same θ throughout the three charts since the number of pre-split curves is always the same for the same θ. The plotted line indicates the relative memory consumption in % relative to $\mathbf{BVH_{Woop}}(B)$ (= 100%).

As expected, using subdivided curve segments B_σ, we increase the overall number of primitives while decreasing the threshold θ. As a consequence, all three bounding volume hierarchies become deeper and the overall memory consumption increases. This memory increase becomes especially apparent when building $\mathbf{BVH_{Woop}}(B)$ over the segments B_σ (see Figure 4b), where memory increases by up to 3 times of the original $\mathbf{BVH_{Woop}}(B)$. This is due to the fact, that many OBBs are generated in the deeper levels (almost leaf levels) of the hierarchy. In this case quantizing inner nodes does not help much, because only few AABBs are generated in the first place.

Our approach tackles this memory increase by only allowing OBBs at inner nodes, preventing leaf nodes to become OBBs. This reduces the amount of memory for OBBs by up to one fifth of the original data as depicted in Figure 4c. The increase in AABBs is handled efficiently by quantization resulting in an overall lower memory footprint than a regular hierarchy using axis aligned bounds only.

Overall, for the lowest threshold θ_3 we use only 20% more memory than the original $\mathbf{BVH_{Woop}}(B)$, which uses no pre-subdivision of the input Bézier curves.

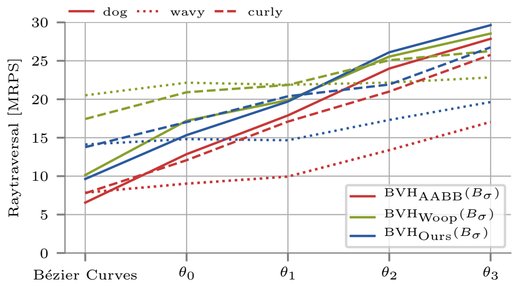
4.3. Ray Tracing Performance

Figure 5 depicts ray traversal performance in million rays per second (MRPS) for coherent (Figure 5a), and incoherent rays (Figure 5b) for three different hair models shown in Figure 1. Dotted lines show the performance for the wavy, dashed lines for the curly and continues lines for the dog model, respectively.

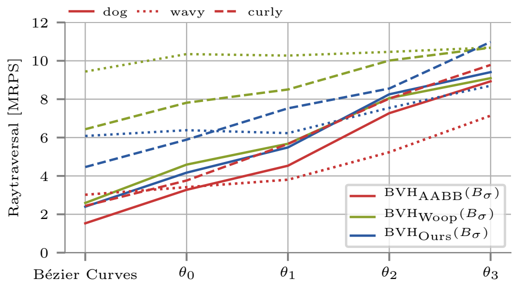
For any of the hair models all BVH approaches outperform the regular $\mathbf{BVH_{AABB}}(B)$ in both cases of coherent and incoherent rays. However, the performance of $\mathbf{BVH_{AABB}}(B)$ improves as θ becomes smaller.

$\mathbf{BVH_{Woop}}(B)$ performs the best for similarly shaped hair as seen in the wavy human hair model. $\mathbf{BVH_{Ours}}(B_\sigma)$ on the other hand, performs worse for similarly shaped Bézier curves, because in this case traversing the deeper BVH with its quantized bounds is more costly. In case of wavy hair, only for smaller thresholds θ the bounds can approximate the curve tightly enough so that traversal can be stopped earlier and performance just reaches the one of $\mathbf{BVH_{Woop}}(B)$.

However, when the dispersion of the Bézier curves becomes more apparent, as is the case for curly hair, and even more for the dog fur model, $\mathbf{BVH_{Ours}}(B_\sigma)$ performs increasingly better.



