Interrogative Visualization of the Visible Human Datasets

Chandrajit L. Bajaj Fausto Bernardini Valerio Pascucci Daniel R. Schikore

Shastra Lab & Center for Image Analysis and Data Visualization
Department of Computer Sciences
Purdue University*

Abstract

We present an environment for interrogative visualization of the Visible Human datasets. Components of the environment have been developed to support rapid computation, rendering, and querying of large amounts of data. A cluster of graphics workststations connected by ATM to an Intel Paragon provides the computational and display power required to support visualization in an interactive sense. We discuss the three components of interrogative visualization and how they are addressed in our design.

1 Interrogative Visualization

We are developing the tools necessary for Interrogative Visualization of large datasets such as the Visible Human data. Interrogative Visualization consists of three primary components:

- **Computation** - rapid computation of isosurfaces, reduced meshes, volume rendering, etc.
- **Display** - interactive rendering of graphics primitives
- **Querying** - extraction of quantitative information from accurately constructed models

It is the premise of this paper that interaction with and interrogation of large datasets allows the visualization user to more effectively browse through the large amounts of data, focus on and query interesting features, and more easily comprehend and make use of the data. We approach all three of the key components through accelerated methods for contour extraction, mesh reduction for improved interactive display, and smooth surface reconstruction for querying. These tools are being developed in an integrated environment called SHASTRA. The SHASTRA collaborative substrate aids in our development of a tele-collaboritive interrogative visualization environment implemented on an ATM workstation cluster with a high-speed link to an Intel Paragon. The computational visualization environment is illustrated in Figure 1.

2 Isosurfacing

Traditional computationally intensive visualization techniques such as isosurfacing do not allow the user real-time interaction for large datasets if implemented with a brute-force approach. A major factor in the efficiency of isocontouring algorithms is the search for cells which contain part of the surface of interest. A major class of accelerated isocontouring algorithms are designed specifically to speed this search. In our approach, we first limit the search to a subset S of cells, with the property that all connected components of any isosurface pass through at least one cell in S. We construct an interval search structure over the cells in S which permits a $O(\log n + k)$ search for all intersected cells in S, where n is the size of S and k is the number of intersected cells. Finally, a contour propagation algorithm is applied to compute the entire surface corresponding to each *seed cell* found in the search phase. In the following sections we will give more detail on each phase.

2.1 Contour Propagation

Selection of the set S and the properties that it holds are heavily dependent on the contouring propagation algorithm applied, thus we will introduce propagation first.

The central idea is that, given an initial cell (a seed cell) which contains the surface of interest, the remainder of the surface can be efficiently traced by performing a breadth-first search in the graph of cell adjacencies, as illustrated for a 2D contour extraction in Figure 2. Use of a breadth-first search keeps the memory requirements minimal, as only the “advancing front” of the surface needs to be stored

