Visualization Tools for Computational Electrocardiology

Robert S. MacLeod, Ph.D.
Nora Eccles Harrison
CVRTI
macleod@vissgi.cvrti.utah.edu

Christopher R. Johnson, Ph.D.
Dept. of
Computer Science
crj@cs.utah.edu

Mike A. Matheson, M.Sc.
Utah Supercomputing
Institute and IBM
ibmmam@snow.usi.utah.edu

University of Utah, Salt Lake City, UT 84112

ABSTRACT

We have developed a suite of visualization and model construction tools for use in computing voltages and currents in the human thorax due to cardiac electrical activity. The programs support interactive digitization of MRI images, automatic generation, manipulation and editing of three-dimensional meshes, and display of both surface and volume potential distributions and volume current vectors.

2. INTRODUCTION

This paper contains descriptions of two visualization systems which we have developed for studying problems in computational electrocardiography, that is, the biophysical investigation of electrical fields in the thorax due to current sources in the heart[1, 2, **?**, **?**, **?**]. These tools have proved invaluable at virtually all stages of the research, from the digitization of the MRI images which formed the basis of our geometrical model of the human thorax[3] through the examination and editing of the resulting triangulated surfaces and the tetrahedralized volumes[2, **?**], to, finally, the display of both the input and output data from our simulations[**?**, **?**]. At the time we began this project, none of the available software packages provided the performance and flexibility that we required. Thanks to the power and versatility of the present generation of UNIX workstations and the availability of rich graphics software libraries, development of interactive, three-dimensional graphics software, while still not trivial, has become manageable for scientists with reasonable programming skills.

The visualization tools we have developed can be divided into two categories, a set of interactive programs for display and editing of geometry and surface-bound data, created at the CVRTI, and the "IBM visualization system", a set of distributed, batch-mode utilities for generating rendered images. Programs in the CVRTI visualization system were written in C, using the Graphics Library (GL) from Silicon Graphics, executed either on a Silicon Graphics 210/4D VGX or on an IBM RS/6000 520 workstation. Original versions of some of the CVRTI tools were written for Macintosh II computers, and Evans and Sutherland PS-390 graphics terminals. The IBM visualization system also uses the GL programming interface, although some display routines also work under the "Programmer's Hierarchical Interactive Graphics System" (PHIGS), and its X-windows implementation, PEX.

In this paper, we also describe some of the details of the construction of realistic, three-dimensional geometric models based on scanned image data. The purpose of these models was to develop numerical solutions to problems in electrocardiology arising from the bioelectric activity of the heart. Here, too, due

to the lack of available software, we have developed our own set of tools which allow us to create complete triangulated and/or tetrahedralized meshes from digitized outlines of surfaces and boundaries (in our case, the human body) with a high degree of flexibility and control.

3. MODEL CONSTRUCTION

3.1 From MRI to points

Construction of our model began with the nuclear resonance imaging of a healthy subject, with scans spaced at 5 mm; all frames which included any part of the heart were gated to the subject's heart beat. Images were stored on digital tape, transferred to a Macintosh II, and manually digitized[3]. Since it is the electrical, or more precisely, the passive conductive properties of the tissue that determine the potential and current distributions in the thorax, we digitized the surfaces that divided tissues of different conductivities: the outer boundary of the torso; the subcutaneous fat/muscle interface; the inner border of the skeletal muscle; the lungs; and the outer (epicardial) surface of the heart.

We used the points digitized from each image to construct parametric, bi-cubic spline equations for the outline of each surface, and then regenerated node points at a constant (and adjustable) spatial separation. This allowed us to control the internodal separation and apply some smoothing to the inevitably jagged hand-digitized point set. From each image, the result was a set of evenly spaced, closed loops, one for each surface, which we refer to as a 'layer'.

3.2 Surface Segmentation

In order to apply the boundary element method (BEM) to realistic geometric models (see sections 4.1), and also to create graphical representations of the data (see section 4.3), it is necessary to connect the points into polygonal surface descriptions. Triangular elements make an excellent choice for surfaces as irregular as those in the human body, and their use is supported by criteria by which "optimal" triangles can be constructed from given point sets[4, 5, 6].

