Information-Aware 2^n-Tree for Efficient Out-of-Core Indexing of Very Large Multidimensional Volumetric Data

Jusub Kim
Institute for Advanced Computer Studies
Department of Electrical and Computer Engineering
University of Maryland, College Park, MD 20742
E-mail: jusub@umd.edu

Joseph JaJa
Institute for Advanced Computer Studies
Department of Electrical and Computer Engineering
University of Maryland, College Park, MD 20742
E-mail: joseph@umiacs.umd.edu

***Abstract*— We discuss a new efficient out-of-core multidimensional indexing structure, information-aware 2^n-tree, for indexing very large multidimensional volumetric data. Building a series of (n-1)-Dimensional indexing structures on n-Dimensional data causes a scalability problem in the situation of continually growing resolution in every dimension. However, building a single n-Dimensional indexing structure can cause an indexing effectiveness problem compared to the former case. The information-aware 2^n-tree is an effort to maximize the indexing structure efficiency by ensuring that the subdivision of space have as similar coherence as possible along each dimension. It is particularly useful when data distribution along each dimension constantly shows a different degree of coherence from each other dimension. Our preliminary results show that our new tree can achieve higher indexing structure efficiency than previous methods.**

I. Introduction

As the speed of processors continues to improve, researchers are performing large scale scientific simulations to study very complex phenomena at increasingly finer resolution scales. Such studies have resulted in the generation of datasets that are characterized by their very large sizes ranging from hundreds of gigabytes to tens of terabytes, thereby generating an imperative need for new interactive visualization capabilities. Consider for example the fundamental mixing process of the Richtmyer-Meshkov instability from the ASCI team at the Lawrence Livermore National Labs [1]. The simulation produced about 2.1 terabytes of data, which shows the characteristic development of bubbles and spikes and their subsequent merger and break-up over 270 time steps. The resolution of each time step is $2,048 \times 2,048 \times 1,920$ ($\sim$8 GB). Such high resolution simulations allow elucidation of fine scale physics.

A typical way of visualizing such a large multidimensional volumetric data set is to first reduce the dimension of the data set using techniques such as slicing and then to render the result using one of the isosurface or volume rendering techniques [2]. Slicing is a very useful tool because it removes or reduces occlusion problems in visualizing such a multidimensional volumetric data set and it enables fast visual exploration of such a large data set.

In order to efficiently handle the process, we need an efficient out-of-core indexing structure because such a data set very often does not fit in main memory. A typical way of building indexing structures in the case of time-varying volumetric data such as the one described above is to build a separate indexing structure on each time step of the data set. For example, Sutton and Hansen's temporal branch-on-need structure (T-BON) [3] is the most representative. Their strategy is to build an out-of-core version of Branch-On-Need-Octree (BONO) [4], in which each leaf node is of disk page size, for each time step and to store general common infrastructure of the trees in a single file. However, the way of building (n-1)-dimensional trees along one particular dimension such as the way that the T-BON adopts unfortunately results in the size increase linearly with the resolution size at the particular dimension (the number of time steps in the case of T-BON) because it does not exploit any type of possible coherence across the particular dimension. This lack of scalability becomes more problematic as we generate higher and higher resolution data in every dimension including the time dimension.

In this paper, we present a new efficient out-of-core multidimensional indexing structure, information-aware (IA) 2^n-tree. The strategy is to basically build a n-Dimensional indexing structure on n-Dimensional data because it can exploit the coherence across all n dimensions and thus lead to compact size, addressing the scalability problem. However, we need to also consider indexing effectiveness as well as the scalability. The effectiveness of out-of-core indexing can be defined by how much data is actually what we needed from the loaded data because the finest indexed object is not an individual voxel, but a group of data which is of disk page size. Thus, using the new structure, we seek to increase the ratio of indexing effectiveness to indexing structure size, which we define as *indexing structure efficiency*.

