MeshView: A Tool for Exploring 3D Unstructured Tetrahedral Meshes

Carole S. Gitlin[1] and Christopher R. Johnson[2]
[1]Autometric, Incorporated
Email: cgitlin@autometric.com
[2]Department of Computer Science
University of Utah, Salt Lake City, UT 84112.
E-Mail: crj@cs.utah.edu
Web: http://www.cs.utah.edu/ ~sci/

Abstract. We have developed a visualization module, *MeshView*, for viewing large-scale 3D unstructured tetrahedral finite element meshes. The module includes two different approaches for visualization and analysis, plus the capability to edit meshes after the researcher has analyzed the mesh and located problem areas. The first approach offers various real-time interactive manipulation options for visualizing the geometry of the mesh. The second approach provides interactive histograms for displaying quantitative data about the elements and their neighbors. Finally, there are several editing options that allow the user to add nodes and delete elements or nodes from the mesh.

keywords. mesh visualization, unstructured grids, analysis

1 Introduction

Computational solutions of large-scale 3D boundary value problems in science and engineering usually require that the solution domain be partitioned into a set of contiguous polyhedral elements, or a mesh. These meshes typically range from several hundred thousand elements to millions of elements. As such, mesh visualization is important for studying these sets and for assessing solution accuracy, checking geometrical integrity, identifying missing and/or distorted elements, and interactively editing the mesh. Furthermore, with adaptive mesh refinement and derefinement methods that alter the structure of the mesh, [1, 2, 3, 4] it is especially helpful to visualize the changes taking place as an area of the mesh is altered. It is important for the researcher to be able to verify visually and to isolate the (de)refinement.

The computational mesh is usually one of two types, structured or unstructured. Other types of meshes are hybrids of these two distinct forms. In structured meshes, the polyhedral elements are of a uniform shape on a Cartesian grid that may be mapped through linear and nonlinear transformations [5]. Therefore, it is simple, provided that the mesh is not too dense, to identify the shape and position of elements. A researcher can visualize the geometry and underlying structure using traditional graphical techniques, comprising rotation, translation, scaling, and clipping planes.

Unstructured meshes, however, have no inherent geometric structure or form and require the difficult task of understanding the spatial relationships of unstructured 3D objects in 2-space. The elements give the researcher no clues as to how they fit together, since, unlike in structured meshes, the elements are not arranged on a Cartesian grid. It is not even always possible to determine the shape of an element. Fig. 1 shows examples of regular and irregular unstructured tetrahedral meshes, both rendered with the *MeshView* program, while Fig. 2 shows the unstructured mesh we use for computing the bioelectric fields within a large-scale model of the human thorax, the project that gave rise to our new visualization module [6, 7].

Although people have spent years researching ways to create meshes and the numerical methods used to approximate numerical solutions to the governing boundary value problems associated with the science and engineering problem at hand, little work has been done in the visual analysis of such meshes until 1990. Since then, much of the research on this topic has been in one of three main areas: volume rendering, planar slices, and graphical analysis tools. Volume rendering, however, is used primarily to view the solution of a finite element approximation and is not as useful in analyzing the actual mesh used in the approximation.

The two more analytical approaches are planar slicing and graphical analysis tools. With planar slicing [5], a plane cuts through the finite element mesh, and then the contours of the elements are displayed on the slice. This reduces

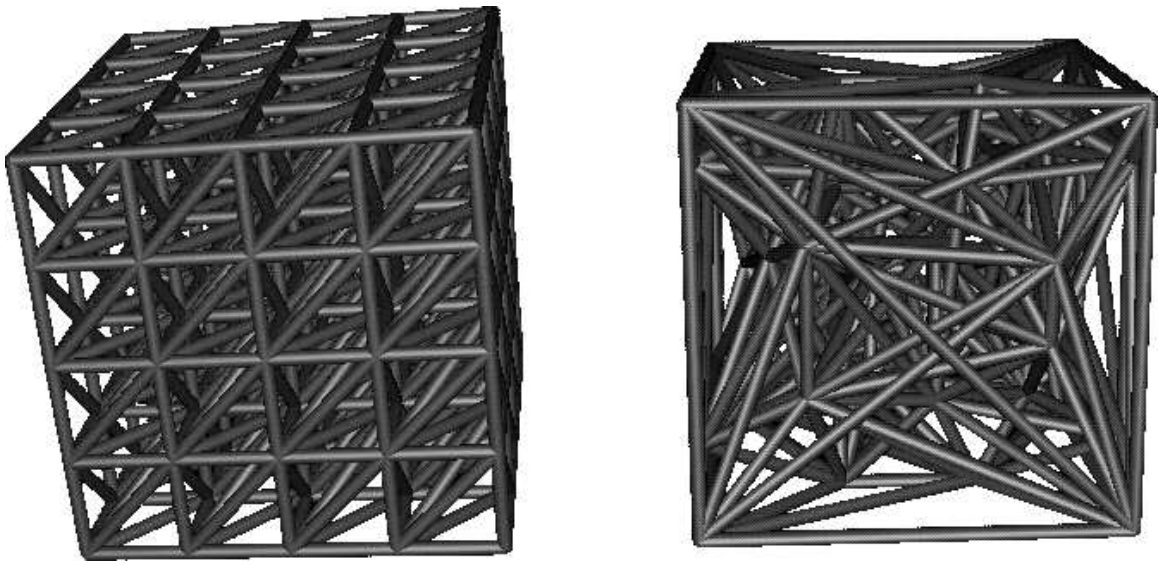

Figure 1: Example of a regular, structured tetrahedral mesh containing 125 nodes and 320 elements and an irregular, unstructured tetrahedral mesh containing 8278 nodes and 51,125 elements.

a complex 3D entity into a simple 2D plane, thus causing the important 3D relationships of the elements, as well as the volume and aspect ratio data, to be lost.