Lacing triangles While the general problem of creating an optimal triangulation from points in three dimensions is quite complex, significantly simpler approaches can be used if the points are constrained to lie in planes, as they almost always do in data collected from imaging systems. If the surfaces to be constructed lie more or less perpendicular to the image planes, then the job is reduced to one of "lacing" the points in adjacent layers into optimal triangles. Algorithms for performing the lacing operations have been described[7, 8, 9] and shown to work, even in situations in which the surface bifurcates, as in, for example, the structure of blood vessels[8, 9].

In order to apply these techniques to the construction of a model of the human torso, we combined the points from adjacent layers into three-dimensional "slices", which were the input for our lacing programs. Triangulation of each slice involved the following steps:

1. Determine an anatomically based starting point and locate the nearest node from each layer of the slice to that starting point. We typically chose the anterior midline of the body for a starting location.
2. Order the points for each surface in each layer in such a way that they form a continuous sequence, commencing at the starting point and proceeding in a common direction (we chose counterclockwise,

as viewed cranio-caudally) around the surface. This was typically performed during the cubic spline fitting of the sampled points.

3. Determine a starting point on one of the layers, and find the nearest point in the second layer; join these to define the first connective segment.

4. Proceeding to the next point in each layer, determine which of the two possible pairs of triangles that can be formed by joining them with the endpoints of the first segment yield the shortest diagonal (and hence the most optimal triangulation in terms of maximizing the size of the smallest angle in each triangle[5]).

5. With this pair of triangles defined, a new connective segment is formed, and the previous step can be repeated until all points have been included, and the surface completely triangulated.

Figure 1 shows this process in a simple case with 10 points in each layer of a slice. The starting point is at point 1, which is connected to point 11. Testing of the two possible diagonals (from points 1 to 12 or points 2 to 11), shows that the latter produces triangles with larger minimum angles and hence is the proper choice. A second triangle, consisting of points 2, 11, and 12 is also immediately constructed and the segment between points 2 and 12 becomes the new starting point for further lacing.

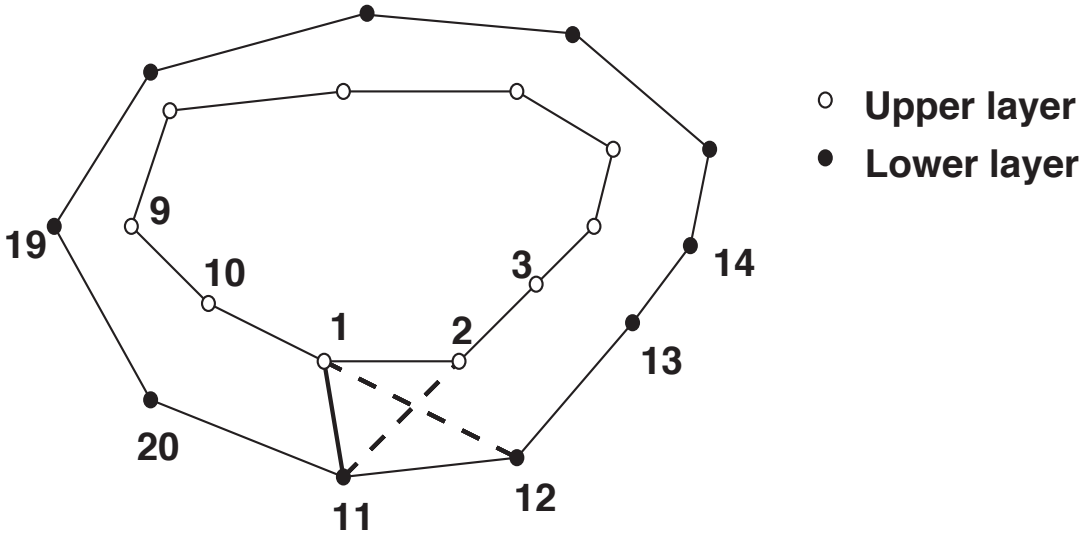

Figure 1: Example of lacing two layers to triangulate the surface enclosing the resulting slice. Points form the upper layer are shown as empty circles, those from the lower as filled circles.

While this basic algorithm will work *most* of the time, there are a number of situations in which additional mechanisms are required. In some case of extremely irregular surfaces the lacing scheme must be augmented with some checks for triangles which are too large, or must be formed, not between nodes on different layers, but in the plane of one of the layers. Errors frequently occur in regions of significant concavity, in which triangle segments run outside the hull of the body, as, for example, in the crescent-shaped regions of the lower lungs. Simple examples of these situations are given in Figure 2. In panel **A** the upper layer describes a smaller surface than the lower layer, and a single point (point P in the figure) is shared by seven triangles, instead of two or three, as for all the other points. Panel **B** shows a situation in which erroneous triangles (shown in thicker, dashed lines) are constructed across a region which is outside the actual surface. Such triangles must be detected and removed, ideally without user input.

