Parallel Quadtree Construction on Collections of Objects
SMI 2017 Manuscript number CAG-D-17-00114
Response to reviewers

We appreciate the insightful comments of the reviewers that resulted in what we believe to be a much improved version of the paper. Text in the paper that we have added or changed is indicated in blue. Below are responses to specific comments.

Reviewer #1

In Algorithm 1, it would be good to explicitly mark atomic operations, or say that they are not needed given certain memory ordering model.

Algorithm 1 requires no atomic operations. We have updated the paper to indicate this.

The title of 3.4.1 is confusing because it is not yet clear what a(s) is. Figure placement on page 4 is sub-optimal, it would be better to co-locate the figures with the portions of the text that reference them. Figure 4: panels e,f not discussed in caption.

We have addressed these issues in the paper, including an introduction to a(s) just before Section 3.4.1.

The prose in Section 3.3 is a bit too much tied to the code and reads awkwardly ("BCells", "FacetMap", ..).

We modified Sections 3.3 and 3.3.1 so that it reads more smoothly. We start with the conceptual ideas of the algorithm and then tie in a few specifics of the data structures.

Reviewer #3:

Another pass through the paper is needed to clean up many errors. Here a just a few.
36-38 almost repeated sentence or idea from prev paragraph
101 We are unaware of any GPU quad-tree construction methods that are fully adaptive and able to resolve the separation between objects.
132 Morton codes
152 separate
177 root to C

We addressed these errors in the paper.

It should be mentioned there has been lots of work in parallel construction of kd-trees, and other adaptive bounding volume hierarchies. In particular, the ray-tracing and rendering communities have put a lot research into developing algorithms that are tuned for GPUs.
http://rsim.cs.illinois.edu/denovo/Pubs/10-hpg-parkd.pdf
More recent work than Karras that uses Morton Codes.
http://dl.acm.org/citation.cfm?id=2566638

Thank you for the references. We have added them to the related work section along with commentary.

It is great that your source will be made available. It would be nice to have a little more detail on the mapping to CUDA. For example, line 168 mentions a CUDA kernel, but you don't precisely layout your kernel stages anyway in the paper.

Please see our new implementation section of the paper.

Lines 330 to 336 hit on a big issue: determining the depth a-priori to set the right number of integer bits for the Morton codes. Details can be show how this choice impacts performs. Choosing too many bits will result in wasted effort in pruning. Too few bits will cause more effort in refinements. Somewhere you should mention how you choose the number of bits.

We now address this in the implementation section of the paper.

The approach appears sound to me, but the paper could use more analysis and detail. For example, how does the number of bits used in the Morton codes impact the performance and result?

Again, the implementation section.

I'd love to see graphs as content complexity increases. Scalability plots with respect to the other methods would be interesting and go a long way to validating that this approach is good. How does it perform on GPUs with fewer cores? As the number of cores increases do you lose efficiency, or do you get linear scaling?

We have added Figure 10 which shows scalability as data complexity increases. We regret that we don't yet have a suitable scalability study with respect to the number of cores.

Reviewer #4

The previous works are also correctly discussed, I would suggest those two additional references: [VoxelPipe, Pantaleoni] and ["Octree-Based Sparse Voxelization Using The GPU Hardware Rasterizer", Crassin, Green] which are fully adaptive.

Thank you for the additional references. We have added the Crassin reference to the related work section and we reference VoxelPipe in section 3.3.1.

However, while this technique focus especially on the efficient implementation on a GPU parallel architecture, the description generally stays at an algorithmic level, and giving more indeep technical and implementation details would be useful. Technical elements such as memory organization of the data structures, scheduling and synchronization, atomicity of memory operations, or occupancy information, would improve the description as well as the reproducibility of the paper.

We have added kernel scheduling and scaling analysis to the paper. We use a single atomic for binary radix tree pruning, and have addressed it in section 3.2. Additional discussion on memory organization of data structures has been added to 3.3 and to our new implementation section.

Also, I don't believe that the time and memory complexity analysis presented in the paper provide any useful information to the reader in order to evaluate the technique or to compare it to other parallel implementation...

This is a helpful insight, and we have moved the complexity analysis to the appendix.

The result section also lacks some information about the API used, the toolkits abstracting some operations, etc. More generally about the results, it seems a bit difficult to really judge the efficiency improvements of the proposed technique over [Edwards et al] since the technique are very different, target very different architectures (GPU vs CPU) and were tested on arbitrary CPU and GPU without providing elements of comparison such as peak/avg flops and memory bandwidths.

We have addressed some of these questions in our new implementation section.

...For instance the choice of a point-sample based construction strategy could be better justified. I understand the motivation for using [Karas 2012] which is based on point samples, but does the complexity introduced by points for the polylines separation justify that?

We addressed this with new text in the first paragraph of Section 3.

The case of intersection of polylines from different objects should be discussed also. Is the approach robust to such intersections ? Fig 9 seems to suggest this is supported...

We address this in the new implementation section of the paper.

- The exact same paragraph appears repeated twice in the introduction section.
- Sec 1: l42 + l32 "course grid" -> "coarse grid"
- Fig 8: Quadtree (b) doesn't seem to be the same than (a)

We have addressed these issues.