Project Proposal - EEC289Q

Parallel Recursive Spoke Darts for Delaunay/Voronoi Meshing

Ahmed H. Mahmoud, Muhammad Awad

Abstract

We propose to implement the Recursive Spoke Darts (RSD) algorithm; a local hyper-plane sampling method for 3D Delaunay and Voronoi meshing presented in [4]. The algorithm has two overlapping objectives. Through reliance on simple local operations (recursive line-hyperplane trimming), the authors claims to present the first exact Delaunay/Voronoi meshing algorithm that breaks the curse of dimensionality by improving the scalability through eliminating communication between compute entities (processors, threads, blocks, etc). However, the algorithm was only implemented on MPI settings where it improves over the state-of-the-art. The improvement comes in terms of speedup and the ability to construct the mesh exactly in higher dimensions where other methods give approximate solution, prohibitively slow or fail completely. In this project, we would like to scratch the surface of this algorithm when mapped into the GPU by confining ourselves to 3D space. This will give a concert idea on the challenges of implementing it on higher dimensions. The algorithm presents many interesting sub-problems due to the inherent irregularity of the problem which troubles the memory accessing pattern, load balance and control flow.

Motivation

Delaunay Triangulation (DT) of a point set has important applications in many fields including data visualization and imaging, terrain modeling, finite element mesh generation, surface reconstruction, and structural networking for arbitrary point sets. This is due to its nice geometric properties and desirable qualities. For example, in $\mathbb{R}^3$, DT minimizes the maximum radius of the minimum containment sphere of the tetrahedra which guarantees the convergence of the finite element solution. Accelerating the computation of DT is a necessity with the rapid increase in the problem size from thousands of points to millions and even billions [8].

There has been a decent amount of work devoted to parallelize the DT computation most of which is an adaption of one of the serial algorithms. Examples include parallelize Dwyer's divide and conquer algorithm [5, 3, 6, 8], incremental/point insertion algorithm [2, 1] and flipping algorithms [7]. Even though all these implementations have achieved good speedup over the serial counterparts, we hypothesize that in order to further accelerate the computation, one should start with an algorithm that is designed for parallel execution. RSD has the potential to be the first DT algorithm that is deigned with parallel mindset. RSD abandons the dependence on the empty sphere principle which gives it the foundation needed for scalable consistent meshing. Additionally, the simple operations the RSD algorithm relies on make it more favorable for GPU settings.

Milestones

- **Week 1 (26 February - 4 March):** Preparation of input point clouds, k-d tree for nearest neighbor search queries, initial implementation using one thread, and validating the output.
- **Week 2 (5 March - 11 March):** Extending the one thread implementation to multiple blocks and multiple threads.
- **Week 3 (12 March - 18 March):** Optimization of memory accesses and caching into shared memory, comparing against other parallel algorithms for 3D Delaunay meshing.

Deliverables

1. We will deliver the code that implements RSD in 3D with the following inputs and outputs:
   - Input: sample points in 3D space
   - Output: Delaunay-based tetrahedral meshing of the points based on RSD algorithm
2. Evaluation:
   - Correctness (validating the correctness of our Delaunay mesh using TetGen [9]).
   - Runtime comparison with (gDel3D: GPU accelerated algorithm for 3D Delaunay triangulation [2]).

We will give the necessary directions for validation with TetGen.

References

[1] Batista, V. H., Millman, D. L., Pion, S., and Singler, J. (2010). Parallel geometric algorithms for multi-core computers. *Computational Geometry*, 43(8):663–677.

[2] Cao, T.-T., Nanjappa, A., Gao, M., and Tan, T.-S. (2014). A gpu accelerated algorithm for 3d delaunay triangulation. In *Proceedings of the 18th meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 47–54. ACM.

[3] Cignoni, P., Montani, C., and Scopigno, R. (1998). Dewall: A fast divide and conquer delaunay triangulation algorithm in ed. *Computer-Aided Design*, 30(5):333–341.

[4] Ebeida, M. S. and Rushdi, A. A. (2016). Recursive spoke darts: Local hyperplane sampling for delaunay and voronoi meshing in arbitrary dimensions. *Procedia Engineering*, 163:110 – 122. 25th International Meshing Roundtable.

[5] Fuetterling, V., Lojewski, C., and Pfreundt, F.-J. (2014). High-performance delaunay triangulation for many-core computers.

[6] Lee, S., Park, C.-I., and Park, C.-M. (1997). An improved parallel algorithm for delaunay triangulation on distributed memory parallel computers. In *Proceedings. Advances in Parallel and Distributed Computing*, pages 131–138.

[7] Liparulo, L., Proietti, A., and Panella, M. (2015). Fuzzy clustering using the convex hull as geometrical model. *Advances in Fuzzy Systems*, 2015:6.

[8] Lo, S. (2015). 3d delaunay triangulation of 1 billion points on a pc. *Finite Elements in Analysis and Design*, 102:65–73.

[9] Si, H. (2015). Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software (TOMS)*, 41(2):11.