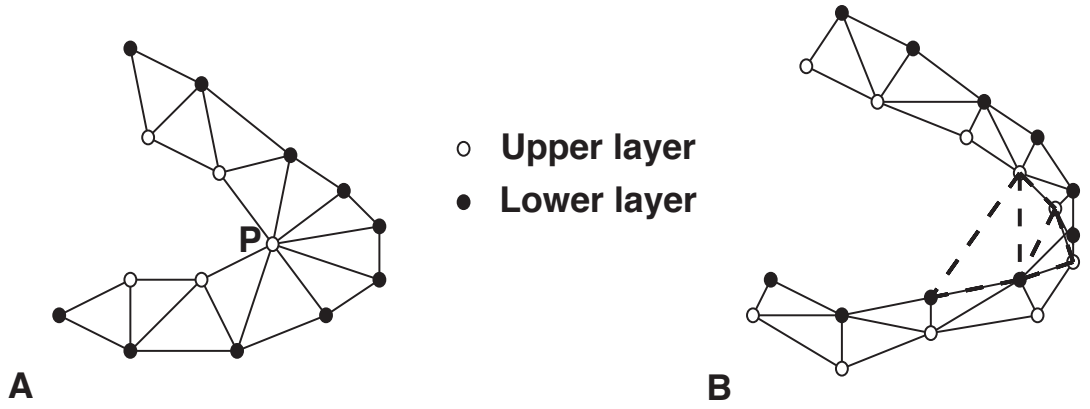

Figure 2: Two examples of special cases in lacing triangles. In **A**, the two layers join to form a very flat surface since one layer does not lie directly over its neighbor. In panel **B**, a sharply curved surface can result in erroneous triangles which lie outside the actual surface (shown as dashed lines.)

Inside/Outside algorithm Determining when a triangle lay outside the surface was one of several related problems that arose during the model construction (and will be described in more detail below) in which the basic question was whether a point was *inside* or *outside* a surface. To solve this, we again made use of the fact that all the points in our tomographic data lie in discrete planes. If a point lies inside the curve defining the intersection of a surface and a plane (the "surface curve"), then the algebraic sum of all the angles about the point which are formed between adjacent points on the curve, should be equal to 2π. For a point outside the curve (and hence the surface partially defined by the curve), the sum of the angles is zero. Using this fact, and provided that the points have already been ordered in a strict sequence of first-order neighbors, we can construct a set of tests with which the location of a point, relative to a surface, can be determined. This same concept can be extended directly to three dimensions if, instead of planar angles, the sum of the *solid angles* about a point is computed. However, this calculation requires knowledge of the outward normals on an already complete surface tessellation and is therefore unsuited to this application. Once completed, however, a computation of the sum of the solid angles is an excellent means of checking for complete closure and consistency of the outward normals.

Figure 3 shows the principle behind the inside/outside algorithm, in which the sum of all angles θ_i over the ordered set of nodes about the point P is equal to 2π if P is inside the curve, and 0 if P is outside the curve.

In the particular application of triangulation, in which we wish to detect and remove triangles which lie outside the concave surface, there is, however, potential for error. Since any point that is tested, for example, the midpoint of each side of the triangle under examination, actually lies between the two planes of the data points, we must project it onto one or the other of the planes and test the projection of the point relative to the surface curve. Sufficient practical accuracy can be achieved, however, by testing each such point, and perhaps several others in the triangle, projected onto *both* planes. We have found that checking the location of the projected centroid of the triangle against surface curves on both layers (and eliminating the triangle if the centroid lies outside either) is adequate for most purposes.