*West Lafayette, Indiana
47907-1398 USA

{bajaj,fxb,pascucci,drs}@cs.purdue.edu
http://ww.cs.purdue.edu/research/shastra

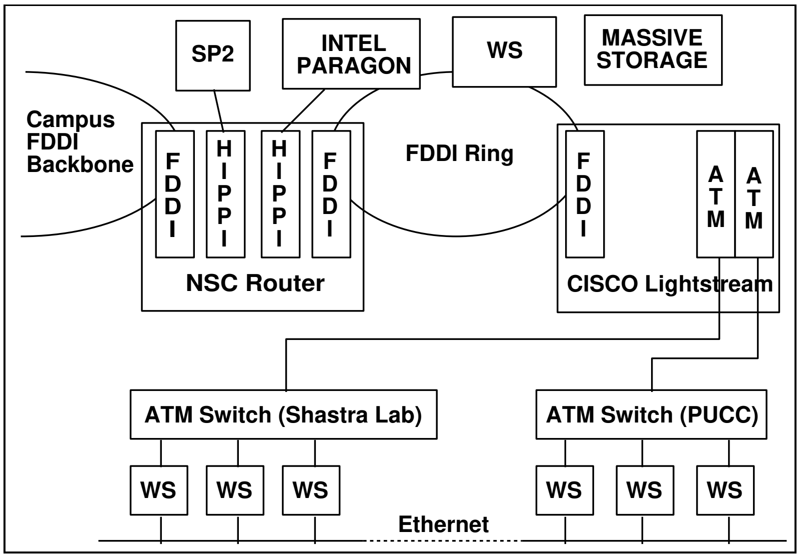


FIGURE 1: Network environment for interrogative visualization

at any one time. One advantage to using a propagation approach over other techniques is that surfaces are easily transformed into a *triangle strip* representation for more efficient rendering [6]. Also of importance is the fact that shared vertices between cells are more efficiently located, as we are considering only a single closed contour at any given time. In our implementation, carefully hashed indexing of the advancing front allows us to efficiently eliminate recomputing intersections when the advancing front closes on itself, completing the extraction of a connected component. Similar to related caching techniques [1, 7], the cache is made efficient by discarding entries which are known to not be referenced again, based on the maximum number of cells which share a given edge.

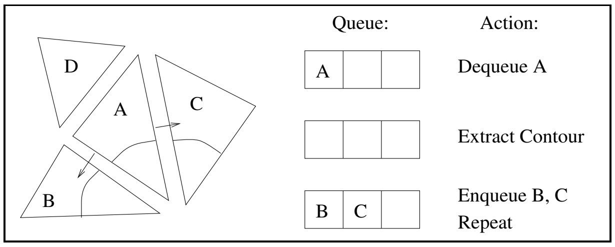


FIGURE 2: Illustration of contour propagation

2.2 Seed Set Construction

The set S of cells is called a *seed set*, as the goal is to seed the contour propagation algorithm defined above with cells from S. We describe a specialized algorithm for determining a seed set for a structured mesh, and cite performance characteristics for several subvolumes from the Visible Human data.

Initially, let the set S consist of all cells c, with an associated range $T(c)$ which is the range of scalar values taken on over the given cell, $R(c)$. We proceed in visiting each cell in a regular marching order, propagating *responsibility ranges* in the direction of the cell traversal. Intuitively, a cell must treat a *responsiblity range* as the set of values for which previously visited cells have deferred *responsibilty*. Upon visiting a cell, incoming responsibility must either be further propagated, or accounted for by the current cell. The end result is that a set of simple tests and range operators can be applied to each cell in a regular traversal, rapidly pruning cells from the seed set S.

For a more complete discussion and details of seed set construction, as well as a proof of seed set construction, see [5].

2.3 Search Structures

Given a seed set, we need to determine all cells in the seed set whose range contains a given value. Several search structures exist which will perform this operation in $O(\log n)$ time. Two in particular, the *interval tree* and the *segment tree*, are being investigated in further detail.

2.4 Contouring Results

Preprocessing statistics for several subvolumes of the Visible Human datasets are given in Table 1. Note that the size of the seed set is 1-2 orders of magnitude smaller than the total size of the data. *Tree Size* denotes the number of cell labels stored in a segment tree data structure for the seed set. Again, storage overhead is much less than the total size of the data.

Surface extraction data are presented in Table 2. Note that the seed cell approach results in a very consistent performance rate as measured by *triangles/sec*, indicating that the average complexity of surface extraction is linear with respect to the size of the resulting surface. This claim is further supported by the graph in Figure 2.4, which plots performance for multiple isovalue queries on a subvolume of a femur of the Visible Male.

