Massive Data Visualization: A Survey

Kenneth I. Joy[1]

Institute for Data Analysis and Visualization
University of California, Davis
kijoy@ucdavis.edu

Summary. Today's scientific and engineering problems require a different approach to address the massive data problems in organization, storage, transmission, visualization, exploration, and analysis. Visual techniques for data exploration are now common in many scientific, engineering, and business applications. However, the massive amount of data collected through simulation, collection and logging is inhibiting the use of conventional visualization methods. We need to discover new visualization methods that allow us to explore the massive multi-dimensional time-varying information streams and turn overwhelming tasks into opportunities for discovery and analysis.

1 Introduction

Scale is the grand challenge of visualization! We face a situation where the problem analyst is continually overwhelmed with massive amounts of information from multiple sources, where the relevant information content exists in a few "nuggets." The visualization challenge is to create new methods that allow the analyst to visually examine this massive, multi-dimensional, multi-source, time-varying information stream and make decisions in a time critical matter.

The definition of "massive" is a moving target, because it changes over time as computational resources and algorithmic methods improve. We will apply the term "massive" to information streams that overwhelm some critical computation or display resource necessary for analysis. The most common application of this term is to information streams that do not fit into local disk or main memory, although many researchers apply the term to data that cannot be displayed on a "single display device."

Various approaches have been used to address the problems of scale. There is a continual effort to construct computer systems that can store and process massive amounts of data, and design display systems that can display data at ever increasing resolution. However, our ability to collect and generate data is increasing at a faster rate than the projected increase in computational and display power, and we wish to focus on methods that can address the problems of scale independent of computer system and display advances. These methods can be classified as follows:

- Compression and simplification – Reduce the size of the information stream by utilizing data compression methods or by simplifying the data to remove redundant or superfluous items.
- Multiresolution methods – Design multiresolution hierarchies to represent the information at varying resolutions, allowing the user (or the application) to dictate the resolution displayed.
- Memory external methods – Create innovative methods that store the data externally and develop methods that access relevant data subject to the interactivity required by the user.
- Abstraction – Create new visual metaphors that display data in new ways.

Many areas outside of traditional visualization are also critical in meeting the challenge of scale, including an understanding of application areas. Visualization research must reach out to encompass issues from numerical methods, data analysis, software engineering, database methods, networking, image processing, cognitive and perceptual psychology, human-computer interaction, and machine learning to address the problems before us. We must also develop evaluation techniques to "measure" scalability so new tools can be analyzed.

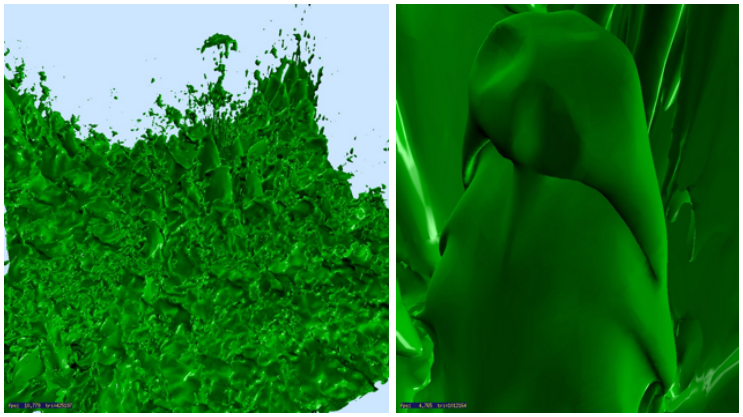
This paper attempts a partial survey of the methods used to address scale in visualization systems. We do not attempt to reproduce existing surveys for compression [**?**, **?**, **?**], but focus on simplification, multiresolution techniques, memory-external methods, and visual scalability through abstraction.

2 Driving Problems

Our interest in massive data is driven by the many domains in which people actively collect data. Simulation applications are driven by problems in computational fluid dynamics, engineering analysis, high-energy physics, and microprocessor design. Technical advances are now producing massive time-varying data in medical, biomedical and biological imaging applications. Bioinformatics data collections (*e.g.,* The Human Genome Project) will provide massive multi-dimensional data to be analyzed. The Sloan Digital Sky Survey is collecting eight terabytes per year of astronomical data, and satellites turned towards the Earth collect vast amounts of image data as well. The dramatically decreasing cost and increasing capabilities of sensors have led to an explosion in the collection of real-world data. Dynamic processes, arising in business or telecommunication, generate massive streams of sensor data, web click streams, network traffic logs, or credit card transactions.