The key feature of the IA 2^n-tree is to provide higher indexing structure efficiency than the previous 2^n tree or $n \times 2^{n-1}$ trees. While a typical 2^n-tree recursively subdivides the n-Dimensional volumetric data into 2^n subvolumes based only on the volume extent, our 2^n-tree determines its dimension ra-

tios of a subvolume based on the information embedded in the data so that the subvolumes can contain as similar coherence as possible along each dimension, resulting in higher indexing effectiveness. Our method is particularly useful when data distribution along each dimension constantly shows a different degree of coherence from each other dimension.

We used the IA 2^n-tree in 4-D (time-varying 3-D) data sets to retrieve necessary data, given slicing and isosurface queries. We compared our tree with previous major indexing structures used for the same purpose, and achieved higher indexing efficiency.

The rest of this paper is organized as follows. We discuss major related out-of-core techniques in Section 2 and describe our new indexing structure in Section 3. A summary of our experimental results is given in Section 4 and we conclude in Section 5.

II. Previous Out-of-Core Techniques

Disks have several orders of magnitude longer access time than random access main memory because of their electromechanical components. A single disk access reads or writes a block of contiguous data at once. The performance of an out-of-core algorithm [5] is often dominated by the number of I/O operations, each involving the reading or writing of disk blocks. Hence designing an efficient out-of-core visualization algorithm requires a careful attention to reducing the number of I/O operations and organizing disk accesses in such a way that active data blocks are moved in large contiguous chunks to main memory.

During the past few years, a number of out-of-core techniques have appeared in the literature to handle several visualization problems. Cox and Ellsworth [6] show that application-controlled paging and data loading in a unit of subcube with the ability of controlling the page size can lead to better performance in out-of-core visualization. Out-of-core isosurface extraction algorithms for static datasets are reported in [7], [8], [9]. Of more interest to us is the previous work on out-of-core algorithms dealing with time-varying data. Chiang [10] proposes an out-of-core isosurface extraction algorithm based on a time hierarchy for irregular grids. This hierarchy uses the Binary-Blocked-I/O interval trees (BBIO) [7] as secondary structures to support I/O optimal interval searches. However, it can not efficiently support the slicing process since cells are organized by their interval values. Sutton and Hansen [3] introduce the Temporal Branch-on-Need-octree (T-BON) to extract isosurfaces for each time step separately. Another related work is the PHOT data structure developed in [11]. While it achieves asymptotically optimal internal memory search, its size is substantially large. Silva et al. [12] provide a good survey on out-of-core algorithms for scientific visualization and computer graphics.

III. Information-Aware 2^n-trees

Information-Aware 2^n-trees (IA 2^n-trees) are basically 2^n-trees (e.g. quadtrees for 2-D and octrees for 3-D [13]) for n-dimensional space. However, it is different in terms of how it decides the extent ratios of a subvolume when multi dimensions are integrated into one hierarchical indexing structure. The coherence information along each dimension is extracted and used for the decision so that each subvolume contains as similar coherence as possible along each dimension.

A. Dimension Integration

We present an entropy-based dimension integration technique. Entropy [14] is a numerical measure of the uncertainty of the outcome for an event x, given by $H(x) = -\sum_{i=1}^{n} p_i \log_2 p_i$, where x is a random variable, n is the number of possible states of x, and p_i is the probability of x being in state i. This measure indicates how much information is contained in observing x. The more the variability of x, the more unpredictable x is, and the higher the entropy. For example, consider a series of scalar field values for a voxel v over the time dimension. The temporal entropy of v indicates the degree of variability in the series. Therefore, high entropy implies high information content, and thus more resources are required to store the series. Note that the entropy is maximized when all the probabilities p_i are equal.

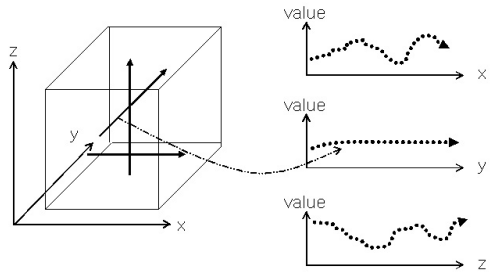

Fig. 1. Entropy estimation in each dimension. Note that the y dimension has almost zero entropy in this example.

