Eurographics Symposium on Rendering - Experimental Ideas & Implementations (2016)
E. Eisemann and E. Fiume (Editors)

Node Culling Multi-Hit BVH Traversal: Supplemental Materials

Christiaan Gribble[†]

Applied Technology Operation
SURVICE Engineering

As noted in Section 3 of the main text, we provide graphs comparing average frame time for the *find-some-intersections* case for each test scene in Figure 1. Similarly, we provide data for ray/node intersection tests, node traversal operations, ray/primitive intersection tests, and average frame time in Table 1.

We also provide source code for the experimental reference implementation used to evaluate our node culling multi-hit BVH traversal algorithm in Section 3 of the main text. The key elements of this project include:

- **`BVH.h`** defines and implements a BVH with optional (compile-time) support for every- and leaf-node TCBs, support for unconstrained ICBs, and convenience functions for standard first-hit traversal. As noted in Section 3 of the main text, the BVH structure is built using a readily available surface area heuristic construction algorithm [WBS07].
- **`CMakeLists.txt`** provides content for compiling the reference implementation using the CMake build system.
- **`Constants.h`** defines useful values employed throughout the reference implementation, including the default number of primitives at which to construct a leaf node.
- **`Node.h`** defines and implements an intersectable BVH node using Williams-style ray/box intersection [WBMS05].
- **`Renderer.{h,cc}`** define and implement an abstract base class from which specific renderers are derived, as well as several inner classes, including per-ray payload data.
- **`RendererFH.{h,cc}`** define and implement a standard first-hit renderer with simple eyelight shading.
- **`RendererMH.{h,cc}`** define and implement the various multi-hit renderers described in this work, including naive multi-hit and uICB-, eTCB-, and lTCB-based node culling multi-hit renderers. Each multi-hit renderer defines callback functions appropriate to its intended behavior, and all multi-hit renderers inherit an intermediate multi-hit base class that provides a shader implementing either eyelight shading at first-hit visible surfaces or alpha-blending across multiple surfaces, depending on the current rendering configuration.
- **`mhBVH.cc`** implements a driver program for rendering images using any of the renderers described above, configurable at run-time via various command line arguments.

The `README` file in the top-level source directory provides instructions for building and running the driver program.

Source code in the **`common/`** subdirectory provides common elements used by the reference implementation (file I/O, math operations, and so forth), but is not directly related to the multi-hit ray traversal techniques described in this work.

Similarly, the **`scenes/`** subdirectory provides example scene, view, and geometry/material files for the Cornell Box scene. To render different scenes, generate files that mimic the basic structure and format of the Cornell Box example.

Access to the most recent stable release of the reference implementation is available via the project homepage at:

`http://www.rtvtk.org/~cgribble/research/mhBVH/`

Additionally, read-only access to the reference implementation development repository is available via HTTP with `git`:

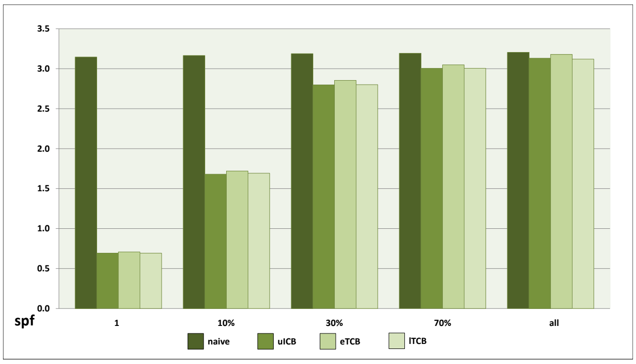
```
git clone http://www.rtvtk.org/code/mhBVH.git
```

Unless otherwise stated directly in the source, the reference implementation is distributed under the BSD 3-Clause License. Please see the `LICENSE` file distributed with the source for more information.