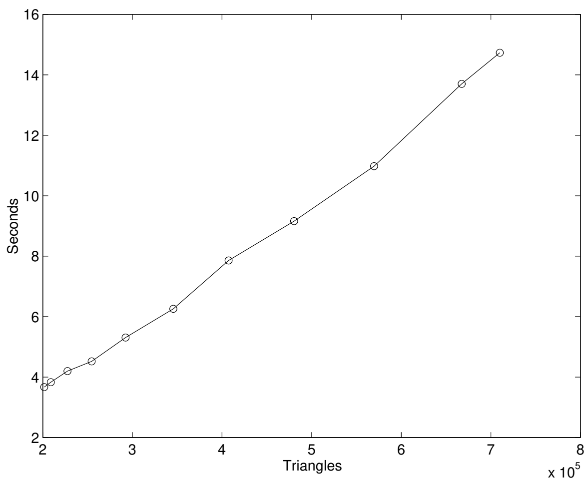


FIGURE 3: Linear performance for surface extraction from Male Femur

3 Surface Simplification

Interactive rendering of large surfaces, defined by isosurfaces or otherwise constructed, demands hierarchical data structures which accurately represent a surface at multiple levels of detail. Construction of a hierarchy of models may be integrated with the extraction of isosurfaces or may additionally be performed as a post-process. In our mesh reduction work, arbitrary precision and topological consistency can be attained, while preserving both the geometry of the surface as well as any number of scalar functions defined on the surface. Our algorithm follows the general structure of a *point-deletion* approach to simplification. The addition of error bounds which are propagated through successive deletions of vertices provides global error control on both the geometry and any number of functions defined.

3.1 Surface Mappings

The development of a mapping between successive surfaces allows us to compute the amount of *introduced error* from a topological surface operator such as a point deletion. Projection of edges introduced through hole-triangulation onto the original surface results in a piecewise continuous mapping between triangulations, as illustrated in Figure 5.

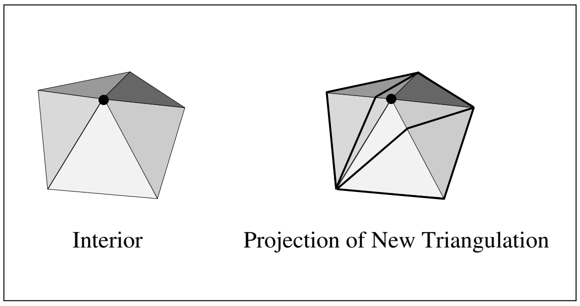


FIGURE 5: Mapping for an interior vertex

3.2 Computing Error Bounds

The previously defined mapping from a triangulation around a candidate vertex to a retriangulation permits us to compute errors introduced over the surface. The mutual projection segments the triangulation into pieces within which the functions and the geometry all vary linearly, as shown in figure 6. Thus, it suffices to compute errors at the intersections of the projected edges in order to compute an upper bound on the error in each triangle of the new mesh.

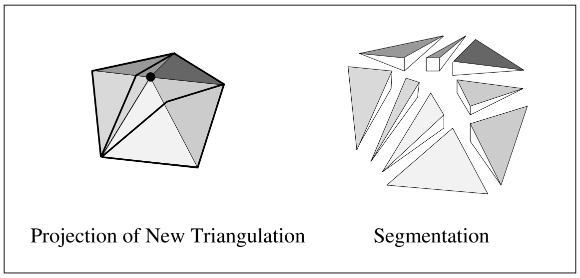


FIGURE 6: Segmentation of mutual projection

3.2.1 Error Propagation

We choose a simple scheme for propagation of error from one triangulation to the next. For the geometry and each scalar variable, we associate two variables with each triangular face. One variable is a current upper bound on the

Data	Resolution	Seed cells	% of total	Distinct Seeds	Tree Size	Preprocessing Time (s)
Male Head	512x512x181	2217568	4.72%	3294	8922765	596.8
Female Head	512x512x208	2276972	4.21%	3154	8548789	669.7
Male Pelvis	325x150x270	1110420	8.56%	2517	6602785	183.2