There is ongoing work by several different groups on graphical analysis tools [8, 9]. These tools allow the user to interact with complex meshes and to pull out important quantitative information about the mesh and the elements of the mesh. However, for our needs, we found deficiencies in all of these programs.

To overcome these problems, we have developed a graphical analysis tool called *MeshView* for visualizing and analyzing 3D unstructured meshes. *MeshView* contains a suite of options to provide the user with interactive control for mesh visualization. The module is part of a data-flow based computational steering system, SCIRun, used for the interactive solution and visualization of science and engineering problems [10, 11].

We present two different approaches to mesh visualization. The first approach provides visualization tools that allow the user to explore the mesh visually and therefore can enhance the understanding of spatial relationships. For this exploration, we have developed two different techniques. First, we developed "clipping surfaces," which are similar to clipping planes except that partial elements are discarded instead of drawn. This leaves an uneven surface of complete elements, rather than a confusing set of line segments. Clipping surfaces help researchers understand the spatial relationships of the elements since they can watch as elements are added or removed.

The second visualization technique for exploration is a "growth" algorithm. We start with a single tetrahedron. Each successive level of the visualization contains all those tetrahedra that share a face with a tetrahedron in the previous level. A switch allows the user to view either only the elements at the current level or all elements up to and including the current level. By looking only at the current level, the user can see the mesh grow through a propagation wave and can also see all elements at a certain distance from the mesh. With the other option, the user can watch the mesh as a whole as it grows. Both options give the user a better sense of how the individual elements relate to each other and to the entire mesh.

The second approach to the visualization problem provides a variety of quantitative measures pertaining to the mesh. This information is displayed as an interactive 2D histogram and includes element volume, aspect ratio, mesh connectivities, and solution error. Users can slide two bars on the histogram to specify a range of values to be displayed. Alternatively, the user can investigate the elements that fall *outside* the range, thus comparing simultaneously elements with the largest and smallest values of data.

After analyzing a mesh, researchers may decide to edit the mesh. Perhaps there is a large element that should be broken up into smaller elements, or there may be a flat element that should be removed. In our program, we have included editing techniques to meet these needs. In addition to an option that allows a user to add a node in the center of an element to create four new elements out of the old one, there are also two deleting options. The first option allows the user to delete the four nodes of an element, while the second will delete only one node of the

Figure 2: Example of an unstructured tetrahedral mesh for a model of the human thorax for large-scale bioelectric field simulations.

element. After the mesh has been edited, *MeshView* re-tetrahedralizes the mesh.

The goal in creating this visualization environment is to allow users to validate the structure and integrity of their meshes and, if necessary, to edit the mesh. The combination of the two different approaches provides users full manipulative control over the mesh and allows them both to understand the 3D structure and to locate problem elements.

2 Methods

The exploratory techniques provided by *MeshView*, those that enhance the user's understanding of the mesh and provide visual verification, fall into two different categories: manipulative techniques and quantitative techniques. The third category is for editing techniques, which allow the user to add and delete nodes in the mesh.

2.1 Manipulative Techniques

Manipulative techniques allow the user to interactively explore the geometry of a mesh. The various options of our system are designed to maximize the user's understanding of the spatial relationships of the tetrahedral elements. We currently have two different approaches that work together: clipping surfaces and a growth algorithm.

Clipping surfaces

Conventional clipping planes fail when used to view unstructured 3D meshes because they cut across portions of elements. As the plane is moved, elements being sliced are reduced to a set of line segments that merely add to the visual confusion of the mesh rather than decreasing it. To overcome this problem, we implemented clipping surfaces.

Instead of cutting off part of an element, this utility draws elements only in their entirety, so that if the plane goes through a particular element, no part of the element is drawn. The resulting surface is irregular, as the elements are of varying sizes, but all the elements are complete. By moving any one of the six orthogonal planes, the user can see how entire elements are added and deleted. This is especially useful for visualizing the spatial positioning of an element. The position of the surface when an element is added or removed enables the scientist to have a better grasp of where that element is in space and in relation to neighboring elements.

Growth algorithm

With the growth algorithm, the user selects a starting element, from which to grow the mesh out, level by level, where a level is defined as all elements that share a face with an element in an inner level. Thus the first level contains the starting element, the second contains the four tetrahedra that share a face with the starting element, and so on. The user not only has control over how many levels are drawn, but also whether only the outermost level is drawn or all previous levels are also drawn. Viewing only the outermost level facilitates visualizing all elements at a certain "distance" away from the starting element. To ease viewing of all the levels, the outermost level is drawn in a different color from the inner levels. Similarly, the starting element is also drawn in a different color to make it easy to find. These various techniques all allow the user to gain a better understand of how the elements of the mesh fit together. The growth algorithm can also aid in finding potential problem spots. For example, if a place is found where there is a high concentration of neighboring elements, the user might decide that needs to be edited. Fig. 3 shows four consecutive stages in the growth algorithm, starting with the seed tetrahedron in Fig. 3 and progressing out three levels.

A preprocessing stage sets up a data structure with the connectivity information of the elements. The algorithm starts with a seed element and then builds the structure up level by level, using a stack. Once the data structure is filled, it needs to be modified only if the user changes the starting element. The user can then interactively grow and shrink the mesh. Preprocessing the level information requires more storage than calculating connectivities on the fly, but it allows for real-time interaction.

2.2 Quantitative Techniques

Complementary to visualization of the meshes is the display of quantitative data. These data include tetrahedral volume, aspect ratio, element size versus neighboring elements sizes, and solution error. For each type of data, an interactive 2D histogram is drawn. The histogram shows how many elements there are for different values of the data. The user is able to specify graphically a range of values and have all the elements in that range be displayed in the viewing window. With this feature, the user can look at, for example, all tetrahedra that have a large aspect ratio or near zero volume. Without being able to pull out such information, the user is unable to know where such elements may be, if they exist at all. By using this information to locate problem areas, the user can determine precisely where to refine the mesh.