Consider the various data types that possibly affect the domain examined by the problem analyst. These could include the following:

- Textual data – Massive textual data from documents, speeches, e-mails, or web pages now influence the problem domain. This data can be truly massive, contain billions of items per day, and much of it must be analyzed in a time-critical manner. [**?**, **?**, **?**]
- Simulation Data – Terascale simulations are now producing tens of terabytes of output for several-day runs on the largest computer systems. As an example, the Gordon-Bell-Prize-winning simulation of a Richtmyer-Meshkov instability in a shock-tube experiment [**?**], produces isosurfaces of the mixing interface



(a) (b)

Fig. 1. Scientific Simulations can produce massive isosurface data using conventional techniques (courtesy: M. Ducaineau)

with 460 million unstructured triangles using conventional extraction methods. The pipeline necessary to visualize these massive data sets is described well by Duchaineau *et al.* [**?**].

- Databases – Many corporate and government entities have constructed huge databases containing a wealth of information. We require new algorithms for the efficient discovery of previously unknown patterns in these large databases.
- Geospatial data – consider the data collected by satellites that image the earth. We now have satellites that can create images at less than 1 meter resolution and that can collectively image the land surface of the planet in a very short time. These images must be examined in a time-critical matter.
- Sensor data – the revolution in miniaturization for computer systems has allowed us to produce a myriad of compact sensors. The sensors can collect data about their environment (location, proximity, temperature, light, radiation, etc.), can analyze this data, and can communicate between themselves. Collections of sensors can produce very large streaming sets of data.
- Video and image data – Image and video data are being used more and more to enhance the effectiveness of the security in high-risk operations. Content analysis, combined with massive recording capabilities, is also being used as a powerful tool for improving business processes and customer service. This streaming data paradigm creates new opportunities for visualization applications.

Each of these categories can produce massive data streams containing information that is applicable to a specific application domain. The grand challenge in the area of scalability is to develop new tools to distill the relevant nuggets of information from these widely disparate information streams, creating an explorable information space that can be examined by analytic or visual means to influence the decision of the data analyst. We must provide mechanisms that can visualize connections between the relevant information in the information streams, and allow the problem analyst to make decisions based on the totality of the information.

3 How Do We Explore Massive Data?

Nearly all methods to explore large, complex data sets rely on two methods: data transformations and visual transitions. To transform a massive data set into an "explorable" format, most researchers create a set of simple transformations that can be repeatedly applied to the data. To display meaningful visualization without screen clutter, the researcher must develop alternation visual metaphors and abstractions to illustrate the data. These transformations and abstractions are similar to the "zooms" presented by Bosch *et al.* [**?**] and Stolte *et al.* [**?**, **?**, **?**] Their framework utilizes multiple zooming panels for both data in visual abstractions. They think of multi-scale visualizations as a graph, were each node corresponds to a particular data representation and visual abstractions, and each edge represents a zoom. Zooming in the multi-scale visualization is equivalent of traversing this graph.

In the case of massive multidimensional abstract data, consider the (possibly infinite) set of data and visual representations of this information. Together, these represent the nodes of our graph. To explore this information, we need to establish edges between these nodes, and establish methods to find paths through this massive graph. What we wish to find, of course, are the representations that satisfy the basic principles we desire: uncluttered visualizations, maximal information, maximal decision support, etc.