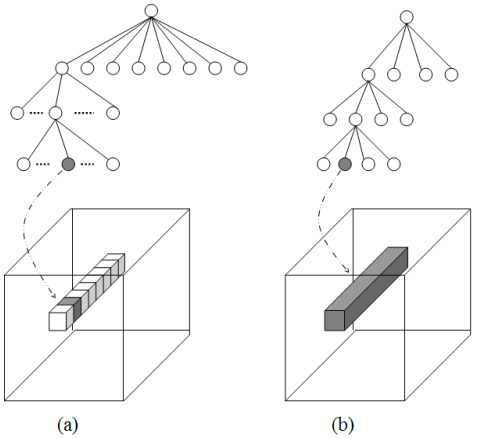

Fig. 2. Different metacell sizes and corresponding hierarchical indexing structures for the data of Figure 1: (a) standard metacell; (b) information-aware metacell.

We use the entropy notion to determine the relative sizes of the extents of a metacell, which is a subcube corresponding to a leaf node in the trees. Higher entropy of a dimension relative

to the other dimensions implies that this dimension needs to be split at finer scales than the other dimensions. For example, if a temporal entropy is twice as much as the spatial entropy, we design the metacell to be of size $s \times s \times s \times \frac{s}{2}$ ($x \times y \times z \times t$), where s is the size of the spatial dimension of the metacell.

Figures 1 and 2 show how this entropy-based dimension integration leads to an indexing structure for the 3-D case. Figure 1 shows an extreme case in which the values along the y dimension remain almost constant over all possible (x, z) values (that is, the entropy of y is almost zero) while each of the x and z dimensions has some degree of variability. The metacell size and the corresponding hierarchical indexing structure will be designed as shown in Figure 2 (b), that is, it has a quadtree structure unlike the standard octree of Figure 2 (a) in which the metacell has the same size in each dimension.

To estimate the ratios of the entropy values among n dimensions, we randomly select a set of n-Dimensional subvolumes and for each subvolume, obtain the ratios by simply computing each entropy value along each dimension. The ratios are averaged and globally applied in building indexing structures. In computing the entropy values, if the number of the possible scalar field values is large (as in the case of floating point values), we first quantize the original values into n values using a non-uniform quantizer such as the lloyd-max quantizer.

Even though it can apply to general cases, we are primarily concerned about establishing the relationship between spatial and temporal dimensions because there is usually constant difference in the coherence of data values between the two different types of dimensions. Thus we compute the *spatio-temporal entropy ratio* defined as the ratio of the average spatial entropy to the temporal entropy.

We note that in general a time series will consist of a number of temporal domains during which the spatio-temporal entropy ratio can be different. Our general strategy is to decompose the time series into a set of temporal regions, each of which will be characterized by its spatio-temporal entropy ratio. Hence we build a separate IA-Octree for each temporal region.

B. Indexing Structures

We make use of the entropy ratios for the purpose of guiding the branching of the tree and ultimately adjusting the size of metacells by dividing the dimension of high entropy more finely and that of low entropy more coarsely. It is simply carried out by multiplying the original size of each dimension by its entropy value, which becomes the 'effective' size of the dimension, and then using the 'effective' size instead of the original size in branching of the tree. In addition to that, We adopt the Branch-On-Need strategy [4] by delaying the branching of each dimension until it is absolutely necessary.

For efficient isosurface rendering, each tree node contains the minimum and maximum values of the scalar fields in the region represented by the node. The size of the tree can be reduced by pruning nodes in which the minimum and maximum values are the same because they do not contribute to isosurface extraction.

IV. Experimental Results

We compared the indexing structure efficiency of our IA 2^n-tree with a typical 2^n-tree and also the T-BON scheme [3], which is one of the two most popular schemes for time-varying isosurface rendering and can also handle slicing queries. For evaluation, we consider two large time-varying volumetric data sets: the Richtmyer-Meshkov data set for time steps $100-139$, each down-sampled by two along each spatial dimension, and the Five Jets data set [15] consisting of 2000 time steps. Each time step of the Richtmyer-Meshkov data set involves a $1024 \times 1024 \times 960$ grid with one-byte scalar values resulting in total 40 GB data set. The Five Jets data set consists of $128 \times 128 \times 128$ grid with 4-bytes floating point values resulting in total 16 GB.