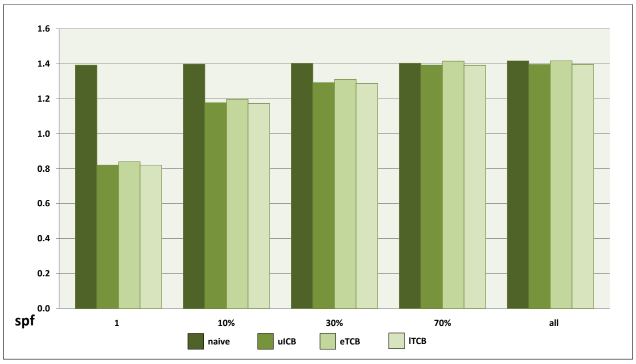
References

[WBMS05] WILLIAMS A., BARRUS S., MORLEY R. K., SHIRLEY P.: An efficient and robust ray-box intersection algorithm. *Journal of Graphics, GPU, and Game Tools 10*, 1 (2005), 49–54. 1

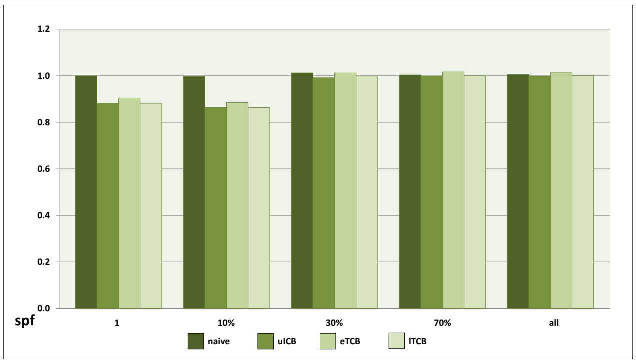
[WBS07] WALD I., BOULOS S., SHIRLEY P.: Ray tracing deformable scenes using dynamic bounding volume hierarchies. *ACM Transactions on Graphics 26*, 1 (January 2007), 6. 1