Additionally, there is a switch on the user interface that allows the user to specify which elements are to be displayed, those inside the range or those outside the range. This feature allows the user to compare elements with the smallest and largest values simultaneously. For rendering the elements, we provide two options: only the elements in the range are drawn, or those elements plus the rest of the mesh in wireframe are drawn. Drawing the rest of the mesh in wireframe permits the user to see easily where the specified elements are within the mesh.

Volume

Although the value of an individual element's volume is not overly important or indicative of the state of the mesh, the range of volumes compared to each other is an indication of mesh gradation. Having several orders of magnitude between the minimum and maximum volume can be an indication of a malformed mesh if the mesh is poorly graded. Additionally, the volumes of the mesh are indicative of the condition number of the matrix used in the finite element method.

The condition number is a measure of how close the matrix is to being singular. The condition number is the reciprocal of the relative distance from the matrix to the set of singular matrices [12]. Matrices with large condition numbers are said to be ill-conditioned [13]. The ideal problem conditioning occurs for orthogonal matrices that have $\kappa(A) \approx 1$, while an ill-conditioned matrix will have $\kappa(A) \gg 1$. When one inverts a matrix that has a very large condition number (i.e. solves the linear system), the solution process is unstable and is highly susceptible to errors.

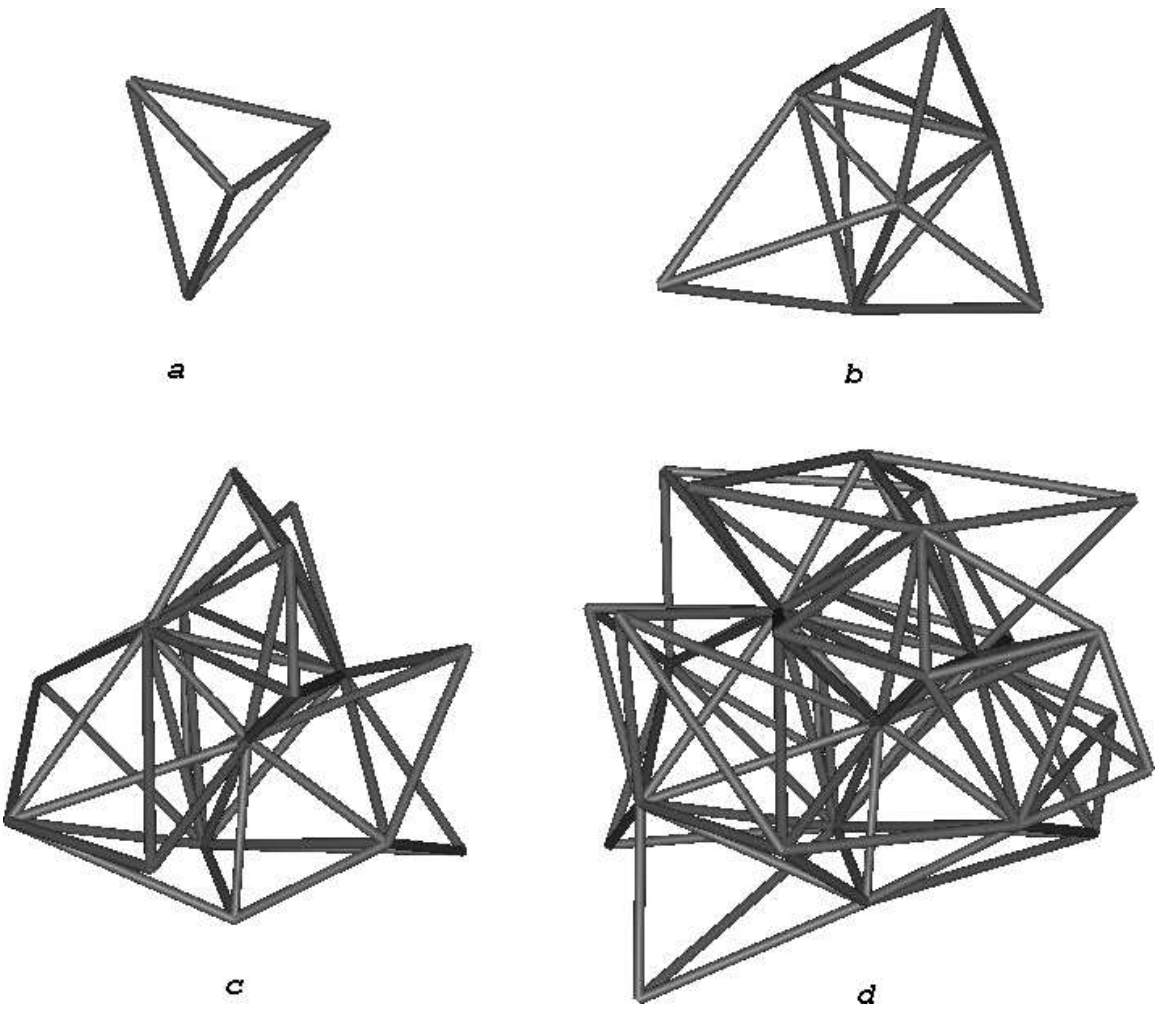

Figure 3: Sequence of four consecutive growths of the mesh. a) Seed tetrahedron. b) All tetrahedra sharing a face with the seed tetrahedron. c) All tetrahedra sharing a face with those in b. d) All tetrahedra sharing a face with those in c.

