Hierarchical View-Frustum Culling for Z-buffer Rendering

FEL CTU in Prague

Aleš Koblížek

20. 5. 2019

Overview

1) Choose algorithm parameters
   - max triangles per leaf
   - traversal optimizations
2) Evaluate the algorithm

- using prerecorded routes
- 6 scenes

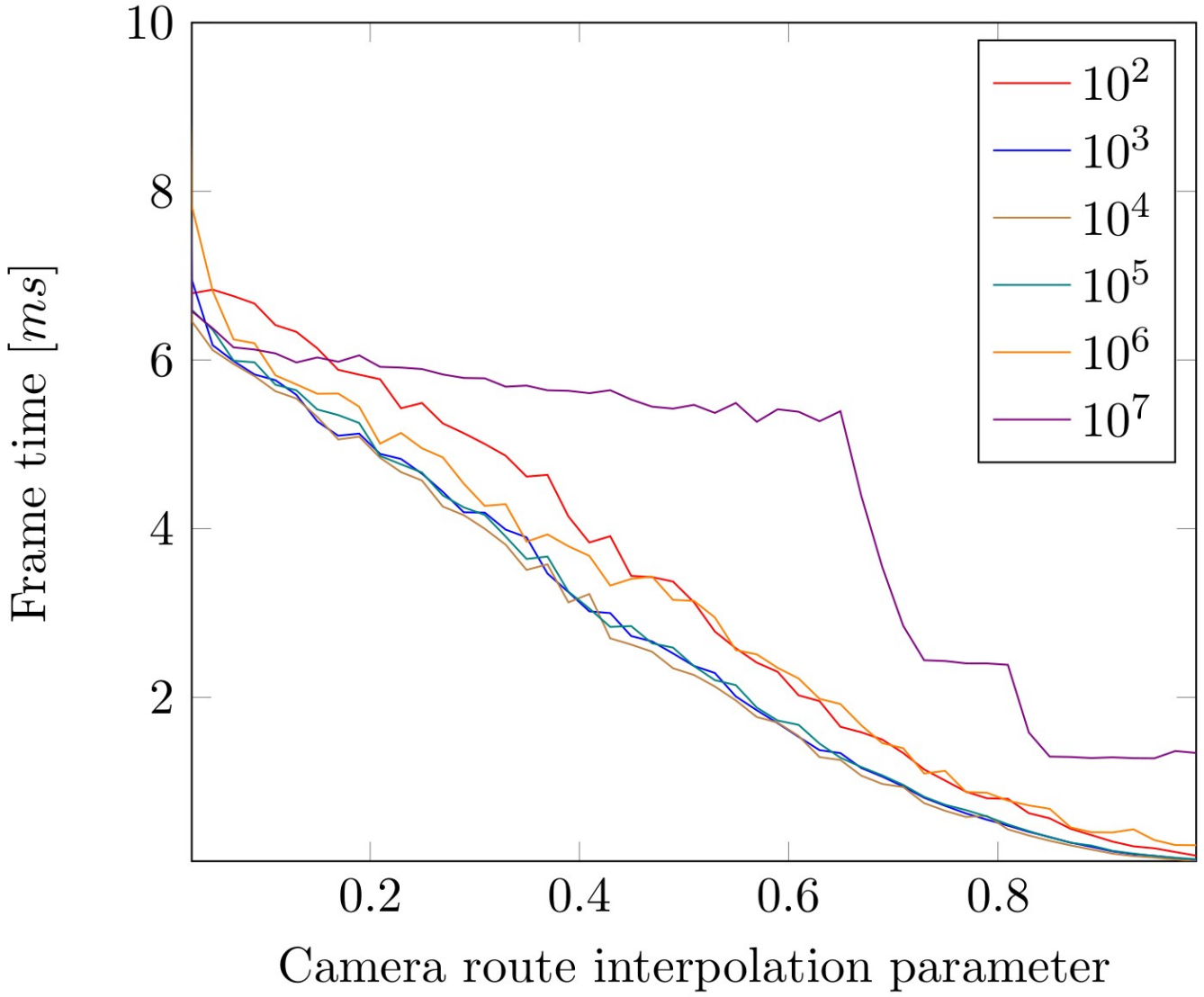

Figure 1: Frame time over the same camera route for various *MTPL* values, measured in scene 1