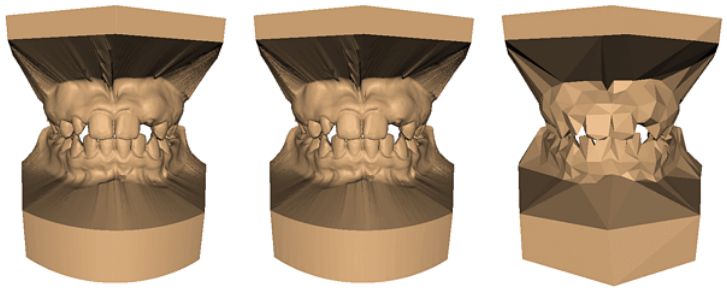
Given an information stream that must be explored, we can view the "visualization process" as a huge, possibly infinite, graph. Each node of this graph contain a representation of the data, and a visual representation method to be used on the data. The edges of the graph represent transitions between two data representations, two visual representations, or two data-representation/visual-representation pairs. In many scientific and engineering applications, the visual representation is given by a representation of three-dimensional space, and only the data transitions must be generated. In applications that work with abstract data, there is no inherited geometry, and visual metaphors and transitions are paramount. We will use this "transformation" and "transition" model throughout this paper.

4 Simplification Methods

Simplification and multiresolution methods for meshes, especially polygonal surface meshes, has dominated the research literature for several years. Most methods attempt to simplify the data through elementary data transformations.

In the case of triangle mesh simplification, most algorithms create a simple data transformation by first removing a contiguous set of triangles from the mesh, creating a hole. The hole is then re-triangulated, resulting in a mesh with fewer triangles. These simple data transformations are measured by use of an "error estimation" strategy and ordered by a priority scheme, creating a complex data transformation that (hopefully) preserves the features of the mesh, while producing a simplified mesh that can be examined. These data transformations can be classified into three categories: algorithms that simplify a mesh by removing vertices; algorithms that simplify a mesh by removing edges; and algorithms that simplify a mesh by removing higher-level simplices.

Schroeder *et al.* [**?**] and Renze and Oliver [**?**] have developed algorithms that simplify a mesh by removing vertices and the triangles containing a vertex. They



(a) (b) (c)

Fig. 2. Various simplifications of a data set representing a set of teeth. (Courtesy Mike Garland)

Fig. 3. fig:garland

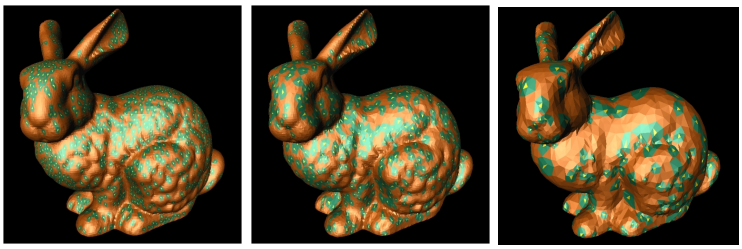
use a recursive loop-splitting procedure to generate a triangulation of the hole, while Renze and Oliver fill the hole by using an unconstrained Delaunay triangulation algorithm.

Hoppe [**?**, **?**] and Hoppe and Popović [**?**] describe a progressive-mesh representation, which effectively removes individual edges and their associated triangles. The "edge-collapse" operation forces a re-triangulation of the resulting hole. They place edges in a priority queue, ordered by the expected energy cost of its collapse. As edges are collapsed, the priorities of the edges in the neighborhood of the transformation are recomputed and reinserted into the queue. The result is an initial coarse representation of the mesh, and a linear list of edge-collapse operations, each of which can be regenerated to produce finer representations of the mesh. Other edge-collapse algorithms have been described by Xia and Varshney [**?**], who use the constructed hierarchy for view-dependent simplification and rendering of models, and by Garland and Heckbert [**?**], who utilize quadratic error metrics for efficient calculation of the hierarchy.

Hamann [**?**, **?**], and Gieng *et al.* [**?**, **?**] have developed algorithms that simplify triangle meshes by removing triangles. These algorithms order the triangles using weights based partially on the curvature of a surface approximation, partially on the changes in the topology of the mesh due to a triangle collapse, and partially due to a predicted error estimate of a collapse operation. Figure **??** illustrates the results of their method.