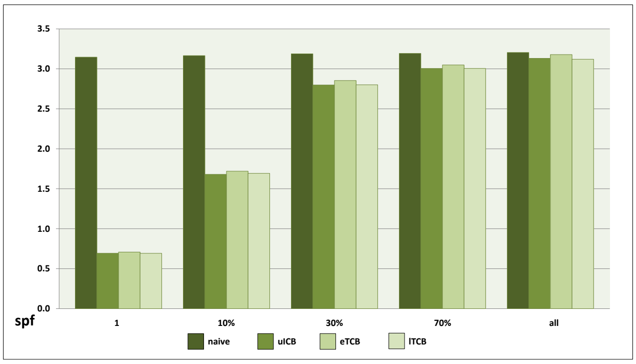
(a) sibe



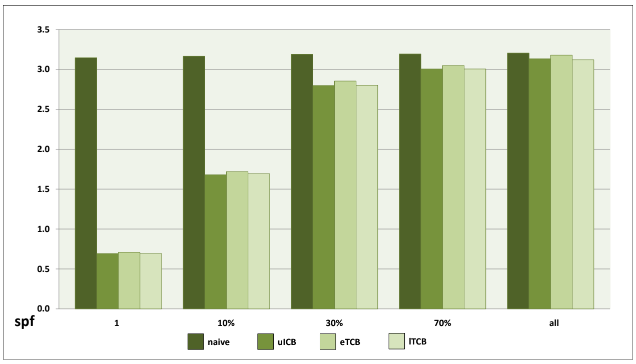
(b) fair



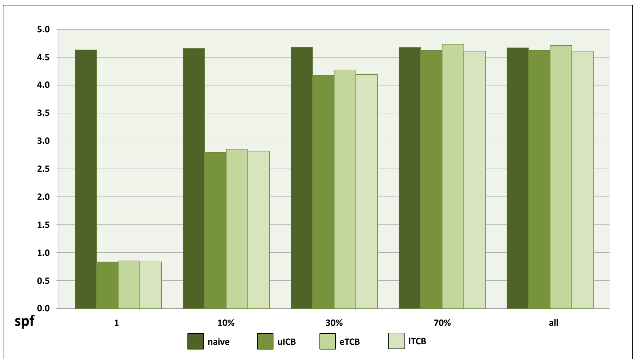
(c) conf



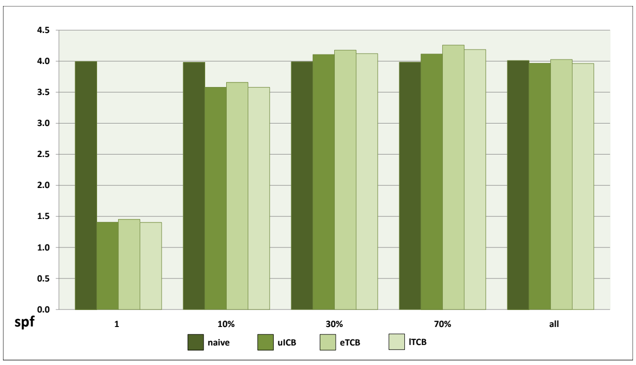
(d) truck



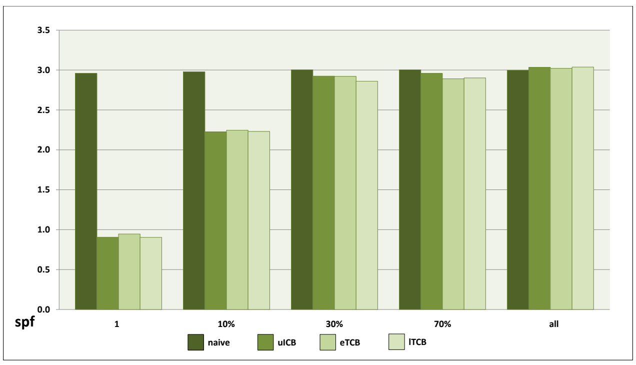
(e) tank



(f) hball



(g) sanm



(h) pplant

Figure 1: Performance of multi-hit variants for *find-some-intersections. Here, graphs compare multi-hit performance in seconds per frame (spf) among multi-hit implementations for various values of* N_{query} *in each test scene.*