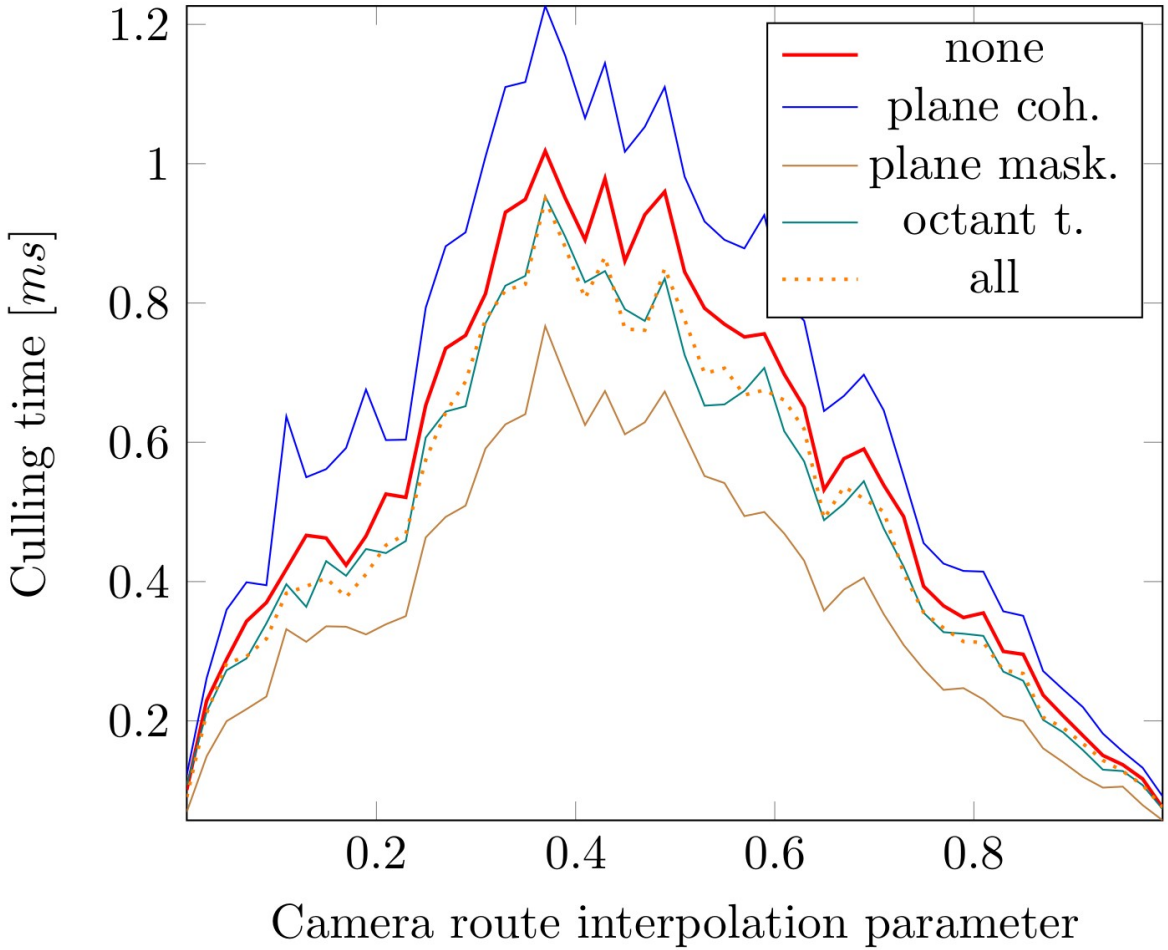

Figure 2: Comparison of culling optimization techniques – culling time along camera route in scene 1. Each of the techniques is measured individually and then all of them combined. Culling time without optimizations is also included.

