External Memory Techniques for Isosurface Extraction in Scientific Visualization*

(Abstract)

Yi-Jen Chiang[†] Cláudio T. Silva[‡]

Isosurface extraction is one of the most effective and powerful techniques for the investigation of volume datasets in scientific visualization. Previous isosurface techniques are all main-memory algorithms, often not applicable to large scientific visualization applications. In this paper we survey our recent work [2, 3, 4] that gives the first external memory techniques for isosurface extraction.

In [2] we give *I/O-filter*, the first I/O-optimal technique for isosurface extraction. We keep the entire dataset in disk, and use the I/O-optimal interval tree of Arge and Vitter [1] as an indexing structure, which enables us to find and fetch only those *active cells* intersected by the isosurface from disk to main memory, in optimal I/O cost. The isosurface itself can then be generated from those active cells. We give the first implementation of the I/O interval tree (where the *corner structure* is not implemented, which may result in non-optimal disk space and non-optimal query I/O cost in the worst case), and also implement our method as an *I/O filter* for the isosurface extraction routine of Vtk [6, 7] (which is one of the currently best visualization packages). The experiments show that the isosurface queries are faster than Vtk by a factor of two orders of magnitude for datasets larger than main memory. In fact, the *search phase* is no longer a bottleneck, and the performance is *independent* of the main memory available. Also, the preprocessing is performed only *once* to build the indexing structure in disk, and later on there is no start-up cost for running the query process. The major drawback is the overhead in disk scratch space and the preprocessing time necessary to build the search structure, and of the disk space needed to hold the data structure.

In [3], we give the second version of *I/O-filter*, by replacing the I/O interval tree [1] with the metablock tree of Kanellakis *et al.* [5]. We give the first implementation of the metablock tree (where the corner structure is not implemented to reduce the disk space; this may result in non-optimal query I/O cost in the worst case). While keeping the query time the same as in [2], the tree construction time, the disk space and the disk scratch space are all improved.

In [4], at the cost of slightly increasing the query time, we greatly improve all the other cost measures. In the previous methods [2, 3], the *direct vertex information* is duplicated many times; in [4], we avoid such duplications by employing a *two-level indexing* scheme. We use a new *meta-cell* technique and a new I/O-optimal indexing data structure (the *binary-blocked I/O interval tree*) that is simpler and more space-efficient in practice (where the corner structure is not implemented, which may result in non-optimal I/O cost for the *stabbing queries* in the worst case). Rather than

---

*Journal version to appear in *Proceedings of the DIMACS Workshop on External Memory Algorithms and/or Visualization (AMS-DIMACS Book Series)*; preprint available at `http://cis.poly.edu/chiang/iso-survey.ps.gz`.

[†]Department of Computer and Information Science, Polytechnic University, 6 MetroTech Center, Brooklyn, NY 11201; yjc@poly.edu. Supported in part by NSF Grant DMS-9312098 and by Sandia National Labs.

[‡]P.O. Box 704, IBM T. J. Watson Research Center, Yorktown Heights, NY 10598; csilva@watson.ibm.com. Partially supported by Sandia National Labs and the Dept. of Energy Mathematics, Information, and Computer Science Office, and by NSF Grant CDA-9626370.

fetching only the active cells into main memory as in *I/O-filter* [2, 3], this method fetches the set of *active meta-cells*, which is a *superset* of all active cells. While the query time is still at least one order of magnitude faster than Vtk, the disk space is reduced from 7.2–7.7 times the original dataset size to 1.1–1.5 times, and the disk scratch space is reduced from 10–16 times to less than 2 times. Also, instead of being a single-cost indexing approach, the method exhibits a smooth trade-off between disk space and query time.

References

[1] L. Arge and J. S. Vitter. Optimal interval management in external memory. In *Proc. IEEE Foundations of Comp. Sci.*, pages 560–569, 1996.

[2] Y.-J. Chiang and C. T. Silva. I/O optimal isosurface extraction. In *Proc. IEEE Visualization*, pages 293–300, 1997.

[3] Y.-J. Chiang and C. T. Silva. Isosurface extraction in large scientific visualization applications using the I/O-filter technique. Submitted for a journal publication. Preprint: Technical Report, University at Stony Brook, 1997.

[4] Y.-J. Chiang, C. T. Silva, and W. J. Schroeder. Interactive out-of-core isosurface extraction. In *Proc. IEEE Visualization*, 1998 (to appear).

[5] P. C. Kanellakis, S. Ramaswamy, D. E. Vengroff, and J. S. Vitter. Indexing for data models with constraints and classes. In *Proc. ACM Symp. on Principles of Database Sys.*, pages 233–243, 1993.

[6] W. Schroeder, K. Martin, and W. Lorensen. The design and implementation of an object-oriented toolkit for 3D graphics and visualization. In *IEEE Visualization '96*, October 1996.

[7] W. Schroeder, K. Martin, and W. Lorensen. *The Visualization Toolkit.* Prentice-Hall, 1996.