	n-isec				n-trav				p-isec				avg ft			
scene	**naive**	**ICB**	**eTCB**	**lTCB**	**naive**	**ICB**	**eTCB**	**lTCB**	**naive**	**ICB**	**eTCB**	**lTCB**	**naive**	**ICB**	**eTCB**	**lTCB**
$N_{query} = 1$																
sibe	55400712	36383664	36383664	36383664	30332690	18979380	18979380	18979380	18144332	7195404	7195404	7195404	1.046	0.660	0.672	0.656
fair	71022840	48850760	48850760	48850760	38217252	25322106	25322106	25322106	19289710	8131664	8131664	8131664	1.392	0.821	0.839	0.820
conf	44550184	42118312	42118312	42118312	25367354	23784096	23784096	23784096	19225252	17152398	17152398	17152398	1.000	0.881	0.904	0.882
truck	91806616	26612608	26612608	26612608	53525104	14354699	14354699	14354699	52293204	10408216	10408216	10408216	3.146	0.693	0.708	0.692
tank	79729640	22500428	22500428	22500428	44636604	11713839	11713839	11713839	29420298	4825322	4825322	4825322	1.940	0.498	0.504	0.494
hball	148147216	39134168	39134168	39134168	86259392	20447722	20447722	20447722	82742544	9456689	9456689	9456689	4.630	0.835	0.854	0.836
sanm	206542256	105857896	105857896	105857896	113218336	54454884	54454884	54454884	56876344	10429580	10429580	10429580	3.995	1.406	1.451	1.402
pplant	103008768	41346636	41346636	41346636	56673416	21532986	21532986	21532986	51131620	10829268	10829268	10829268	2.958	0.906	0.946	0.904
$N_{query} = 10\%$																
sibe	55400712	44431284	44431284	44431284	30332690	23751254	23751254	23751254	18144332	11697977	11697977	11697977	1.043	0.838	0.851	0.833
fair	71022840	62531632	62531632	62531632	38217252	33299612	33299612	33299612	19289710	15250666	15250666	15250666	1.398	1.177	1.197	1.173
conf	44550184	42551396	42551396	42551396	25367354	24087324	24087324	24087324	19225252	17640304	17640304	17640304	0.997	0.864	0.884	0.863
truck	91806616	55968092	55968092	55968092	53525104	31919070	31919070	31919070	52293204	28765286	28765286	28765286	3.165	1.680	1.720	1.693
tank	79729640	44884848	44884848	44884848	44636604	24737136	24737136	24737136	29420298	15216834	15216834	15216834	1.949	1.010	1.031	1.012
hball	148147216	105949736	105949736	105949736	86259392	59789360	59789360	59789360	82742544	48082900	48082900	48082900	4.653	2.793	2.853	2.818
sanm	206542256	189157248	189157248	189157248	113218336	102948224	102948224	102948224	56876344	47863724	47863724	47863724	3.979	3.578	3.656	3.579
pplant	103008768	80821248	80821248	80821248	56673416	44012164	44012164	44012164	51131620	35227704	35227704	35227704	2.978	2.224	2.246	2.230
$N_{query} = 30\%$																
sibe	55400712	54943012	54943012	54943012	30332690	30054604	30054604	30054604	18144332	17854586	17854586	17854586	1.049	1.113	1.131	1.107
fair	71022840	70334360	70334360	70334360	38217252	37806000	37806000	37806000	19289710	18886234	18886234	18886234	1.402	1.292	1.311	1.287
conf	44550184	44364588	44364588	44364588	25367354	25249032	25249032	25249032	19225252	19081206	19081206	19081206	1.012	0.991	1.012	0.995
truck	91806616	84051216	84051216	84051216	53525104	48879564	48879564	48879564	52293204	47348764	47348764	47348764	3.188	2.798	2.854	2.801
tank	79729640	67095828	67095828	67095828	44636604	37470180	37470180	37470180	29420298	24532782	24532782	24532782	1.950	1.566	1.594	1.572
hball	148147216	141312800	141312800	141312800	86259392	81889936	81889936	81889936	82742544	76882504	76882504	76882504	4.679	4.175	4.270	4.190
sanm	206542256	206273264	206273264	206273264	113218336	113050776	113050776	113050776	56876344	56694784	56694784	56694784	3.988	4.107	4.177	4.122
pplant	103008768	98511064	98511064	98511064	56673416	54119788	54119788	54119788	51131620	48402472	48402472	48402472	3.002	2.923	2.921	2.859
$N_{query} = 70\%$																
sibe	55400712	55399652	55399652	55399652	30332690	30332078	30332078	30332078	18144332	18143856	18143856	18143856	1.057	1.042	1.062	1.041
fair	71022840	70989808	70989808	70989808	38217252	38196700	38196700	38196700	19289710	19265268	19265268	19265268	1.403	1.392	1.415	1.391
conf	44550184	44550172	44550172	44550172	25367354	25367350	25367350	25367350	19225252	19225246	19225246	19225246	1.003	0.998	1.016	0.999
truck	91806616	91782232	91782232	91782232	53525104	53509712	53509712	53509712	52293204	52272208	52272208	52272208	3.193	3.005	3.049	3.005
tank	79729640	79606760	79606760	79606760	44636604	44565068	44565068	44565068	29420298	29362708	29362708	29362708	1.959	1.926	1.960	1.933
hball	148147216	148045200	148045200	148045200	86259392	86196080	86196080	86196080	82742544	82666968	82666968	82666968	4.673	4.619	4.732	4.608
sanm	206542256	206538688	206538688	206538688	113218336	113216016	113216016	113216016	56876344	56873620	56873620	56873620	3.984	4.115	4.259	4.187
pplant	103008768	102817536	102817536	102817536	56673416	56565552	56565552	56565552	51131620	50999808	50999808	50999808	3.003	2.957	2.890	2.901
$N_{query} = \infty$																
sibe	55400712	55400712	55400712	55400712	30332690	30332690	30332690	30332690	18144332	18144332	18144332	18144332	1.049	1.039	1.064	1.044
fair	71022840	71022840	71022840	71022840	38217252	38217252	38217252	38217252	19289710	19289710	19289710	19289710	1.417	1.396	1.417	1.396
conf	44550184	44550184	44550184	44550184	25367354	25367354	25367354	25367354	19225252	19225252	19225252	19225252	1.005	0.997	1.012	1.001
truck	91806616	91806616	91806616	91806616	53525104	53525104	53525104	53525104	52293204	52293204	52293204	52293204	3.205	3.132	3.179	3.120
tank	79729640	79729640	79729640	79729640	44636604	44636604	44636604	44636604	29420298	29420298	29420298	29420298	1.952	1.914	1.961	1.933
hball	148147216	148147216	148147216	148147216	86259392	86259392	86259392	86259392	82742544	82742544	82742544	82742544	4.667	4.618	4.709	4.608
sanm	206542256	206542256	206542256	206542256	113218336	113218336	113218336	113218336	56876344	56876344	56876344	56876344	4.011	3.964	4.027	3.963
pplant	103008768	103008768	103008768	103008768	56673416	56673416	56673416	56673416	51131620	51131620	51131620	51131620	2.997	3.033	3.021	3.037

Table 1: Key metrics for multi-hit performance. *For each scene, we report the number of ray/node intersection tests (n-isec), node traversal operations (n-trav), ray/primitive intersection tests (p-isec), and average frame time in seconds (avg ft) before correctly satisfying the multi-hit query for each implementation. As expected, node culling significantly reduces the amount of work required to satisfy each query relative to naive multi-hit, but is identical across node culling implementations. Differences in average frame time among culling techniques thus arise from the relative number of times the required callback functions are invoked.*