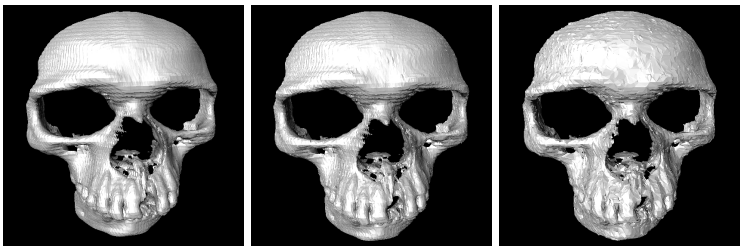
These surface algorithms have been lifted to tetrahedral meshes by several researchers [**?**, **?**, **?**] Staadt and Gross [**?**] have extended the progressive mesh algorithm of Hoppe [**?**] to tetrahedral meshes. Trotts *et al.* [**?**] use a strategy similar to the triangle collapse algorithm of Gieng *et al.* . Chopra and Meyer [**?**] identify individual tetrahedra T and remove the immediate neighbors of T. They force a retriangulation of the hole by simulating a collapse of the tetrahedron T to a point.

Cignoni *et al.* [**?**] treat the tetrahedral mesh problem by using a top-down, Delaunay-based procedure. The mesh is refined by selecting a data point whose



(a) (b) (c)

Fig. 4. Simplification by Triangle Collapse: Individual triangles are identified and collapsed to a point. The collapse point is based upon a best-fit quadratic polynomial to the surface in the neighborhood of the triangle.



(a) (b) (c)

Fig. 5. Tetrahedral Collapse Algorithm: The method of Trotts *et al.* [**?**] is used on a skull data set. Isosurfaces from the original mesh is shown in (a), an intermediate mesh in (b), and a coarse mesh in (c).

associated function value is poorly approximated by an existing mesh, and the mesh is modified locally to preserve the Delaunay property.

Error approximation for the elementary data transformation steps vary among the algorithms. Vertices to be removed by Schroeder *et al.* [**?**] are identified through a distance-to-simplex measure. The data reduction problem is formulated in terms of a global mesh optimization problem by Hoppe [**?**], ordering the edges according to an energy minimization function. The most frequently used measure is the quadric error metric, introduced by Garland and Heckbert [**?**].

Error approximation between meshes is addressed by several researchers. Cohen *et al.* [**?**] utilize an edge collapse-strategy to simplify polygonal models. They order the edges by using the distances between corresponding points of the mapping. A similar technique by Hoppe [**?**] is used for error calculations in level-of-detail rendering of terrain modeling. Klein *et al.* [**?**] calculate the Hausdorff distance between two meshes.

5 Multi-Resolution Methods

Several of the simplification methods of the previous section can also be used in a multiresolution manner. For example, the progressive mesh representation of Hoppe [**?**] can be used to create an initial coarse representation of a mesh, together with a linear list of edge-collapse operations. These can be used to regenerate various resolutions of the mesh by collapsing and expanding appropriate edges.

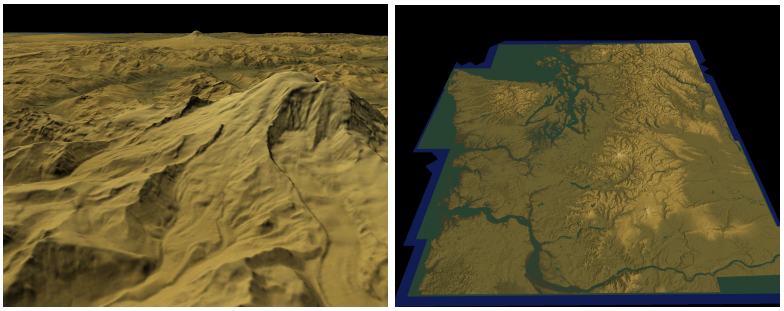
Many multiresolution geometry schemes exist for terrain rendering. Systems that use a regular grid approach are built on a hierarchy of right triangles [**?**, **?**], while triangulated irregular networks (TINs) [**?**, **?**, **?**] work to solve this problem by not restricting triangulations to a regular grid. Both schemes have advantages and disadvantages.