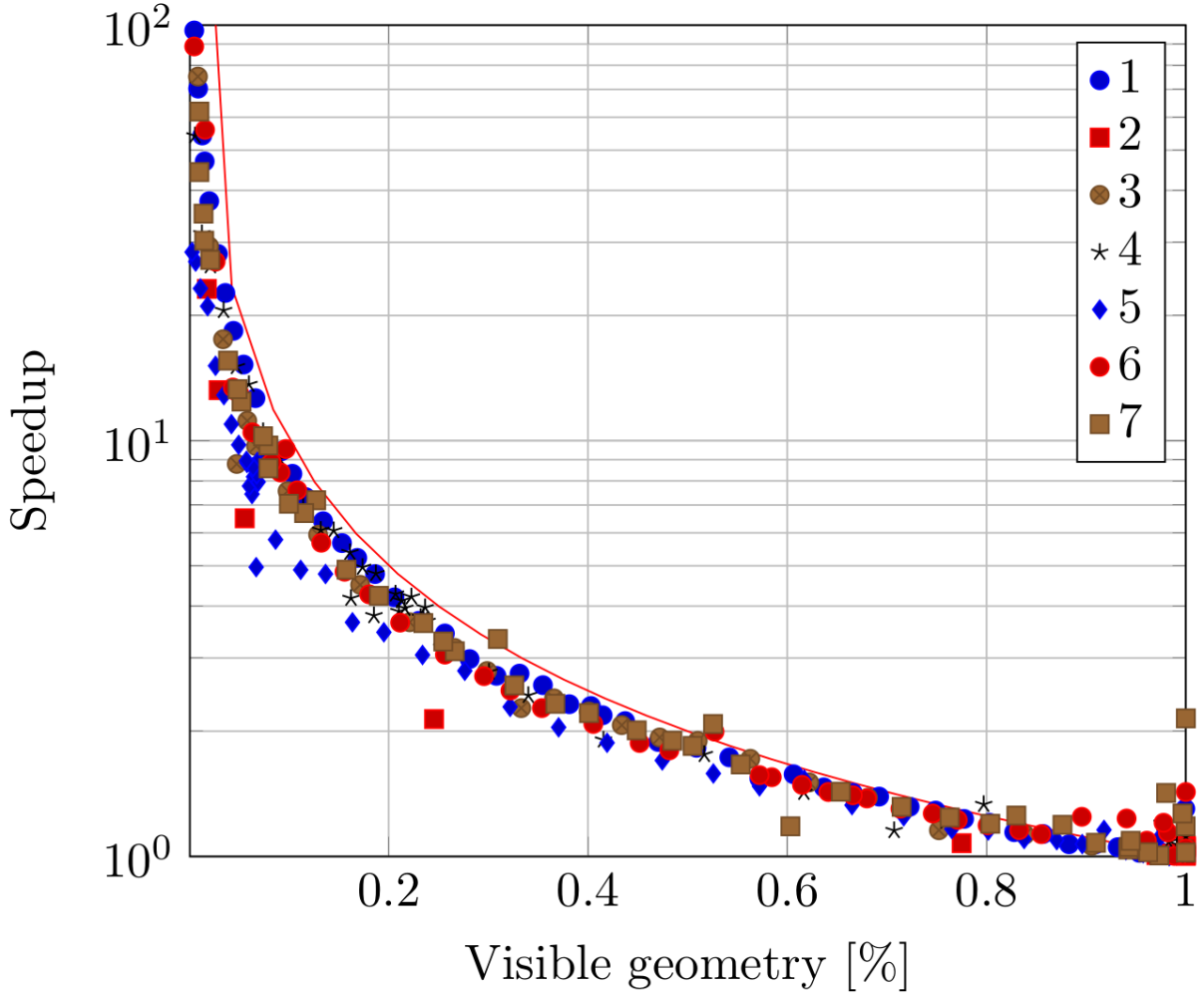

Figure 3: Frame time speedup obtained by frustum culling ($MTPL = 10^4$, *octant test*, *plane masking*) depending on the percentage of visible geometry. Measured on all scenes, numbered as in table 1 (according to the amount of geometry). The red line shows the optimal speedup: $\frac{1}{percentage\_of\_visible\_geometry}$.

Scene num.	Scene name	# tris.	# nodes
1	part_of_pompeii	28.0 m	8803
2	PowerPlantM	12.7 m	4623
3	asianDragon	7.2 m	2221
4	ten_blocks_in_pompeii	5.6 m	1783
5	City4M	4.7 m	1135
6	block_in_pompeii.high_lod	3.1 m	1137
7	vienna_cropped	0.9 m	283

Table 1: List of scenes used for testing. The given number of BVH nodes is for $MTPL = 10^4$.

CPU	Intel Xeon E3-1231 v3 @ 3.4 GHz, 8 MB cache, 4 cores, 8 threads
RAM	16 GB
GPU	Nvidia GeForce GTX 1070 Ti
Operating system	MS Windows 10
Compiler	MSVC 2017
Architecture	amd64
Compiler options	O2 (maximize speed), inline function expansion, enable intrinsic func.

Table 2: Hardware and software configuration used for measurements.