We ran all the tests on a single Linux machine which has dual 3.0 GHz Xeon processors with ∼50 MB/s maximum disk I/O transfer rate. In all our experiments, we made use of only one of the two processors. Also, we used a simple buffer management system in order to control disk I/O.

	Ratio (IA-2^n/ 2^n)
Loaded Data	1.02
Effectiveness	0.97
Disk Access Time	1.01
Tree Traversal	0.93
Total Time	1.00

TABLE I

Query performance comparison between IA 2^n-tree and 2^n-tree for the Richtmyer-Meshkov data set. The results are the average values over various types of slicing and different isovalues.

We first compare the IA 2^n-tree with a 2^n-tree for the Richtmyer-Meshkov data set. Using the entropy measure, we obtained a spatio-temporal entropy ratio equal to 1.5 over the time steps $100 - 139$ for the data set, resulting in 30% less indexing structure size than the 2^n-tree due to coarser subdivision along the temporal dimension. However, Table I shows that it experiences only 3% indexing effectiveness reduction. Note that the tree traversal time decreases because the number of nodes that the tree has to visit decreases. Overall, it results in 1.4 times better indexing structure efficiency.

Now we also compare the IA 2^n-tree with the T-BON scheme for both the Richtmyer-Meshkov and the Jet data set to show how much redundant information is retained in a series of (n-1)-Dimensional indexing structures and how that affects performance. The size of IA 2^n-tree is only about 1/9 of the T-BON structure for the Richtmyer-Meshkov data set. For the Jet data set, we arbitrarily divided the temporal domain of the Five Jets data set into four time regions (see Figure 3) having respectively the spatio-temporal entropy ratios of 0.5, 1, 3, and 4. We separately built our tree on each time region. The total size of the four IA 2^n-trees is only about 1/8 of the T-BON structure for the Jet data. The space

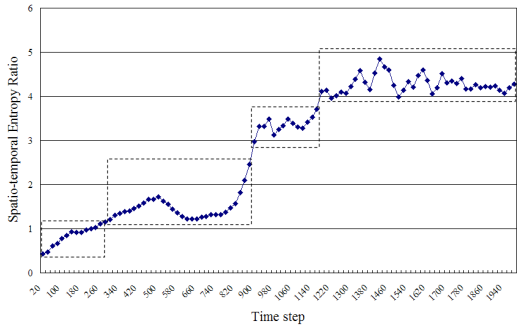

Fig. 3. Spatio-temporal entropy ratios computed at uniformly selected 100 reference time steps among the 2000 time steps in the Five Jets data. Each dashed box corresponds to a time region.

	Richtmyer-Meshkov	Jet
	Ratio (IA-2^n / T-BON)	Ratio (IA-2^n / T-BON)
Loaded Data	1.10	1.05
Effectiveness	0.91	0.94
Disk Access Time	1.15	1.09
Tree Traversal	0.71	0.70
Total Time	0.98	0.98

TABLE II

QUERY PERFORMANCE COMPARISON BETWEEN IA 2^n-TREE AND T-BON FOR THE RICHTMYER-MESHKOV AND THE JET DATA SET. THE RESULTS ARE THE AVERAGE VALUES OVER VARIOUS TYPES OF SLICING AND DIFFERENT ISOVALUES.

reduction mainly comes from using temporal coherence in the indexing structure. However, Table II shows that the indexing effectiveness reduction is only 9% and 6% respectively for each of the two data sets. It results in about 8 times better indexing structure efficiency.

The experimental results show that we can even obtain slightly better timing results. It is because the effect of the increased data transfer due to the reduced effectiveness can be mitigated by memory cache effect, but there is no way that the longer tree traversal time of the larger T-BON structure and 2^n-tree can be mitigated in the course of successive queries.