Several researchers have based terrain rendering on regular grids. Lindstrom *et al.* [**?**] present a method based upon an adaptive 4-8 mesh, using longest-edge bisection (LEB) as a fundamental operation to refine the mesh. They use a bottom-up vertex-reduction method to reduce the size of the mesh for display purposes. Duchaineau *et al.* [**?**] present a system for visualizing terrain also based upon a LEB paradigm. Their system uses a dual-queue management system that splits and merges cells in the hierarchy according to the visual fidelity of the desired image. Lindstrom and Pascucci [**?**] describe a framework for out-of-core rendering and management of terrain surfaces. They present a view-dependent refinement method along with a scheme for organizing the terrain data to improve coherence and reduce the number of paging events from external storage to main memory. Again, they organize the mesh using a longest-edge bisection strategy, using triangle stripping, view frustrum culling and smooth blending of geometry. Pajarola [**?**] utilizes a restricted quadtree triangulation, similar to an adaptive 4-8 mesh for terrain visualization.

Cignoni *et al.* [**?**, **?**, **?**] have demonstrated the ability to display both adaptive geometry and texture of large terrain data sets in real-time. They utilize a quadtree texture hierarchy and a bintree of triangle patches (TINs) for the geometry. The triangle patches are constructed off-line with high-quality simplification and triangle stripping algorithms. Hierarchical view frustrum culling and view-dependent texture and geometry refinement are both performed each frame. Textures are managed as rectangular tiles, resulting in an quadtree representation of the textures. The rendering system traverses the texture quadtree until acceptable error conditions are met, and then selects corresponding patches in the geometry bintree system. Once the texture has been chosen, the geometry is refined until the geometry space error is within tolerance.

Hwa *et al.* [**?**] adaptively texture a dynamic terrain mesh using an adaptive 4-8 mesh structure coupled with a diamond data structure and queues for processing both geometry and texture. Results of their technique are showing in Figure **??**. Their method can process massive textures in a multiresolution out-of-core fashion for interactive rendering of textured terrain.

Multiresolution methods can also be constructed around a novel data structure. The refinement of a tetrahedral mesh via longest edge bisection is described in detail in several papers. In Zhou et al. [23], a fine-to-coarse merging of groups of tetrahedra is used to construct a multilevel representation of a dataset. Their representation approximates the original dataset to within a specified tolerance. Gerstner and Pajarola [8] utilize a coarse-to-fine splitting of tetrahedra to extract topology preserving isosurfaces or to perform controlled topology simplification. Rendering of



(a) (b)

Fig. 6. Hwa *et al.* [**?**] display massive texture on terrain using a multiresolution system based on 4-8 textures. These illustrations show two views of a $120k \times 120k$ texture mapped onto terrain data from the State of Washington.

multiple transparent isosurfaces and parallel extraction of isosurfaces are presented by Gerstner [6] and by Gerstner and Rumpf [7]. Both of these algorithms extract isosurfaces from the mesh in a coarse-to-fine manner.

In Roxborough and Nielson [20], the coarse-to-fine refinement algorithm is used to model 3-dimensional ultrasound data. The adaptivity of the mesh refinement is used to create a model of the volume that conforms to the complexity of the underlying data.

Gregorski *et al.* utilizes a novel longest-edge-bisection mesh to build a multiresolution hierarchy of a volume dataset. They combine coarse-to-fine and fine-to-coarse refinement schemes for this mesh to create an adaptively refinable tetrahedral mesh. This adaptive mesh supports a dual priority queue split/merge algorithm similar to the ROAM system [**?**] for view-dependent terrain visualization. Sets of tetrahedra that share a common refinement edge are grouped into diamonds, which function as the unit of operation in the mesh hierarchy and simplify the process of refining and coarsening the mesh. At runtime, the split/merge refinement algorithm is used to create a lower resolution dataset that approximates the original dataset to within a given error tolerance. The lower resolution dataset is a set of tetrahedra, possibly from different levels of the hierarchy, that approximate the volume dataset to within this isosurface error tolerance. The isosurface is extracted from the tetrahedra in this lower resolution representation using linear interpolation. The authors have shown this method to be useful for fly-throughs of both static and dynamic three-dimensional data sets.

Abello and Vitter [**?**] provide an excellent presentation of external memory algorithms and their use in visualization. Abello and Korn [**?**, **?**] present a multiresolution method to display massive digraphs. Their method changes many graph nodes into a single node depending on the available screen space to display the graph. They utilize out-of-core methods to access relevant portions of the data.