TABLE 1: Data and Preprocessing Characteristics

Data	Isovalue	# Tri	Marching Time (s)	Our Time (s)	Speedup	Marching Tri/sec	Our Tri/sec
Male Head	600.5	1329920	323.52	28.40	11.39	4111	46828
same	1224.5	2110962	322.53	47.20	7.05	6545	44723
Female Head	600.5	1415154	381.10	35.60	10.71	3713	39752
same	1224.5	2196390	385.59	56.23	6.85	5696	39060
Male Pelvis	1224.5	1879088	94.20	45.56	2.07	19947	41244

TABLE 2: Surface Extraction Timing Statistics

"positive" error accumulated, the other represents "negative" error. Given the accumulated errors on the faces surrounding a candidate vertex, we compute and add in the errors which are introduced through deleting the vertex. Figure 7 illustrates the process of computing an upper bound on the error along an introduced edge based on propagated error from the previous triangulation and introduced error caused by retriangulation.

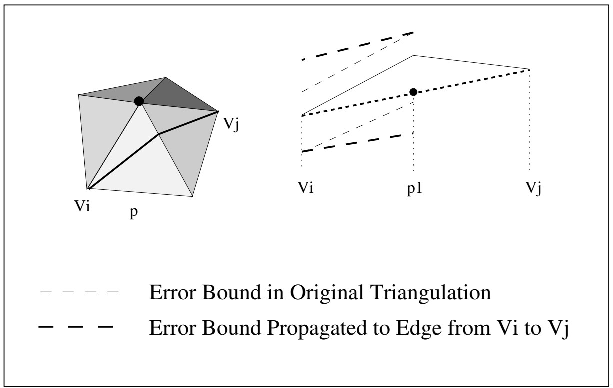


FIGURE 7: Propagating error to a new triangulation

3.3 Simplification Examples

Figure 3.3 demonstrates the simplification of a lower resolution isosurface of the head of the Visible Woman. An error bound of less than 1% of the representative size of the surface (maximum diameter) yields a simplification by greater than an order of magnitude.

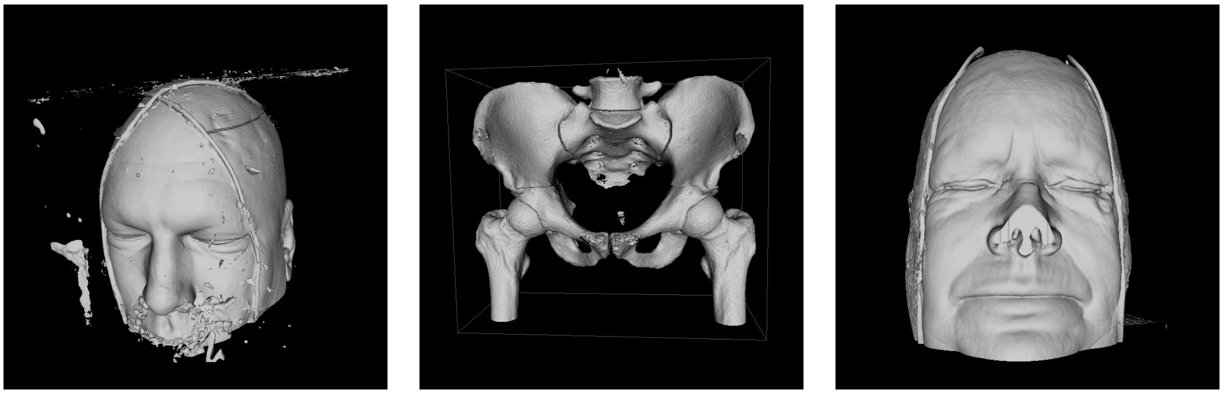
4 Surface Modeling

Extraction of quantitative information from models requires accurate reconstruction techniques. To this end, we have developed a reconstruction algorithm which generates a C^1 continuous surface representation from a uniformly distributed set of points in space. The first step is to construct a 3D Delaunay triangulation of the points, from which we extract an α-solid, that is a subset of the tetrahedra in the 3D triangulation that form a solid shape, whose boundary interpolates the data points.