Completing the model Once laced into multiple slices, the points and triangle connectivities must be concatenated to make a complete surface description. We usually perform this operation on a surface-by-surface basis so that the end result is a set of points and connectivities for each of the surfaces in

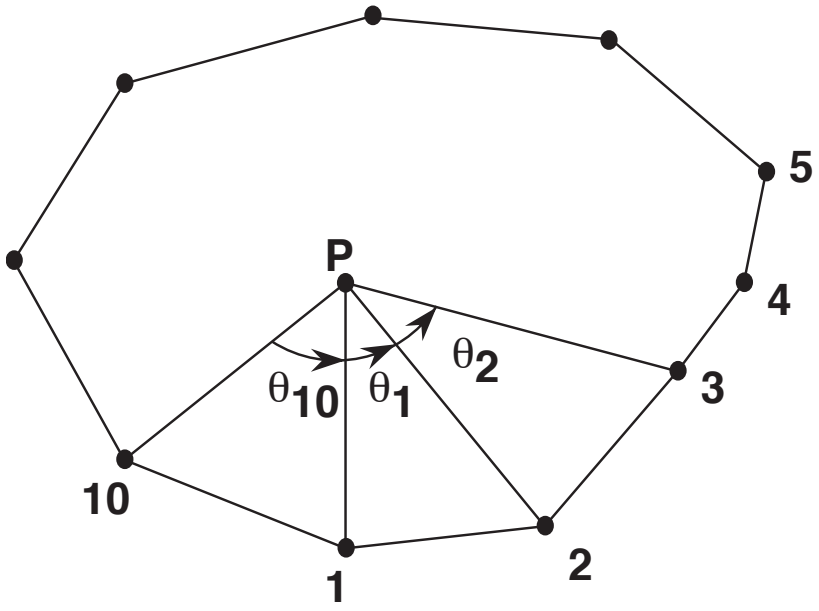

Figure 3: The inside/outside algorithm, used to determine if point P lies inside our outside the outline formed by joining the order set of nodes i. If P is inside, $\sum_{i=1}^{i=N} \theta_i = 2\pi$; if P is outside, the sum is equal to zero.

the complete model. This allows for easy selection of the surfaces that are to be included in a computation and/or visualization. Before they are ready to use, however, there are two final steps in the model construction process.

The first involves connecting the points which form the 'ends' or 'caps' of the model, that is, the first and last layer of each surface. This can be performed in one of two ways, either by triangulating the points in the layer, or by adding a grid of points to fill in the region defined by the points in the end layers, and then triangulating the resulting augmented layer in two dimensions. We employ both approaches, depending on the size of the opening at each end and the desired density and spatial resolution of the model we are constructing. If a grid is added, its density is, likewise, dependent on the desired resolution of the model. The triangulation of the resulting two-dimensional point set is a straightforward problem and numerous programs exist for this purpose, some even in the public domain (*e.g., voronoi* from the netlib collection, available via electronic mail from netlib@ornl.gov). What is usually missing from these algorithms, however, is a provision for concave curves, in which case triangles are constructed which lie outside the curve. Hence we have implemented a check of all the triangles formed by the two-dimensional triangulation using the inside/outside test described above, applied to the centroid of each triangle.

The final step in constructing a surface model of the geometry involves setting the direction of the outward normal vectors to each of the triangular elements. Finding a normal is easily done by computing the cross product of any two sides of the triangle. Ambiguity remains, however, in the final choice of which of two opposite directions is outward, and which inward. This can be resolved in most cases by applying, once again, a test to the endpoints of both candidate normal vectors to determine which lies inside, and which outside the surface. We add both normal vectors to each of the three vertices of the triangle, then test the endpoints relative to the surface to which the vertex belongs, until we achieve an unambiguous result of one endpoint found inside, the other outside the surface. The coding of this information into the model itself occurs in the order in which the points of each triangle are recorded: the points are always

taken in counterclockwise order, when viewed from the outside of the surface. As a final, complete check of the integrity of the model, we select a point inside each surface, and compute the total solid angle about that point. If the surface description is complete (no holes), and all the outward normals are correct, the result must be 4π.

Manual editing At some stage of the automatic segmentation of the model, there inevitably arise situations in which the algorithms fail. Our pragmatic response to this situation was to develop programs for manual tuning of the digitization and triangulation of the geometrical data. In constructing the human torso model, we found it necessary to align and adjust whole layers because of shifts in the subject's position between scans and movement noise induced by breathing. Some correction was possible using spatial smoothing algorithms but we usually required a final pass with an interactive graphics program which allowed us to move individual points within the layers. In order to adjust the triangulation we used an interactive triangle editor developed at the CVRTI that allowed the user to manipulate and examine the triangle mesh, and delete or rebuild any triangles. In our experience, the time required for such manual adjustment of the geometry was minimal compared with the effort required to develop completely correct, universal, and robust algorithms for automated model construction.