6 External Memory Methods

External memory algorithms [**?**], also known as out-of-core algorithms, have been rising to the attention of the computer science community in recent years as they address, systematically, the problem of the non-uniform memory structure of modern computers. This issue is particularly important when dealing with large data-structures that do not fit in main memory since the access time to each memory unit is dependent on its location. New algorithmic techniques and analysis tools have been developed to address this problem in the case of geometric algorithms [**?**, **?**, **?**, **?**] and visualization [**?**]. Closely related issues emerge in the area of parallel and distributed computing where remote data transfer can become a primary bottleneck in the computation. In this context, space-filling curves are often used as a tool to determine, very quickly, data distribution layouts that guarantee good geometric locality [**?**, **?**, **?**].

Space filling curves [**?**] have been also used in the past in a wide variety of applications [**?**] because of their hierarchical fractal structure as well as for their well known spatial locality properties. The most popular is the Hilbert curve [**?**] which guarantees the best geometric locality properties [**?**]. The pseudo-Hilbert scanning order [**?**, **?**, **?**] generalizes the scheme to rectilinear grids that have different number of samples along each coordinate axis. Recently Lawder [**?**, **?**] explored the use of different kinds of space-filling curves to develop indexing schemes for data storage layout and fast retrieval in multi-dimensional databases.

Balmelli at al. [**?**] use the Z-order (Lebesque) space-filling curve to navigate efficiently a quad-tree data-structure without using pointers (See Figure **??**). Out-of-core computing [**?**] addresses specifically the issues of algorithm redesign and data layout restructuring, necessary to enable data access patterns with minimal out-of-core processing performance degradation. Research in this area is also valuable in parallel and distributed computing, where one has to deal with the similar issue of balancing processing time with data migration time. The solution of the out-of-core processing problem is typically divided into two parts:

1. algorithm analysis to understand the data access patterns of the problem and "redesigning" the data to maximize locality;
2. storage of the data in secondary memory with a layout consistent with the access patterns of the algorithm, amortizing the cost individual I/O operations over several memory access operations.

In the case of hierarchical visualization algorithms for volumetric data, the 3D input hierarchy is traversed to build derived geometric models with adaptive levels of detail. The shape of the output models are then modified dynamically with incremental updates of their level of detail. The parameters that govern this continuous modification of the output geometry are dependent on runtime user interaction, making it impossible to determine, *a priori*, what levels of detail will be constructed. For example, parameters can be external, such as the viewpoint of the current display window or internal, such as the isovalue of an isocontour or the position of an orthogonal slice. The general structure of the access pattern can be summarized into two main points: (i) the input hierarchy is traversed coarse to fine, level by level so that the data in the same level of resolution is accessed at the same time and (ii) within each level of resolution the data is mainly traversed coherently in regions that are in close geometric proximity.

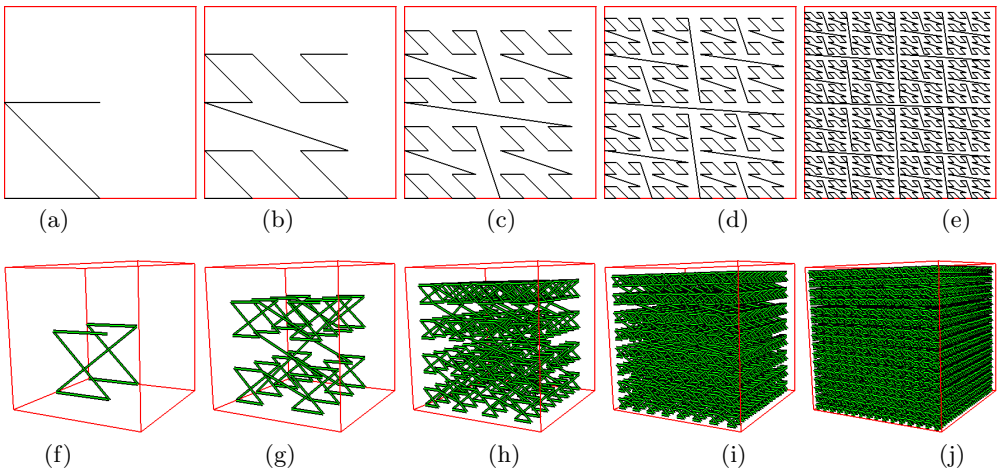


Fig. 7. (a-e) The first five levels of resolution of the 2D Lebesgue space-filling curve. (f-j) The first five levels of resolution of the 3D Lebesgue space-filling curve.