For ill-conditioned problems, small relative errors in the input data can cause large relative errors in the results. The condition of a matrix also depends on the precision level of computations and is a function of the size of the problem. For example, if the condition number exceeds a linear growth rate with respect to the size of the problem, the problem will become increasingly ill-conditioned. The condition number of a matrix A is defined as

$$\kappa(A) = \| A \| \, \| A^{-1} \|, \tag{1}$$

using one of several norms, including the 1-norm, 2-norm, and $\infty - norm$. The $\infty - norm$ is defined as

$$\| A \|_{\infty} = \max_{1 \leq i \leq n} \sum_{j=1}^{n} | a_{ij} | \, . \tag{2}$$

The size of the condition number is an inherent property of the matrix and is proportional to the size of the matrix or the number of degrees of freedom [12]. Additionally, the number of degrees of freedom is proportional to the mesh refinement, which is based on the volume of the elements [14]. While a quantitative relationship between the

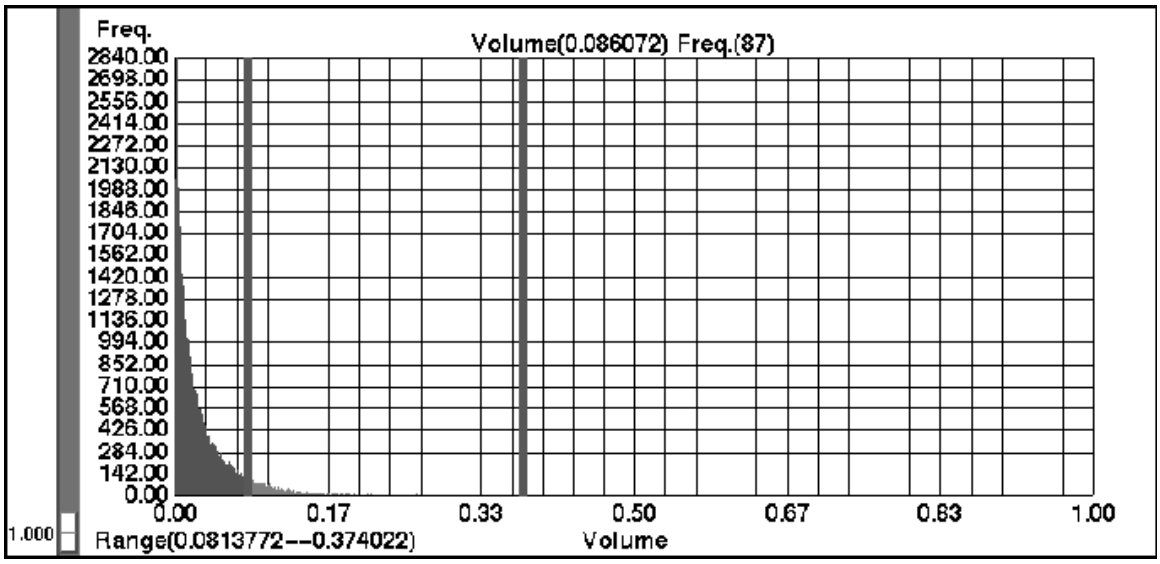

Figure 4: Interface for the interactive histogram. The vertical bars are movable and allow the user to specify the range to look at.

condition number and the range of volumes within an unstructured tetrahedral mesh is still an open problem, it should be clear that there is a correspondence between the two. By examining the volumes of the elements of the mesh with the histogram, the researcher is able to narrow in on sections of the mesh that might need to be refined.

Aspect ratio

The aspect ratio of a tetrahedron gives an indication of how well formed that tetrahedron is. The aspect ratio is the measure of how "regular" an element is, where a regular element is defined as an equilateral tetrahedron, which has an aspect ratio of 1. Therefore, an optimal mesh is formed with tetrahedra as close to equilateral as possible, with aspect ratios close to 1. A degenerate, *flat*, element will have an aspect ratio of 0. Users can use the histogram to isolate elements with "bad" aspect ratios. One formula for the aspect ratio of a tetrahedron is $\sqrt{\frac{3}{2}}\frac{h_k}{\rho_k}$, where ρ_k is the diameter of the sphere circumscribed about the tetrahedron and h_k is the maximum distance between two vertices [15]. Fig. 5 shows a histogram of the aspect ratios of a mesh with approximately 10,000 nodes and 30,000 elements.

Size versus neighbors

The volume of an element versus the volume of its neighbors is related to the aspect ratio of the element and is, again, a good indication of how well formed the mesh is. We compare the element's volume with the four neighboring elements' volumes and take the largest difference. If the test element's volume is much different from that of one of its neighbors, the difference could be an indication that we have a poorly graded mesh. In order for the volumes to be quite different, the smaller element will, of necessity, be quite flat, thus producing a large aspect ratio.

Error

The three measures we have looked at thus far have all been preprocessing measures. That is, the researcher would look at these measures while creating the mesh and before computing the solution using the finite element method. We now turn to look at a postprocessing measure. As the finite element method provides only an approximation to the true solution, the researcher needs to determine what the error is between the approximation and true solution. Through adaptive mesh refinement and derefinement, the discretization error is reduced such that the approximation will converge towards the true solution.