3.3 Volume Segmentation

To apply three-dimensional finite element methods to a problem the solution domain must be discretized into a mesh of volume elements. Many problem domains can be described in terms of analytic functions, and yield to automated mesh generation schemes which produce grids which are regularly spaced, or *structured*. Models which are based on a set of fixed sample points, on the other hand, are known as *unstructured* meshes. While structured meshes are easier to construct and require less computer memory (since they can be described parametrically), they are poorly suited to irregularly-shaped objects, and especially poorly suited to follow realistic boundaries between regions within the domain (*e.g.,* the organs in a model of the human body). There are, however, techniques by which a set of three-dimensional points can be joined in some optimal sense to form a mesh of volume elements and many of these techniques are based on an extension of the Delaunay criteria to three dimensions[10, 11].

Our approach was something of a hybrid method, a combination of a regular grid applied within the irregular surface outlines from the digitized MRI images. Starting with the same point set used for surface triangulation, we first added a planar grid of points to each of the layers, then removed all points that were within some tolerance of an existing surface or boundary. The spacing of the grid determined the overall density of the mesh. We then concatenated adjacent pairs of grided layers to form three-dimensional slices, which were the input data to a program which automatically generated optimal (in the maximum smallest angle sense of Delaunay) tetrahedral elements. Subsequent quality control of the tetrahedralization consisted of removing tetrahedra which were flat, too large (and hence contained other smaller tetrahedra), or which spanned concave regions of the outer boundaries of the domain. To detect elements which lay outside the domain the location of the midpoint of each side of the tetrahedron was tested against the surface enclosing the domain, using, once again, the inside/outside check; all tetrahedra any part of which lay outside the domain were then removed.

One of the major advantages of the finite element technique, over, for example, the boundary element method, is that the effect of inhomogeneities, even those of an anisotropic nature, can be included directly in the formulation of the problem. In the case of electrical activity in the human torso, there exist different tissue types which possess different passive conductivity characteristics, and hence affect the distribution of electric potential and current in the body. The associated step in constructing our finite element model

of the human torso was, therefore, to identify those elements which lay in certain regions of the body, and to assign each a conductivity tensor. Locating the elements was performed by projecting the centroid of the tetrahedra onto the nearest surface and then subjecting that point to a chain of inside/outside tests on each of the boundary surfaces of the model until the its relative position was determined. For example, a point which lay inside the body surface but outside the boundary between subcutaneous fat and skeletal muscle, was part of the fat layer; a point which was inside the body surface, the fat/muscle boundary, and the inner boundary of the muscle, but outside of all other surfaces, was not part of any distinct tissue group, and was assigned a conductivity equal to the mean for the thorax. We did not assign numerical conductivity values until execution of the finite element solver, employing a look-up table based on group numbers assigned to each tissue region.

The final step in the three-dimensional model construction was the concatenation of the tetrahedralized slices into a complete geometrical description of points, connectivities, and tissue regions. In order to facilitate transfer of the simulation results back to the surface description of the model, we kept tables which described the location of each point, in terms of both the volume and surface point numbers.

We have constructed two torso models from a single set of MRI scans, one which uses all the images and maintains an internodal spacing of about 5 mm (high resolution model), the other utilizing every second MRI scan, with a mean internodal spacing of 11 mm (medium resolution). The models contain 42,000 (11,000) surface nodes joined to form 85,000 (19,000) triangles and 82,000 (22,000) total nodes to form 500,000 (130,000) tetrahedra, where the first numbers are for the high resolution and the number in parentheses are for the medium resolution model.

3.4 Visualization tools

The purpose of the interactive visualization tools we developed for constructing the model was to view the geometry and evaluate the success of the segmentation. For the automated tessellation, we needed to evaluate the efficacy of control parameters by displaying the mesh, before we applied necessary manual correction. Interactive manipulation of the large meshes from the torso models proved to be particularly taxing to the graphics hardware and required some care in developing the code. We restrained ourselves from applying elaborate shading techniques or lighting models and used the simplest possible interface so that new functions and features could be added quickly without the extra overhead of re-programming the user controls. While there were no graphics applications available at the time this project began which offered the necessary flexibility and programmability, we were able to utilize the impressive performance of the latest generation in modern graphics hardware, combined with a very complete and well established standard in high performance graphics software. We chose the Graphics Library (GL) from Silicon Graphics, teamed with VGX hardware on a Silicon Graphics 4D/210. Once developed, the programs could be easily ported to other computers which supported the GL standard, including the IBM RS/6000 Model 520 that was also used for model computations. Further details of the interactive programs are contained in section 4.2.

