CGI2015 manuscript No.
(will be inserted by the editor)

Using Half-Precision Floating-Point Numbers for Storing Bounding Volume Hierarchies

Matias Koskela, Timo Viitanen, Pekka Jääskeläinen, Jarmo Takala & Ken Cameron

Abstract Bounding volume hierarchies, which are commonly used to speed up ray tracing, are heavily accessed during ray traversal. Reducing the memory footprint for the bounding volume hierarchies leads to a better cache hit ratio and, therefore, faster ray tracing. This paper examines an approach of using 16-bit half-precision floating-point numbers to store bounding volume hierarchy inner nodes. Compared to prior work on BVH node compression, which uses integers, floats are interesting because of their dynamic range. Compared to single-precision, half-precision inner nodes have half memory footprint and, therefore, they save an average of 1.7% total power. This comes with the small conversion cost overhead, average of 2.5% more ray-box intersection tests and average of 15% more ray-triangle intersection tests. If half floats are used hierarchically, rather than simply storing world coordinates as halfs, roughly the same accuracy is achieved as with single-precision. This eliminates the additional intersection tests, but it requires additional computations to obtain the world coordinates during traversal.

Keywords Ray tracing · Half-precision floating-point · Bounding volume hierarchy

1 Introduction

Historically, ray tracing has been widely used only for off-line rendering, but there is now active exploration

M. Koskela, T. Viitanen, P. Jääskeläinen, J. Takala
Department of Pervasive Computing,
Tampere University of Technology, Finland
E-mail: firstname.lastname@tut.fi, timo.2.viitanen@tut.fi

K. Cameron
Rational Circus, Bristol, United Kingdom
E-mail: ken@kencameron.net

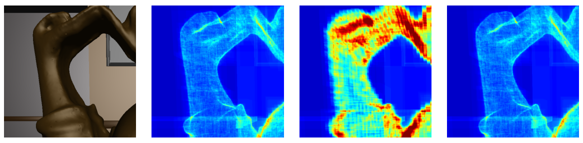

Fig. 1 The Buddha's hand far away from origin of the scene. Images from left to right: rendered scene, and workloads for the single-precision BVH, half-precision BVH, and hierarchical half-precision BVH. This is a difficult case for the world coordinate half-precision BVH, but the hierarchical version is indistinguishable from single-precision.

of its real-time application. Modern ray tracing makes use of acceleration data structures, such as kd-trees and *Bounding Volume Hierarchies* (BVH), which reduce the number of intersection tests required to find the closest intersecting triangle in a 3D scene.

The BVH usually stores *Axis Aligned Bounding Boxes* (AABB) in a tree like structure. Rays traverse the structure in a recursive manner by testing whether they intersect a child's bounding box or not. Many modern ray tracers have BVHs with a high branching factor and use SIMD instructions to compute multiple intersection tests in parallel; this is called *Multi Bounding Volume Hierarchy* (MBVH) [4]. Both BVHs and MBVHs can be constructed based on *Surface Area Heuristics* (SAH) [8]. This means that changes on data types and the scene might change the tree significantly.

Ray tracing is often a memory bound operation. BVHs have a large memory footprint. The optimal BVH node count is around half the number of triangles in the scene [9]. BVH memory requirement might seem small compared to the memory requirements of the triangles. However, BVH memory is the most frequently accessed area during the traversal. This means that it is more

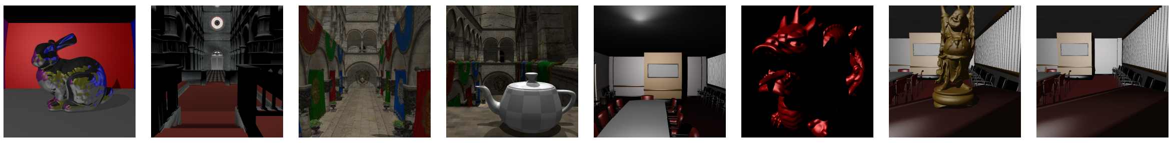

Fig. 2 Test scenes and the number of triangles and point lights in them from left to right: shinny Bunny(7K; 1), Sibenik (75K; 1), Sponza(262K; 2), Teapot inside Sponza(278K; 2), Conference (331K; 1), Dragon (871K; 1), Buddha inside Conference (1 419K; 1) and Buddha far from camera inside Conference (1 419K; 1).

beneficial for cache hit ratio, if the size of the BVH is reduced.