(a) *coherent rays*



(b) *incoherent rays*

Figure 5: *Ray tracing performance of all three approaches for different thresholds θ for coherent and incoherent ray traversal. Dotted lines represent the wavy, dashed lines the curly and continues lines the dog model, respectively.*

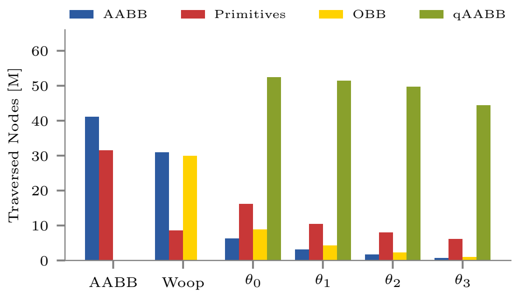


Figure 6: *Traversed nodes in millions split for the curly hair model using our proposed hierarchy $BVH_{Ours}(B_\sigma)$.*

Though, in this case, $\mathbf{BVH_{Ours}}(B_\sigma)$ performs only slightly better than $\mathbf{BVH_{Woop}}(B_\sigma)$ from θ_1 on, it consumes only half of the memory of $\mathbf{BVH_{Woop}}(B_\sigma)$. $\mathbf{BVH_{Ours}}(B_\sigma)$ at θ_2, which needs the same amount of memory as $\mathbf{BVH_{Woop}}(B)$, is 2 times faster than $\mathbf{BVH_{Woop}}(B_\sigma)$ for coherent as well as incoherent ray traversal.

Figure 6 explains the performance behavior seen in Figure 5 for $\mathrm{BVH_{Ours}}(B_\sigma)$. More precisely, it demonstrates the large impact of traversing leaf nodes, explicitly the costly ray-Bézier curve intersection test, which explains the drastic performance gain from $\mathbf{BVH_{AABB}}(B)$ over $\mathbf{BVH_{Woop}}(B)$.

Wavy Hair	Equal Memory coh./incoh. [MRPS]	Equal Performance [MB]
$\mathrm{BVH}_{\mathrm{AABB}}(B_\sigma)$	17.0/7.1	500
$\mathrm{BVH}_{\mathrm{Woop}}(B_\sigma)$	22.1/10.3	820
$\mathrm{BVH}_{\mathrm{Ours}}(B_\sigma)$	21.8/10.0	380
Curly Hair	**Equal Memory coh./incoh. [MRPS]**	**Equal Performance [MB]**
$\mathrm{BVH}_{\mathrm{AABB}}(B_\sigma)$	17.0/5.6	550
$\mathrm{BVH}_{\mathrm{Woop}}(B_\sigma)$	20.2/7.4	900
$\mathrm{BVH}_{\mathrm{Ours}}(B_\sigma)$	26.2/9.5	370
Dog Hair	**Equal Memory coh./incoh. [MRPS]**	**Equal Performance [MB]**
$\mathrm{BVH}_{\mathrm{AABB}}(B_\sigma)$	23.1/6.9	320
$\mathrm{BVH}_{\mathrm{Woop}}(B_\sigma)$	17.2/4.5	600
$\mathrm{BVH}_{\mathrm{Ours}}(B_\sigma)$	29.3/10.6	260

Table 1: *Table comparing $BVH_{AABB}(B_\sigma)$, $BVH_{Woop}(B_\sigma)$, and $BVH_{Ours}(B_\sigma)$ with respect to performance in MRPS when all three approaches require similar memory space, as well as with respect to memory in MB when all three approaches have similar performance.*

The further performance increase of any BVH using curve segments B_σ in case of disperse hair is explained by looking not only at the number of traversed leaf nodes but also at the number of traversed oriented bounding boxes. Even when the number of traversed leaves is slightly increased, in some cases of $\mathbf{BVH_{Ours}}(B_\sigma)$ the reduction of traversed oriented bounding boxes lets us gain overall performance, e.g. $\mathbf{BVH_{Ours}}(B_\sigma)$ at θ_2 as compared to $\mathbf{BVH_{Woop}}(B)$. When the number of ray-Bézier curve intersections is similar we can trace 20% more rays per second.

In case of $\mathbf{BVH_{Woop}}(B)$ the number of traversed leaf nodes is never larger than the one of $\mathbf{BVH_{Woop}}(B_\sigma)$ explaining the continuously good behavior of this BVH, even in case of wavy hair. However, this is the BVH, which by far needs the most memory space (more than 3 times as much as $\mathbf{BVH_{Woop}}(B_\sigma)$).

4.4. Comparative Results

For a better overview we show the evaluation of the three approaches for B_σ for equal memory as well as equal performance. Table 1 shows the results of these two setups for all three approaches.