Given a sufficiently dense and uniform sampling of the surface, the α-solid can be recovered automatically. The boundary of the α-solid is a triangle-mesh interpolating the data points, and can be used for visualization and computation purposes. Notice that the α-solid also provides a tetrahedralization of the object, which can be used (eventually after further subdivision) for FEM analysis.

To obtain a more compact representation, and when a derivative-continuous model is needed, we can build an *A-patch* representation of the model [4]. An A-patch surface is a collection of polynomial patches of low degree, whose continuity (C^1 for degree three patches, C^2 for degree five patches) can be easily achieved. Each patch is defined as the zero-contour of Bernstein-Bézier polynomial, defined within a tetrahedron. We therefore build a tetrahedral mesh containing the surface, and then set the coefficients of the polynomials to approximate the data points to a given tolerance, with the desired continuity constraints.

We have developed two methods to reconstruct a smooth object's model: In the first method [3], we define a volumetric signed distance function based on the α-solid. We then incrementally build a tetrahedral mesh that supports the A-patches. The mesh is adaptively refined until the desired approximation tolerance is satisfied. With the



(a) (b) (c)

FIGURE 4: Isosurfaces from the Visible Male (a-b) and Visible Female (c). Speedups attained were up to 11 times (over a brute-force surface extraction approach)

second approach [2], we first decimate the boundary of the α-solid, then build a collection of tetrahedra (called *simplicial hull*, on both sides of the triangle mesh. A-patches are then built inside the simplicial hull, approximating data points and estimated surface normals. Simultaneous reconstruction of one or more scalar functions defined at the input points allows us to model and query material properties which are derived from the input CT or MRI data.

Figure 9(a) shows the tetrahedral mesh and the associated spline model of the femur. Figure 9(b) shows a reconstructed model for the knee from the VHP data set.

Acknowledgements

This research was supported in part by NSF grants CCR 92-22467 and GER 9253915-02, AFOSR grant F49620-94-10080, and ONR grant N00014-91-1-0370.

References

[1] E. Artzy, G. Frieder, and G. T. Herman. The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm. In *Computer Graphics (SIGGRAPH '80 Proceedings)*, volume 14, pages 2–9, 1980.

[2] C. Bajaj, F. Bernardini, J. Chen, and D. Schikore. Automatic reconstruction of 3D CAD models. In *Proc. of the International Conference on Theory and Practice of Geometric Modeling*. Springer-Verlag, 1996. Accepted for publication.

[3] C. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3D scans. Annual Conference Series. Proceedings of SIGGRAPH 95, pages 109–118. ACM SIGGRAPH, 1995.

[4] C. Bajaj, J. Chen, and G. Xu. Modeling with cubic A-patches. *ACM Transactions on Graphics*, 14(2):103–133, 1995.

[5] Chandrajit L. Bajaj, Valerio Pascucci, and Daniel R. Schikore. Fast isocontouring for improved interactivity. In *Proceedings of 1996 Symposium on Volume Visualization*, October 1996.

[6] C. T. Howie and E. H. Blake. The mesh propagation algorithm for isosurface construction. *Computer Graphics Forum*, 13(3):65–74, 1994. Eurographics '94 Conference issue.

[7] Jane Wilhelms and Allen Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, 1992.