In this paper, we examine the trade-offs in half-precision floating-point format to store BVHs. This decreases the amount of bytes required to store each inner node, but more inaccurate AABBs may lead BVH traversal to examine subtrees unnecessarily, increasing the number of ray-AABB and ray-triangle tests performed. We also investigate a hierarchical storage format which eliminates most unnecessary tests, but requires extra computation for decompression. Figure 1 demonstrates these concepts.

2 Related Work

BVH Data can be compressed by inferring half of the child AABB coordinates from the parent AABB [5]. This format is less applicable to MBVHs since the same number of parent AABB coordinates is shared between an increased number of child AABBs.

Several other compression ideas that work with MBVHs has been proposed. In an extreme solution only two bits are stored in a BVH node [3]. Compression to 8 or 4 -bit integers has also been proposed [9]. The accuracy of the lower levels of the BVH can be increased by making the system hierarchical. However, this kind of compression is always a trade-off between compression ratio, quality of decompressed data and computation overhead needed for decompression.

In this paper the BVH is compressed to 16-bit half-precision floating-point numbers with and without the hierarchical format. Compared to integer's uniform interval quantization points, we expect floating-point numbers' dynamic range to work better with the teapot-in-a-stadium problem. In addition, recent CPUs and GPUs provide hardware-accelerated instructions for at least conversions between single and half-precision float [1,2,7,10]. Despite the available instruction-set support, to the best of our knowledge, the use of half-precision has not been systematically studied yet for storing AABBs in BVHs.

3 Inner nodes as Half-precision Floats

In a typical BVH implementation, every inner node stores the AABBs' upper and lower bounds of its every child in six single-precision floats. In addition, an integer pointer to each child node is needed, resulting in a total memory footprint of $7 \times 4 = 28$ bytes per AABB. This may be padded to 32 bytes per child in order to improve the cache access pattern. If those six values are stored with half-precision floats; BVH inner nodes would use 12 bytes for the bounding volume and 4 bytes for the child pointer. This means 16 bytes in total, which is already cache line friendly without any padding. For example, in a MBVH4, the size of an inner node would halve from 128 bytes to 64 bytes

Half floats can be used as plain world coordinates just like single floats are used. If halfs are too inaccurate as plain world coordinates they can be used hierarchically. This can be done by setting half's minimum value (-65504) to be equal to the parent AABB's lower bound P_{lower} and half's maximum (65505) value to be equal to the parent AABB's upper bound P_{upper}.

Algorithm 1 Decompression of hierarchical data.

```
float32 HierToWorldCoord($v$, $P_{lower}$, $P_{upper}$)
    $value \leftarrow convertTo32Bitfloat(v) - H_{min}$
    $P_{range} \leftarrow P_{upper} - P_{lower}$
    return $P_{range} \times (value \div H_{range}) + P_{lower}$
```

Basic idea for decompressing the hierarchical encoding can be seen in Algorithm 1. In the algorithm, input v is in the compressed data type and all other values are in single-precision floats. The algorithm first converts value to range from 0.0 to 1.0. This is done with the minimum H_{min} value and the range H_{range} of the data type. The result is then used for deciding a point from parent's lower limit to parent's upper limit.

With both, plain world coordinates and hierarchical encoding, the quantization to half-precision causes errors in the bounding volumes, which leads to visible holes in the model. Holes occur when the size of the

Table 1 The number of ray-AABB and ray-triangle tests in a whole frame with BVHs stored in single (F32), half (F16), hierarchical half (F16 H), 16-bit integer (I16), hierarchical 16-bit integer (I16 H) and hierarchical 8-bit integer (I8 H) precisions.

	AABB tests (K)						Triangle tests (K)					
Model	F32	F16	F16 H	I16	I16 H	I8 H	F32	F16	F16 H	I16	I16 H	I8 H
Bunny	22452	1.2%	0.1%	0.1%	0.0%	-4.2%	3668	33%	0.0%	14%	0.0%	2.2%
Sibenik	58352	1.6%	2.0%	2.1%	0.3%	-1.1%	8847	12%	2.8%	9.9%	0.5%	5.5%
Sponza	129148	0.9%	0.3%	0.4%	0.0%	-0.1%	26841	9.9%	0.6%	7.1%	0.0%	7.3%
Teapot	114071	1.7%	1.7%	3.3%	1.5%	3.4%	21814	5.1%	0.0%	15%	0.0%	2.8%
Confer.	23251	5.9%	3.2%	5.2%	0.7%	5.8%	6130	31%	6.7%	28%	1.0%	20%
Dragon	43012	5.4%	-0.1%	0.1%	0.0%	2.1%	9983	16%	0.2%	1.4%	0.2%	-1.2%
Buddha	46749	4.1%	2.5%	5.1%	1.6%	-1.3%	12192	21%	7.4%	25%	5.4%	11%
FarBud.	30006	6.5%	5.3%	5.9%	1.8%	6.0%	9035	24%	14%	21%	0.5%	21%
average	58380	2.5%	1.5%	2.4%	0.7%	1.2%	12314	15%	3.0%	14%	0.8%	7.6%

bounding volume is shrunk so that it does not actually cover all of its children's geometry. The ray tracer needs to make sure that every upper bound of a bounding volume is rounded towards positive infinity and every lower bound is rounded towards negative infinity. Fortunately, at least some of the instruction sets have native support for different rounding modes [2,7].