When comparing all three methods at the same memory footprint, $\mathbf{BVH_{Ours}}(B_\sigma)$ performs the worst in only one case, which is for the wavy model, 5% for coherent and 3% for incoherent rays worse than $\mathbf{BVH_{Woop}}(B_\sigma)$. As soon as hair becomes more disperse, which is the case for the curly hair and dog model, $\mathbf{BVH_{Ours}}(B_\sigma)$ performs faster than the other two approaches. Especially for the dog model it performs more than 2 times faster than $\mathbf{BVH_{Woop}}(B_\sigma)$ in case of equal memory. Note that in order to achieve equal memory, the structure of $\mathbf{BVH_{Woop}}(B_\sigma)$ resembles the one of $\mathbf{BVH_{Woop}}(B)$ as it performs subdivision only for a few of the original Bézier curves. However, in this case also the $\mathbf{BVH_{AABB}}(B_\sigma)$ shows better performance than $\mathbf{BVH_{Woop}}(B_\sigma)$ due to the short nature of the B_σs, which are already well-bounded by AABBs. In this case, $\mathbf{BVH_{Ours}}(B_\sigma)$ is 26% faster than $\mathbf{BVH_{AABB}}(B_\sigma)$, and even 70% faster than $\mathbf{BVH_{Woop}}(B_\sigma)$ for coherent rays.

On the other hand, when all three approaches have similar traversal speed, for all three hair models $\mathbf{BVH_{Ours}}(B_\sigma)$ requires always less memory space, approximately only 50% of $\mathbf{BVH_{Woop}}(B_\sigma)$ and at most 80% of $\mathbf{BVH_{AABB}}(B_\sigma)$.

5. Conclusion

We improved the performance of ray tracing of disperse hair by constructing a bounding volume hierarchy that prunes more efficiently. Key of the performance gain are oriented bounding boxes, compression by quantization and indexing, as well as an adaptive subdivision scheme. In future work, we will investigate the implementation of our scheme on the GPU using compressed wide hierarchies as introduced by Ylitie et al. [YKL17].

References

[CSE06] Cline D., Steele K., Egbert P.: Lightweight bounding volumes for ray tracing. *Journal of Graphics Tools 11*, 4 (2006), 61–71. 3

[DK08] Dammertz H., Keller A.: The edge volume heuristic-robust triangle subdivision for improved bvh performance. In *Interactive Ray Tracing, 2008. RT 2008. IEEE Symposium on* (2008), IEEE, pp. 155–158. 2

[EG07] Ernst M., Greiner G.: Early split clipping for bounding volume hierarchies. In *2007 IEEE Symposium on Interactive Ray Tracing* (Sept 2007), pp. 73–78. 2

[NO02] Nakamaru K., Ohno Y.: Ray tracing for curves primitive. In *WSCG* (2002), pp. 311–316. 2, 4

[Res17] Reshetov A.: Exploiting budan-fourier and vincent's theorems for ray tracing 3d bézier curves. In *Proceedings of High Performance Graphics* (New York, NY, USA, 2017), HPG '17, ACM, pp. 5:1–5:11. 2

[SLM*16] Selgrad K., Lier A., Martinek M., Buchenau C., Guthe M., Kranz F., Schäfer H., Stamminger M.: A compressed representation for ray tracing parametric surfaces. *ACM Trans. Graph. 36*, 1 (Nov. 2016), 5:1–5:13. 2, 3, 4

[SN90] Sederberg T., Nishita T.: Curve intersection using bézier clipping. *Computer-Aided Design 22*, 9 (1990), 538–549. 2

[Wal07] Wald I.: On fast construction of sah-based bounding volume hierarchies. In *Interactive Ray Tracing, 2007. RT'07. IEEE Symposium on* (2007), IEEE, pp. 33–40. 3

[WBW*14] Woop S., Benthin C., Wald I., Johnson G., Tabellion E.: Exploiting local orientation similarity for efficient ray traversal of hair and fur. In *Proceedings of High Performance Graphics* (Aire-la-Ville, Switzerland, Switzerland, 2014), HPG '14, Eurographics Association, pp. 41–49. 1, 2, 3, 4

[WWB*14] Wald I., Woop S., Benthin C., Johnson G., Ernst M.: Embree: A kernel framework for efficient cpu ray tracing. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 143. 4

[YKL17] Ylitie H., Karras T., Laine S.: Efficient incoherent ray traversal on gpus through compressed wide bvhs. In *Proceedings of High Performance Graphics* (New York, NY, USA, 2017), HPG '17, ACM, pp. 4:1–4:13. 6