Once the model was constructed, we wished to display the result in a more elaborate fashion, to highlight certain features or describe facets of the model construction process. For this, we used the IBM Visualization System developed by one of the authors (MM), and described in some detail in section 4.3.

4. RESULTS

4.1 The Simulations

The primary purpose of developing our model of the human torso was to understand the relationship between the measurable voltages on the surface of the body, and their primary sources in the heart. When the known quantities are the potential distributions on the surface of the heart, one solves what is known as a "forward problem" in electrocardiology. Another form of the forward problem consists of calculating the distribution of electric potential and current throughout the thorax in response to defibrillation stimuli, which are used to resolve life-threatening irregularities in the heart rhythm. To characterize cardiac sources from knowledge of the potential on the surface of the body requires that an "inverse problem", which like many inverse problems is ill-posed, be solved (see [12] and [13] for excellent recent reviews). There are two numerical approaches commonly used to solve forward and inverse problems in bioelectricity, the finite element method (FEM) and the boundary element method (BEM). The relative merits of each in electrocardiographic inverse problems are described in[14] and we have developed solutions using both[1, **?**, **?**, 15, **?**, **?**].

4.2 Interactive Viewing of the Results

We developed an interactive program which allowed us to display geometrical models as points, wire meshes, or shaded surfaces and show simulation results as three-dimensional surface distributions and current vector fields. The overall structure of the program reflects a hierarchy of multiple, linked surfaces, each with multiple sets of data to be displayed on them. Each surface, and any data associated with the elements of the surface, can be displayed individually, or in any combination with other surfaces, with each surface in its own separate window, or combined in a single display of multiple surfaces. Multi-frame data sets can be stepped through in sequence, and are assumed to be linked across multiple surfaces, hence data on all surfaces update synchronously. Display of potential distributions consists of either color-code surfaces or contour lines, using bi-linear interpolation over the triangular surface elements.

Ubiquitous throughout the program is the ability to manipulate the geometry, and the data displayed on it, in real time. For this we have found the dial box input device to be ideal, providing a very tactile form of control, with the flexibility of many more, independent degrees of freedom than a mouse can offer. The standard layout for our current eight-dial unit is to have three dials assigned to translation, three to rotation, one to scaling, the the last one to manipulation of clipping planes, or z-buffer limits.

Figure **??** shows an example of a display of two surfaces, portions of the left and right lungs, using point, wire mesh, and shaded surface modes. Four concatenated slices from each lung are shown, with the left lung (right-hand side of the figure) drawn as a triangulated wire mesh and the right lung rendered with triangles which are visible only if they have outward normals directed *towards* the observer. The points (white dots) in the background indicate that the outside of this part of the surface is directed away from the observer and hence triangles are not drawn. The colors for this image were chosen in a sequence to allow the user the quickly determine whether the orientations of the triangle normals were correct and the coverage of the surface was complete. This entire image could be rotated, translated and clipped in real time, with triangles in the right lung appearing and disappearing as the outward side of the surface moved in and out of the observer's line of sight.

Standards One feature of data display that requires considerable flexibility is the scaling of colors or contours to represent quantitative information. Each branch of science has certain paradigms which, be

they particularly meritorious or otherwise, are viewed as standards by the trained observer and must be maintained if a new viewing tool is to meet widespread acceptance. Aspects of these paradigms include features as trivial as the layout of the display, and as important as the scaling models applied to the data or color map choices. The field of electrocardiology possesses such standards and these are reflected in our choice of display parameters. We have included options in the program to select scaling models (linear, exponential and logarithmic), scale scope (based on the current data frame or in some larger context, or even set directly by the user), scale mapping (*eg.,* enforce symmetry about the zero axis, or not), as well as the number of shades of color or contour lines employed and the color map used.

In Figure **??** is shown an example of the potential distribution over the torso and heart (epicardial) surfaces generated by the interactive display program. Red indicates positive potential and green negative, and a clipping plane applied to the anterior of the torso provides the view of the epicardial surface inside. Torso data were recorded using body surface mapping techniques, while the epicardial distribution was computed using an inverse solution based on the boundary element method[1, 15].

