Review of: "High-performance simplification of triangular surfaces using a GPU"

Muaaz Gul Awan[1]

1 Lawrence Berkeley National Laboratory

Potential competing interests: The author(s) declared that no potential competing interests exist.

Authors have presented a new simplification method for triangular surfaces keeping in mind the challenges of a GPU architecture. The new data structures and algorithm has been detailed well while also providing a good context of this study. Authors have claimed that their method provides matching quality as the existing state of the art solutions with faster compute times. Overall the paper has been written well and ideas and their implementations have been explained nicely but I have a few major concerns about the way authors have compared their results against other state of the art methods. I am listing these major concerns below:

- Authors claim that their method is more efficient based on the total runtime presented in Tables 2 and 3. However, the runtime presented for QSlim, MCS and the implementation from reference 12 have been obtained from their respective original papers. Timing for QSlim might still be acceptable as it is a CPU only code but to do a fair comparison (and therefore claim better performance) for GPU codes authors should have run other GPU based competing codes on the same hardware on which they tested their own code. In my opinion comparing different GPU application runtimes by running them on different hardwares is not fair comparison.
- Authors have not provided any details on how these timings were measured for their own code. Was this only the kernel runtime for the simplification kernel? Or the total runtime of all the kernels? Does this time also include data transfer time (CPU to GPU) and GPU setup time?
- The target vertices in Table 2 do not match with Table 1 in reference 31, I am not a domain expert but shouldn’t those two be same for a fair comparison of quality and performance?
- It would be good to have a figure explaining the overall workflow on CPU-GPU system. Right now it is very confusing, like how many kernels there are in total and what is the overall workflow i.e. when is the data transferred from CPU to GPU and copied back. Providing such a figure would also make it simple for the authors to point out the regions they timed for table 2 and 3.
- I do not see any code availability details.

I also have some minor concerns:

- It is mentioned that the machine that author’s had access to had a kernel runtime limitation where a kernel could not run for longer than 8 seconds before being interrupted by the driver, for that reason authors had to do several runs of a kernel for a single run of complete simplification process. However, most of the runtimes reported in Tables 2 and 3 are well below the 8 second limit. Was the kernel interruption and rerun only required for a select dataset? It would be good if authors clarified this.

- If authors use a dedicated GPU for computational purposes, they might not see the 8 second time limit on kernel runs hence simplifying the development.
- Authors have not mentioned the version of CUDA toolkit used for development.
- It would be good to know more details of the kernel implementations i.e. details of grid size, threads per block launched etc.