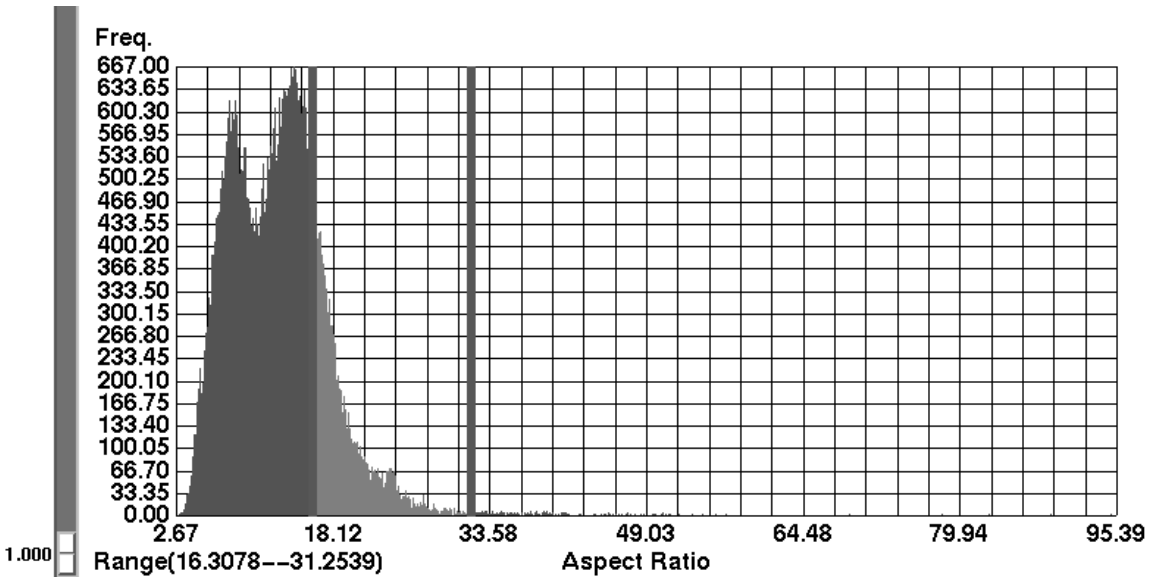

Figure 5: Histogram of the aspect ratios of a mesh with 10,000 nodes.

Typically, when scientists run a finite element analysis, the output contains a long list of numbers indicating the error. Although they can refine the mesh and watch the errors decrease, this method of verification is obviously not optimal. With the interactive histogram, the user will be able to see the range of errors easily and find where the largest errors are occurring.

Being able to examine these various measures gives the user a clear understanding of the mesh. These measures allow the user to isolate anomalous elements and then, if necessary, to refine the mesh in the proper places. When combined with the manipulative techniques, the user has full control of the mesh and is able to study the spatial and quantitative relationships of the elements.

We illustrate this technique in Figs. 6,7,8 for the simulation of electrical current flow due to internally implanted defibrillation electrodes within a large-scale model of the human thorax. Here Fig. 6 shows the tetrahedral elements that fall above the error criteria and need to be refined after a finite element approximation using the original mesh. Figs. 7 and 8 show the elements that still need refinement after two more applications of an h type finite element refinement scheme.

2.3 Editing Techniques

Another useful feature of the *MeshView* tool is its editing capabilities. The program allows the user to add a new node, to delete an element and its four vertices, or to delete a single node. Previously, researchers did not have an interactive way of editing meshes while simultaneously seeing where problems spots were. A user of this new program can first use the histograms to isolate distorted elements and then, depending on the problem, refine the mesh by either adding a node or deleting one or more nodes. The program then re-tetrahedralizes the mesh based on the new point set. When the user is finished editing the mesh, the program will output the new point set and connectivities for later use.

Adding a node

One way of adjusting an element with an overlarge volume is to subdivide the element into smaller tetrahedra. The program allows the user to specify an element that needs editing, and will then add a new node at the centroid of the node. To re-tetrahedralize the mesh, the program simply sends the new point and the existing mesh to a procedure that implements Watson's algorithm [16].

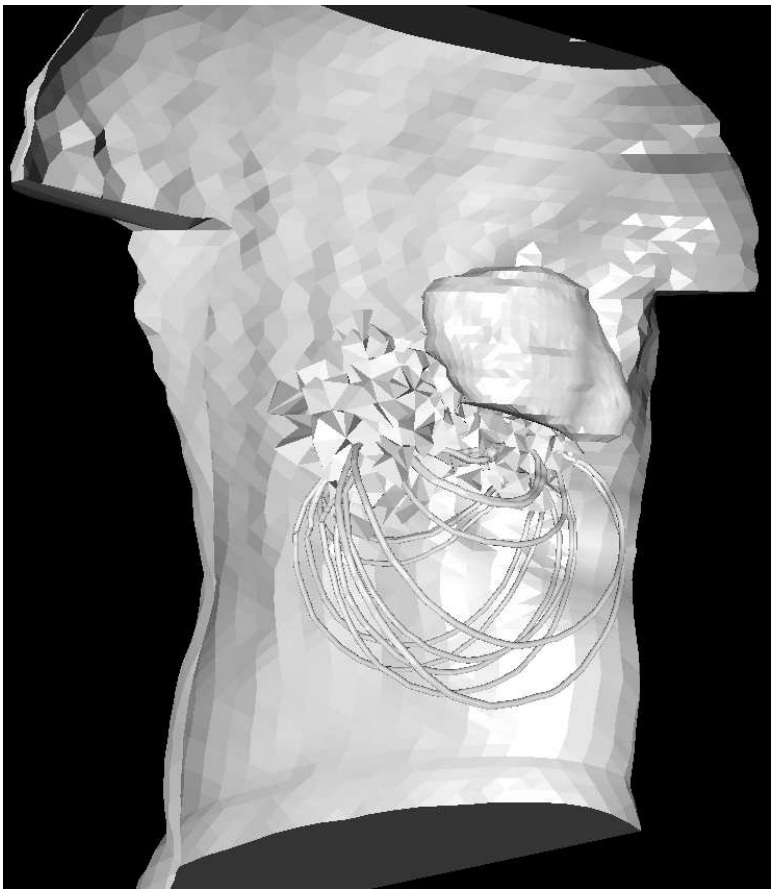

Figure 6: Tetrahedral elements that need refinement within a model of the human thorax for large-scale bioelectric field simulations. Shown here is a cutaway of the thorax, the heart surface and the electric current flow lines for a simulation of internal cardiac defibrillation.

Deleting nodes

The program provides two different options for deleting nodes: it will delete the specified element's four nodes, or it will delete only the node closest to the 3D picker, which the user specifies. Deleting nodes is useful for fixing elements which have bad aspect ratios or near zero volumes. After a node has been deleted, the program then re-tetrahedralize the mesh.

With the two exploratory approaches plus the editing capabilities, we have developed a powerful tool for mesh visualization. The user is able to verify the integrity of the mesh, isolate and correct problem areas, and gain a better understanding of the mesh as a whole.