Tree construction must use the same precision as the resulting BVH, in order to ensure the best SAH result. In order to construct optimal trees for the scene without any hierarchy in coordinate compression, we emphasized half-precision floating-point numbers' better precision close to the origin, by centering the coordinates used in the BVH on the camera. Camera-centered trees perform well in the teapot-in-a-stadium problem. Centering may be impractical when a tree is constructed only once and used to render multiple viewpoints, but in animated scenes it is asymptotically inexpensive compared to reconstruction or refitting that is done on each frame. We also tested other heuristics for choosing the origin point, but the camera position seems to work best in the general case.

4 Evaluation

For evaluation we implemented an OpenCL-based ray tracer which uses Whitted-style ray tracing [12] with primary, secondary and shadow rays. Scenes in Fig. 2 were rendered with a resolution of 1024 x 1024. The ray tracer used MBVH4 for testing all 4 children of a inner node and up to 4 triangles of a leaf node simultaneously.

Both non-hierarchical and hierarchical half-precision float formats were tested. The results were compared to the single-precision and the integer representations with and without hierarchy as proposed by Mahovsky et al. [9]. The resulting numbers of required ray-AABB and ray-triangle intersection tests are in Table 1, where parallel intersection tests are calculated as one test.

As expected, due to the dynamic range, plain halfs perform better than plain integers with teapot-in-a-stadium scenes *Teapot* and *Buddha*. However, with other scenes integers perform better, which leads to about the same average performance for both of them.

With hierarchical encoding, the 16-bit integer's performance is the best on average, but half-precision floats performance is only 1.0% worse. The two are so close to each other that the order is dependent on the targeted hardware. For example on Intel's Haswell CPU architecture, converting a vector of eight halfs to eight singles (VCVTPH2PS) has a latency of 4 cycles and throughput of 1 per cycle [6]. In contrast, converting eight 16 or 8-bit integers to 32-bit (VPMOVSX) and converting the 32-bit integers into floats (VCVTDQ2PS) both have latencies of 3 cycles and throughputs of 1 per cycle [6]. Since halfs can be converted with one-half the instructions and one-third less total latency, they are expected to be somewhat faster.

In addition to test counts, we compared power usage, by running the ray tracer on the ARM Mali-T628 GPU on Odroid XU3 board. In order to isolate the interesting part, we disabled the rendering and most of the shading. On average the memory used 10% less power, with half-precision than with single-precision. In addition, the GPU used 1.4% less power, while the rays per second count increased 0.8%. This results in 1.7% less power in total, since total power usage is dominated by the GPU usage. In contrast, 16-bit integers' improvement on average was not measurable. We believe this is because Mali-T628 is more optimized to half floats than 16-bit integers.

In order to evaluate memory savings in a practical renderer, we also implemented half-precision extension without hierarchy to Intel's Embree ray tracing kernel collection [11]. We used a typical structure of a 4-wide MBVH with a maximum of 4 triangles per leaf node. We replaced the original 64-bit addresses with 32-bit offsets so as to fit each node into exactly 64 bytes, however, this requires some additional computations. The viewpoints in Figure 2 were rendered with the Embree example path tracer at 16 samples per pixel. Tracing was per-

Table 2 Evaluation results with Intel Embree. Memory readings contain the whole BVH including the leaf triangles.

	Mem (MB)		Speed (Mrays/s)		L1 miss (%)	
Model	F32	F16	F32	F16	F32	F16
Bunny	5.0	-8.0%	7.91	-8.5%	3.29	-12%
Sibenik	5.5	-5.5%	5.46	-7.9%	7.41	-22%
Sponza	18.6	-6.4%	5.26	-7.2%	6.41	-26%
Teapot	19.7	-6.6%	5.10	-6.2%	4.88	-23%
Confer.	23.9	-6.3%	5.65	-5.4%	6.26	-10%
Dragon	65.6	-7.3%	5.31	-6.6%	5.40	-16%
Buddha	106.8	-7.3%	5.12	-9.4%	6.47	-11%
FarBud.	106.9	-7.4%	5.08	-6.8%	6.46	-10%
average	44.0	-6.8%	5.61	-7.2%	5.82	-16%