V. CONCLUSION

We introduced a new indexing structure called Information-Aware 2^n-trees. Building a series of (n-1)-Dimensional indexing structures causes a scalability problem in the current situation of continually growing resolution in every dimension. However, building a single n-Dimensional indexing structure can cause an indexing effectiveness problem compared to the former case. The Information-Aware 2^n-tree is an effort to maximize the indexing structure efficiency by ensuring that the subdivision of space have as similar coherence as possible along each dimension.

Our future work plan includes evaluating the goodness of the entropy measure in comparison to other measures and finding out more adaptive way of applying the coherence difference in the subdivision as well as more reasonable way of decomposing the time series.

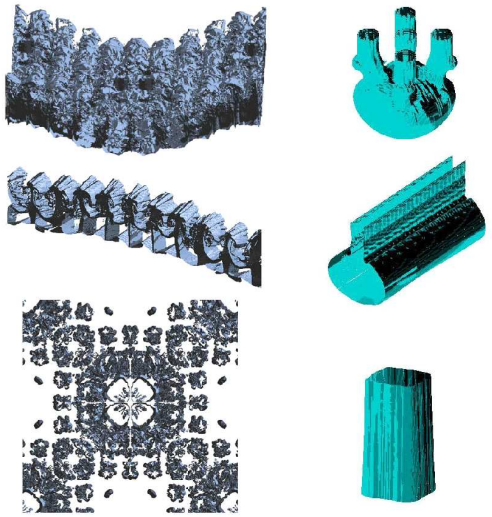

Fig. 4. Sample result images of the slicing + isosurface rendering. The left shows the Richtmyer-Meshkov data set and the right for the Jet data set cut by temporal-, y-, and z-slicing from top to bottom.

REFERENCES

[1] *The ASCI Turbulence project*, Lawrence Livermore National Laboratory, http://www.llnl.gov/CASC/asciturb.

[2] C. Hansen and C. Johnson, *The visualization handbook*. Elsevier Butterworth-Heinemann, 2005.

[3] P. M. Sutton and C. D. Hansen, "Accelerated isosurface extraction in time-varying fields," *IEEE Transactions on Visualization and Computer Graphics*, vol. 6, no. 2, pp. 98–107, Apr 2000.

[4] J. Wilhelms and A. V. Gelder, "Octrees for faster isosurface generation," *ACM Transactions on Graphics*, vol. 11, no. 3, pp. 201–227, Jul 1992.

[5] J. Vitter, "External memory algorithms and data structures: Dealing with massive data." *ACM Computing Surveys*, March 2000.

[6] M. Cox and D. Ellsworth, "Application-Controlled Demand Paging for Out-of-Core Visualization," *Proceedings of the 8th conference on Visualization*, 1997.

[7] Y.-J. Chiang, C. T. Silva, and W. J. Schroeder, "Interactive out-of-core isosurface extraction," in *Proceedings of IEEE Visualization*, 1998, pp. 167–174.

[8] C. L. Bajaj, V. Pascucci, D. Thompson, and X. Y. Zhang, "Parallel accelerated isocontouring for out-of-core visualization," in *Proceedings of the IEEE symposium on Parallel visualization and graphics*, 1999, pp. 97–104.

[9] X. Zhang, C. Bajaj, and V. Ramachandran, "Parallel and out-of-core view-dependent isocontour visualization using random data distribution," in *Proceedings of the IEEE symposium on Parallel visualization and graphics*, 2002, pp. 9–17.

[10] Y.-J. Chiang, "Out-of-core isosurface extraction of time-varying fields over irregular grids," in *Proceedings of IEEE Visualization*, 2003, pp. 29–36.

[11] Q. Shi and J. JaJa, *Efficient Isosurface extraction for large scale time-varying data using the persistent hyperoctree (PHOT)*. Unpublished, 2005.

[12] C. Silva, Y. Chiang, J. El-Sana, and P. Lindstrom, "Out-of-core algorithms for scientific visualization and computer graphics," *IEEE Visualization Course Notes*, 2002.

[13] H. Samet, *The design and analysis of spatial data structures*. Addison-Wesley, 1990.

[14] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley, 1991.

[15] Time-Varying Volume Data Repository, The Five Jets dataset, http://www.cs.ucdavis.edu/∼ma/ITR/tvdr.html.