3 Implementation

MeshView is a graphical analysis tool that is one module within a larger computational steering program, SCIRun [10]. SCIRun is written in C++, TCL/Tk [17], and OpenGL. It currently runs on Silicon Graphics (SGI) and SUN workstations. SCIRun is a dataflow program that allows the user to solve interactively computational problems and visualize them. The program is made up of modules that the user can link together into a "map" to solve and display particular problems. A map typically consists of one or more of each of an input module, a computational module, and an output module.

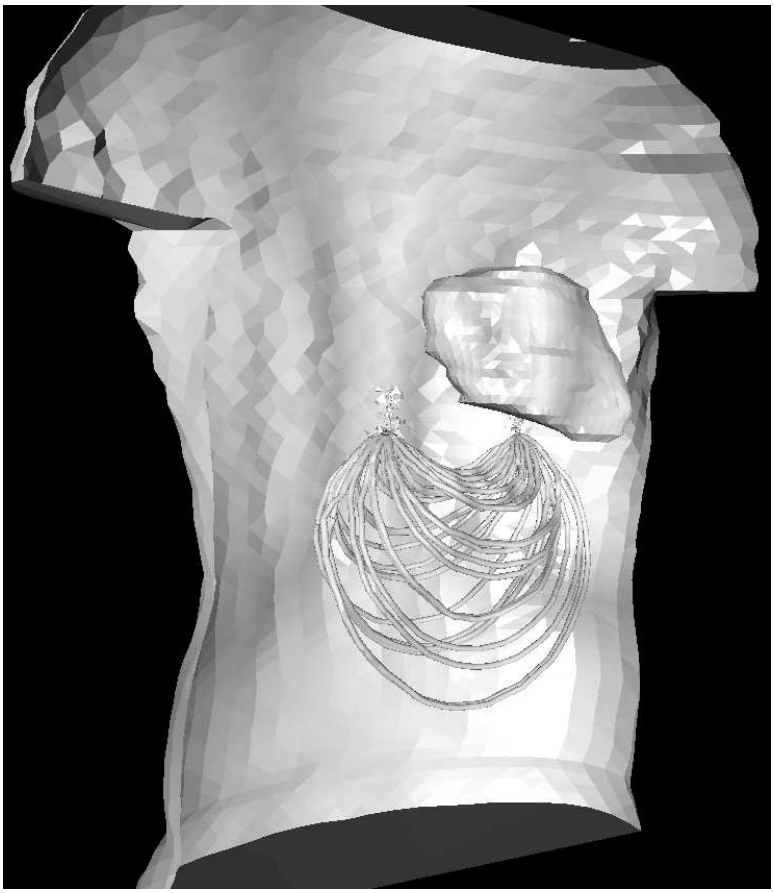

Figure 7: Elements that still need refinement after a single application of an h refinement.

The *MeshView* tool has three modules: an input module that reads in a mesh, the *MeshView* module that contains the interface for interacting with the mesh, and the display window for visualizing the mesh as seen in Fig. 9. While *MeshView* expects the input to be of a certain form, additional input modules can be attached to *MeshView* that would convert meshes in a different format into the SCIRun mesh format.

Size versus neighbors

4 Statistics

We tested the *MeshView* program on two different workstations, a Silicon Graphics Indigo2 workstation with a 200Mhz R4400 processor, 128 Mbytes main memory, and a 1 Mbyte cache and a Silicon Graphics Indigo 2 workstation with a 100Mhz R4000 processor, 96 Mbytes main memory, and a 1 Mbyte cache. We ran several test meshes, which ranged from a few thousand elements to a hundred thousand elements. Because of the way we have currently implemented the data structures, the majority of the time is spent in preprocessing the data into the different levels. While the system is slow in setting up the levels, interaction between levels is real-time. The drawback, though, comes when the user changes the seed. When the seed is changed, the level information must again be processed. Additionally, to determine the new seed element, the system must spend time computing which element the cross-hair widget is in. However, this action does not need to occur in real time, and we have found the longer processing time to be acceptable. Table I shows the timings for the growth algorithm, both in selecting a new seed and in changing levels. Table II shows the average time for doing the clipping with only the first level shown, one-fourth of all levels shown, one-half of all levels, three-fourths, and then all levels. These timings are quite rough, as the time is dependent on

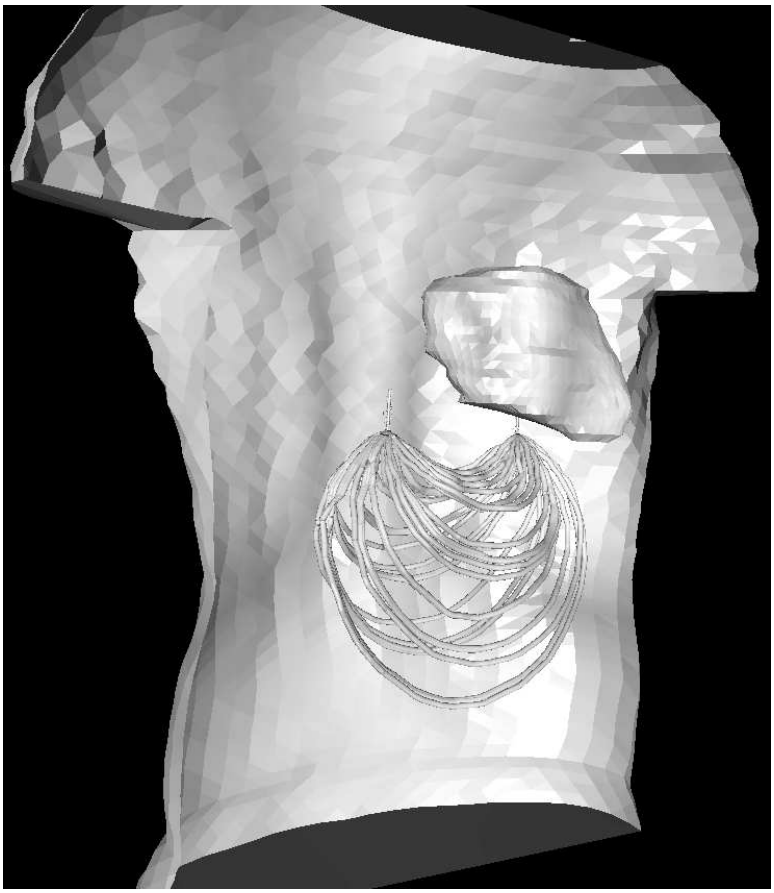

Figure 8: The final solution showing very few elements that still need refinement. Notice the difference in the current flow lines between the first and last simulation.

just how much is being clipped. Table III displays the time it took to calculate the various measures. Again, these calculations are done only once, and then the data stored for future use. Lastly, Table IV shows how long it takes to add or delete a node and then re-tessellate. For all of the tables, the first number is the time it took on the R4400 processor and the second number is the time it took on the R4000 processor.