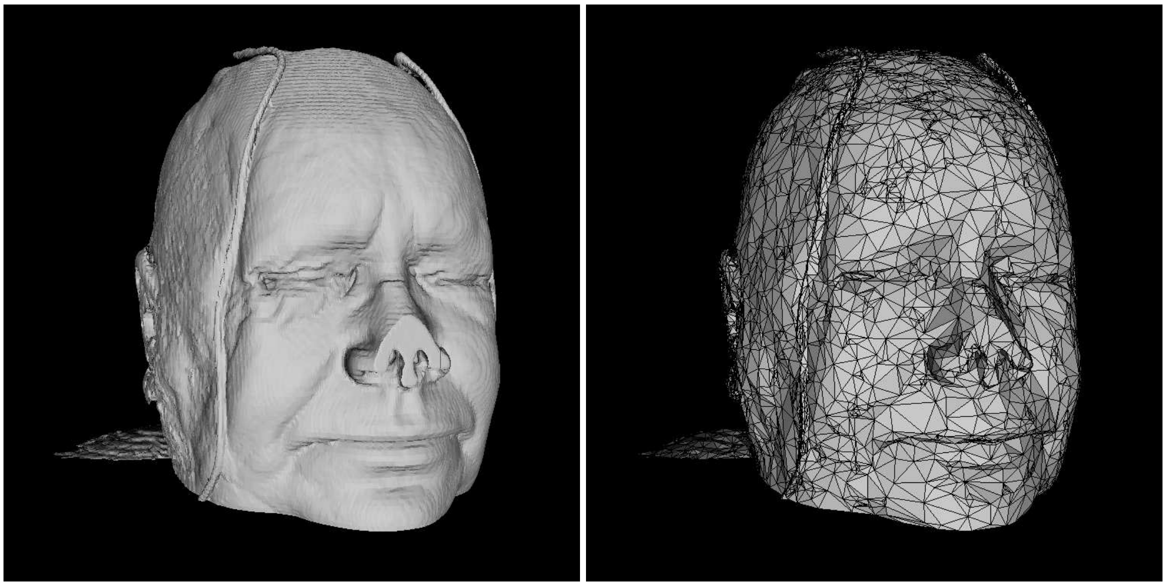
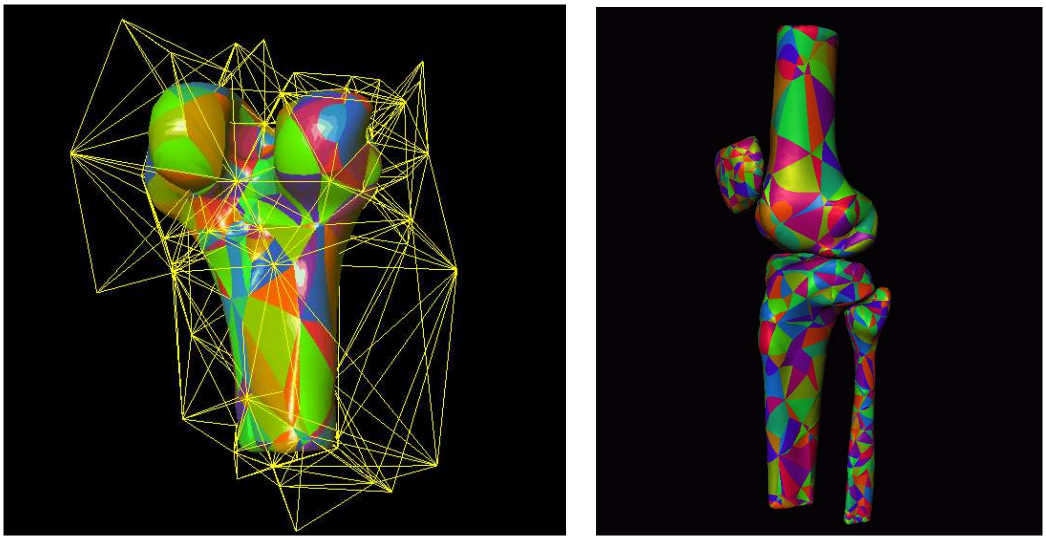


FIGURE 8: Original female head (275048 triangles) and simplified model (24211 triangles)



(a) (b)

FIGURE 9: A-patch reconstruction of a knee