formed with 4 threads on an Intel Core i7-4500U CPU. We measured tracing performance and total memory consumption including both the BVH tree and the leaf primitives. Moreover, data cache misses were measured using the Linux *perf* utility. The results are shown in Table 2. Half-precision floats reduce memory footprint by 7% and cache misses by 16% on average. This shows that, though the triangles take up more memory than nodes, the nodes receive much more traffic. However, due to conversion overheads and the additional node and primitive tests incurred by the non-hierarchical format, performance was approximately 7% lower than the baseline. We also experimented with an 8-wide MBVH which maps better to Intel's conversion intrinsics, reducing the conversion overhead. This improves performance, but not enough to match the baseline.

5 Conclusions

We have shown that when BVHs are stored in half-precision floating-point instead of single-precision the amount of memory bandwidth is halved for a ray tracing traversal step. This is not that obvious because also the pointer to the next inner node needs to be loaded. In a practical renderer, halving BVH inner nodes' storage area reduces L1 cache misses by 16%. We also measured average of 1.7% total power saving in ray traversal.

The downside is the need for average of 2.5% more ray-AABB tests and 15% more ray-triangle tests. If hierarchical encoding of half-precision floats is used, the average test counts are almost indistinguishable from single-precision. However, hierarchical encoding requires more computation during ray traversal. Due to the dynamic range, halfs perform better than 16-bit integers with small triangles close to the origin of the scene. With hierarchical encoding the two are so close to each other that superiority is determined by the targeted hardware's speed of converting the data types to single-precision floats. At least, some hardware is optimized for floating-point operations rather than integers.

In the future, we are interested in storing triangles and natively calculating intersections in half precision.

Acknowledgement

The authors would like to thank Martin Newell, Anat Grynberg, Greg Ward, Frank Meinl, Marko Dabrovic and Stanford 3D Scanning Repository for the test scene models. In addition, the authors are thankful to their funding sources: TUT graduate school, Academy of Finland (funding decision 253087), Finnish Funding Agency for Technology and Innovation (project "Parallel Acceleration 2", funding decision 40081/14), and ARTEMIS JU under grant agreement no 641439 (ALMARVI).

References

1. AMD: 1.5 16-bit floating-point data type, 128-bit SSE5 instruction set (2007). URL http://amd-dev.wpengine.netdna-cdn.com/wordpress/media/2012/10/AMD64_128_Bit_SSE5_Instrs.pdf. Referenced: 3/24/2015
2. ARM: VCVT (between half-precision and single-precision floating-point), realview compilation tools assembler guide (2010). URL http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0204j/CJAIIIFC.html. Referenced: 3/24/2015
3. Bauszat, P., Eisemann, M., Magnor, M.A.: The minimal bounding volume hierarchy. In: VMV, pp. 227–234 (2010)
4. Ernst, M., Greiner, G.: Multi bounding volume hierarchies. In: IEEE Symposium on Interactive Ray Tracing, 2008. RT 2008., pp. 35–40 (2008)
5. Fabianowski, B., Dingliana, J.: Compact BVH storage for ray tracing and photon mapping. In: Proc. of Eurographics Ireland Workshop, pp. 1–8 (2009)
6. Fog, A.: 4. instruction tables (2014). URL http://www.agner.org/optimize/instruction_tables.pdf. Referenced: 3/26/2015
7. Konsor, P.: Performance benefits of half precision floats (2012). URL https://software.intel.com/en-us/articles/performance-benefits-of-half-precision-floats. Referenced: 3/26/2015
8. MacDonald, J.D., Booth, K.S.: Heuristics for ray tracing using space subdivision. The Visual Computer **6**(3), 153–166 (1990)
9. Mahovsky, J., Wyvill, B.: Memory-conserving bounding volume hierarchies with coherent raytracing. In: Computer Graphics Forum, vol. 25, pp. 173–182. Wiley Online Library (2006)
10. NVidia: 3.2.11.1.3. 16-bit floating-point textures, CUDA toolkit v6.5 documentation (2015). URL http://docs.nvidia.com/cuda/cuda-c-programming-guide/#sixteen-bit-floating-point-textures. Referenced: 3/26/2015
11. Wald, I., Woop, S., Benthin, C., Johnson, G.S., Ernst, M.: Embree: A kernel framework for efficient CPU ray tracing. ACM Trans. Graph. **33**(4), 143:1–143:8 (2014). DOI 10.1145/2601097.2601199
12. Whitted, T.: An improved illumination model for shaded display. Commun. ACM **23**(6), 343–349 (1980)