5 Conclusion

We have presented here a graphical analysis tool, *MeshView*, for interactive exploration and analysis of 3D unstructured tetrahedral meshes. Through our innovative manipulative techniques, interactive histograms for quantitative measures, and editing capabilities, we have provided a powerful tool for analysis and verification of finite element meshes. Good visualization is imperative for understanding the information displayed in the meshes. Beyond visually assessing the mesh, the researcher may also need to understand regional changes due to adaptive mesh refinements. We have provided the capability for the researcher to narrow in on these specific regions and to confirm the mesh's integrity or edit regions if necessary.

Our graphical analysis tool is continuously evolving. There are several changes and additions to be made to the editing capabilities. Currently, the user can add a new node only in the center of the selected tetrahedron. As we improve the SCIRun environment by adding dialboxes and a 3D mouse, we will gain the ability to accurately pick a point in 3D space. This will allow us to change the program so that the user can add a new node anywhere in the mesh. These improvements in input devices will also allow us to move nodes and manually redraw connections, two features that we are currently unable to provide.

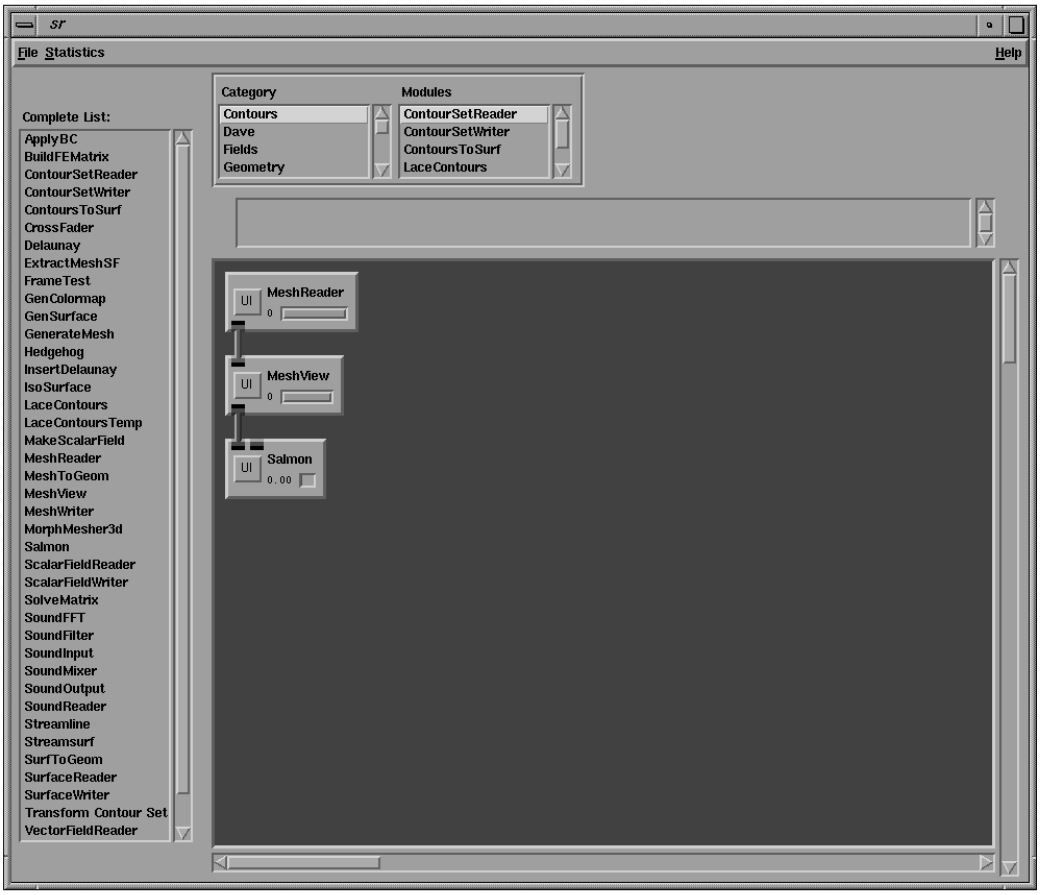

Figure 9: SCIRun program with the map of the MeshView tool

We also have plans to improve the algorithm used for deleting nodes in the mesh. Currently, when the user deletes a node, the program must retetrahedralize the entire mesh, as we do not have an algorithm for determining how far-reaching an effect this deletion will have on the mesh. Implementing an algorithm to localize the retetrahedralization will greatly increase the speed of the deleting options, allowing *MeshView* to be used with large meshes. Another enhancement planned for the editing capabilities of *MeshView* is to allow the user to specify the algorithm used for re-tetrahedralization of the mesh. Currently, all edited meshes are modified using the Delaunay algorithm, even though the original mesh may not have been created in this fashion. Specification of the meshing algorithm would give the user more control over the mesh.

Currently, the quantitative measures of *MeshView* analyze only individual elements or an element and its four face neighbors. Possible extensions to this analysis include measures based on all elements that share a vertex or doing localized measure analysis. This will give the user a better idea of the state of a region of the mesh, as opposed to analyzing on an element-by-element basis. Another possibility for improvement would be to tie together the quantitative analysis with the editing options. Users could specify a range on the histogram and then indicate that they wanted to delete all those elements or insert a new node inside all of the elements. Thus, users can quickly and efficiently fix all elements within a certain "problem" range.