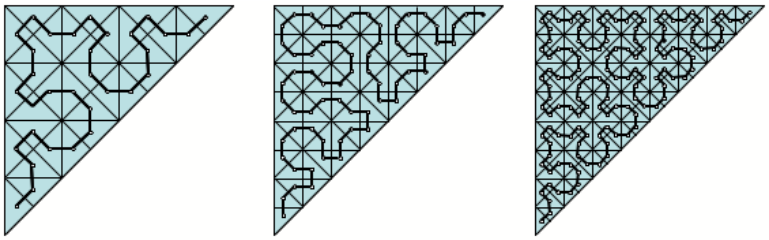


Fig. 8. Hwa *et al.* utilize an out-of-core storage system of bintree elements by utilizing a method based upon a Sierpinski curve. The Sierpinski curve arises naturally from the indexing of a bintree.

Pascucci and Frank [**?**] have introduced a new static indexing scheme that induces a data layout satisfying both requirements (i) and (ii) for the hierarchical traversal of n-dimensional regular grids. In their method, the order of the data is independent of the out-of-core blocking factor so that its use in different settings (e.g. local disk access or transmission through a network) does not require any large data reorganization. Conversion from the standard Z-order indexing to the new indexing scheme can be implemented with a simple sequence of bit-string manipulations. In addition, their algorithm requires no data replication, avoiding any performance penalty for dynamic updates or any inflated storage typical of most hierarchical and out-of-core schemes.

Isenburg *et al.* [**?**, **?**, **?**] utilize a streaming mesh paradigm to process massive meshes. They first sort the triangles of a mesh according to a selected coordinate. They can then input triangles up to the capacity of the system's main memory and process these triangles. Triangles that are no longer necessary can be discarded and

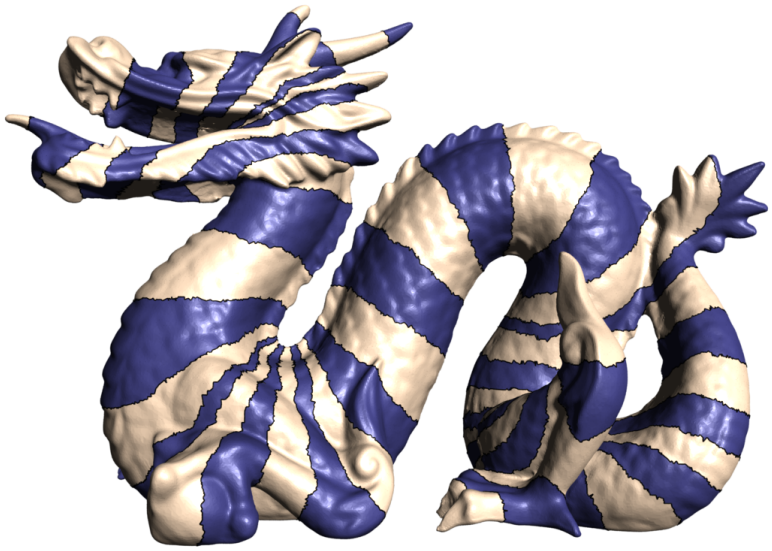


Fig. 9. Streaming Meshes. Triangles of the mesh are sorted by one coordinate, then input in order. This allows processing of a contiguous portion of the mesh, and out-of-core storage. (Courtesy M. Isenburg and P. Lindstrom)

new triangles can be input in order. They have used this methodology to perform simplification algorithms. The processing order on the dragon model is shown in figure **??**.

7 Visual Scalability

Visual scalability refers to the capability of data exploration tools to display massive data, in terms of either the number or the dimension of individual data elements, on a display device. The quality of visual displays, the visual metaphors used in the display of information, the techniques used to interact with the visual representations, and the perception capabilities of the human cognitive system, all affect the methods used to present information.