Table 1: Timings (in seconds) for the *MeshView* program on an R4400 and an R4000.

Points	**Elements**	**Changing Levels**	**Changing seed**
100	477	0.015 / 0.04	0.04 / 0.06
1000	6160	0.11 / 0.20	0.15 / 0.33
5000	32594	0.30 / 0.63	0.84 / 1.83
10000	65813	0.42 / 0.98	1.64 / 3.70
15000	98810	0.63 / 1.30	2.60 / 5.70
20000	132344	0.73 / 1.56	3.46 / 7.83

Table 2: Timings (in seconds) for the clipping planes.

Elements	1 level	1/4 levels	1/2 levels	3/4 levels	All levels
477	0.0075 / 0.015	0.01 / 0.02	0.015 / 0.03	0.02 / 0.04	0.023 / 0.05
6160	0.01 / 0.02	0.02 / 0.06	0.08 / 0.16	0.10 / 0.24	0.12 / 0.28
32594	0.03 / 0.07	0.09 / 0.16	0.34 / 0.79	0.53 / 1.15	0.56 / 1.28
65813	0.05 / 0.11	0.16 / 0.42	0.60 / 1.52	1.00 / 2.43	1.11 / 2.72
98810	0.08 / 0.16	0.25 / 0.53	0.95 / 2.21	1.60 / 3.66	1.77 / 3.91
132344	0.11 / 0.20	0.36 / 0.82	1.25 / 2.95	2.12 / 5.35	2.31 / 5.92

6 Acknowledgements

This work was supported in part by awards from the NIH and the NSF. The authors would like to thank Katharine Coles, Ruth Klepfer, John Schmidt, and Yarden Livnat for their helpful comments and suggestions.

References

[1] S. Hammond and R. Schreiber, "Mapping unstructured grid problems to the connection machine.", in *Unstructured Scientific Computation on Scalable Multiprocessors*, P. Mehrotra, J. Saltaz, and R. Voigt, Eds., pp. 11–30. MIT Press, 1992.

[2] A. Shostko and R. Lohner, "Three-dimensional parallel unstructured grid generation", *AIAA*, 1994.

[3] C.R. Johnson and R.S. MacLeod, "Nonuniform spatial mesh adaption using a posteriori error estimate: Applications to forward and inverse problems.", *Applied Numerical Mathematics*, vol. 14, pp. 311–326, 1994.

[4] F. Yu and C.R. Johnson, "An automatic adaptive refinement and derefinement method.", in *Proceedings of the 14th IMACS World Congress*, 1994, pp. 1555–1557.

[5] D. Speray and S. Kennon, "Volume probes: Interactive data exploration on arbitrary grids", *Computer Graphics*, vol. 24, no. 5, pp. 5–12, 1990.

[6] C.R. Johnson, R.S. MacLeod, and M.A. Matheson, "Computer simulations reveal complexity of electrical activity in the human thorax", *Comp. in Physics*, vol. 6, no. 3, pp. 230–237, May/June 1992.

[7] C.R. Johnson, R.S. MacLeod, and M.A. Matheson, "Computational medicine: Bioelectric field problems", *IEEE COMPUTER*, pp. 59–67, Oct., 1993.

[8] E.S. Brugger, A. Leibee, and J.W. Long, *MeshTV User's Manual*, Lawrence Livermore National Laboratory, May 1994.

[9] R. Rainsberger, *TrueGrid User's Manual*, XYZ Scientific, 1990.

[10] C.R. Johnson and S.G. Parker, "A computational steering model applied to problems in medicine", in *Supercomputing '94*. 1994, pp. 540–549, IEEE Press.

[11] C.R. Johnson and S.G. Parker, "Applications in computational medicine using SCIRun: A computational steering programming environment", in *Supercomputer*, 1995, pp. 2–19.

[12] D. Kahaner, C. Moler, and S. Nash, *Numerical Methods and Software*, Prentice Hall, 1989.

[13] G.H. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 2nd edition, 1990.

Table 3: Timings (in seconds) for calculating measures.

Elements	Volume	Aspect Ratio	Size v Neighbor
477	0.10 / 0.12	0.05 / 0.11	0.05 / 0.07
6160	0.50 / 1.08	0.39 / 0.93	0.30 / 0.75
32594	2.79 / 5.54	2.39 / 5.34	1.94 / 4.32
65813	5.70 / 10.93	5.20 / 10.68	4.16 / 8.40
98810	9.05 / 18.07	8.32 / 17.23	6.66 / 13.86
132344	12.67 / 25.05	11.72 / 24.23	9.71 / 19.15

Table 4: Timings (in seconds) for editing.

Elements	Adding a node	Deleting a node
477	0.07 / 0.13	0.23 / 0.53
6160	0.67 / 1.49	2.93 / 7.10
32594	4.03 / 8.40	18.2 / 43.1
65813	8.59 / 18.2	42.27 / 97
98810	13.9 / 28.4	69 / 134
132344	19.23 / 37.0	104 / 213

[14] C.R. Johnson, *Advanced Methods in Scientific Computing*, Department of Computer Science, 1995, Lecture Notes.

[15] O. Bertrand, "3D finite element method in brain electrical activity studies", in *Biomagnetic Localization and 3D Modeling*, J. Nenonen, H.M. Rajala, and T. Katila, Eds., pp. 154–171. Helsinki University of Technology, 1991.

[16] D.F. Watson, "Computing the n-dimensional Delaunay Tesselation with applications to Voronoi polytopes", *Computer Journal*, vol. 24, no. 2, pp. 167–172, 1981.

[17] J.K. Ousterhout, *Tcl and the Tk Toolkit*, Addison-Wesley, 1994.