Many visualization problems arise from the complexity and dynamic nature of data sets. The effective display of this information is beyond the capability of current presentation techniques. The sheer size of the data makes it very difficult to effectively generate global views and one must generate novel interaction methods, coupled with innovative visual metaphors to allow the program analyst to explore the data.

Eick and Karr [**?**] proposed a scalability analysis and concluded that many visual metaphors do not scale effectively, even for moderately sized data sets. Scatterplots for example, one of the most useful graphical techniques for understanding relationships between two variables, can be overwhelmed by a few thousand points.

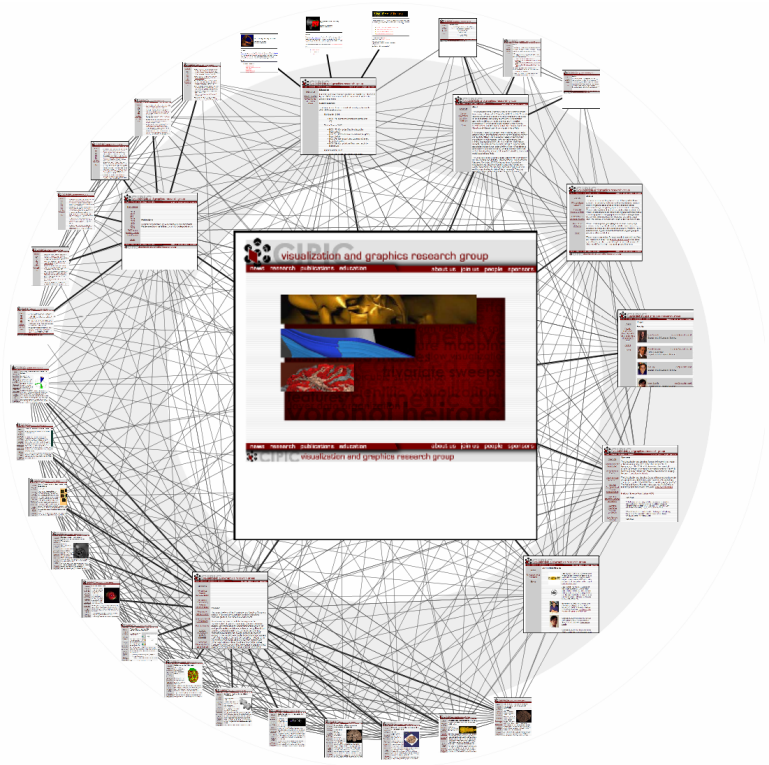


Fig. 10. Focus+Context Views (Courtesy, T.J. Jankun-Kelly)

"Focus+Context" refers to display methods that focus on detail while the less-important data is shown in a smaller representation. Interesting approaches include fish-eye views [**?**, **?**] and Magic Lens filters [**?**]. The table lens [**?**] applies fish-eye techniques to table-oriented data. Plaisant has proposed the SpaceTree [**?**], a tree-browser which adds dynamic rescaling of branches of the tree to best fit the available screen space. Munzner *et al.* have illustrated similar concepts in TreeJuxtaposer [**?**]. Hierarchical Parallel Coordinates [**?**, **?**], a multiresolution version of Parallel Coordinates [**?**], uses multiple views at different levels of detail for representation of large-scale data.

8 Conclusions

The research in massive data exploration is only in its infancy. The grand challenge problem, to explore dynamic multi-dimensional massive data streams, is still largely unexplored. We must extend the state-of-the-art in visual and data transformations to be able to explore these complex information streams. We must develop new interaction methods to explore massive data in a time-critical matter. We must develop new techniques for information fusion that can integrate the relevant nuggets

of information from multi-source multi-dimensional information streams. We must develop new methods to address the complexity of information, and create a seamless integration of computational and visual techniques to create a proper environment for analysis. We also must augment our methods to consider visual limits, human perception limits, and information content limits.

There is much to do.

Acknowledgments

This work was supported by the National Science Foundation under contracts ACR 9882251 and ACR 0222909, and by Lawrence Livermore National Laboratory under contract B523818. Thanks to Valerio Pascucci and Mark Duchaineau for their valuable assistance.