Displaying vector fields From the potentials computed at the nodes of the model, we can calculate the current density distribution within the thorax. The display of such vectorial data in three-dimensional volumes is neither simple, nor in any way standardized[16]. One approach that we have used to represent the vector field draws individual arrows, each directed along the local vector orientation with a length proportional to the local vector amplitude. This works reasonably well, as long as the spacing between individual vectors is generous, and their amplitudes do not vary by more than approximately a single order of magnitude, otherwise the images can become cluttered and overwhelmingly complex and thus defeat the original purpose of the scientific visualization. To reduce the number of individual vectors, averaging can first be carried out so that each arrow represents the mean direction and amplitude for a defined neighborhood. If the range of vector amplitudes is too large, each vector can be scaled, or even drawn at unit length and color-coded to represent its amplitude. We have used all of these approaches and found them to be useful, primarily because they could be easily coded, were relatively fast, and could be applied to any subregion of the total volume. The major disadvantages were that closed vector loops were difficult to follow and because of the high spatial density of our mesh, displays often became cluttered and confusing. We overcame this difficulty by using elements of the IBM Visualization System, as described in section 4.3 to produce views of the current distribution.

Program control Interactive control of the program was provided through the dials, a set of pull-down mouse menus (supported fully from within GL), and a number of single keystrokes. Arrow keys selected which surface(s) (up/down arrows) and which frame of data to display (left/right arrows). Program startup was initiated from the command line, with a lengthy list of UNIX-style arguments with which input files, default colors, window locations, and operation modes were set. Although this approach often resulted in multi-line commands, it proved very flexible because we could invoke the program either from a script file that included the commands, or from another program whose duty it would be to interrogate the user and construct the command line from the responses.

Animation Since temporal information is extremely useful in simulations of the electrical activity of the heart, we have produced animated sequences of potential distributions from both experimentally obtained and simulated data. The source of such animation was a set of images, stored as individual frames, which were first captured from the screen, and then recorded in sequence on video tape. GL provides a means of capturing a specified region of the screen and saving it in an image file so that once a particular temporal

data sequence was selected, the program could display the individual frames for each instant in time, and save them in files. Since each image took several seconds to capture and be stored, and thousands were required for a full animation, we normally set up such sequences under control of the visualization program and let the process run unattended.

4.3 Rendered Viewing of the Results

Rendering techniques A complete description of computer graphics and ray tracing is beyond the scope of this paper and numerous excellent sources exist elsewhere (*eg.,*[17]). However, the basic premise of ray tracing is that an observer sees a point of color on a surface that is the result of the interaction of the surface at that point with rays originating from elsewhere in the scene. This interaction is determined by tracing a ray from the eye position through each image pixel into the object data environment. At each object/ray intersection, a reflected and/or a refracted ray can be generated. The final pixel color is determined by traversing an intersection tree and computing the color contribution according to a shading model and material properties.

The greatest advantage of ray tracing is that this single methodology solves the problems of obtaining hidden surfaces, computing the shading due to direct and global illumination, and calculating the shadows. The disadvantage of ray tracing is the high processing cost that they incur. In an attempt to minimize computation time in this application, the rendering engine was distributed among workstations with a custom socket library interface. A reduction in time spent on computing ray and object intersections was accomplished by using a 'voxel'-based method which maintained information concerning the spatial positions of objects within the scene.

IBM Visualization System The complete IBM Visualization System system included a *database manager* for the consistent handling of a variety of input data types and computers; an *interface manager* which controlled feedback to programs; a *visualization manager* which was responsible for the creation of all images; and an *animation manager* used to create animated sequences. In order to make efficient use of decentralized computational resources, the IBM system was designed to function using shared memory and distributed systems.

The visualization manager possessed capabilities for interactive exploration of data such as arbitrary slicing, construction of contours and isosurfaces, display of vector fields, and particle tracking. Non-interactive images were created by either a high performance polygon renderer, named "HiPPR", or an integrated ray tracer/volume renderer, which we named "RaySpray". Common illumination models such as flat, Gouraud, and Phong were supported as well as numerous features for texture and environmental mapping, and motion blur. Typical image generation times for the high-resolution version of the model ranged from on the order of a minute of CPU for HiPPR, to just under an hour for RaySpray, on an IBM RS/6000-530 Workstation.

Animation methods The video supplement, which is available from the authors, demonstrates the combined use of high performance computers for simulation, and computer graphics and video production technology for the translation of numeric data into a dynamic visual media. The purpose of animation is to create an understandable visual form that represents large quantities of computed information in a manner that suggests spatial and temporal relationships. This allows one to achieve new insight into the physical phenomena being modeled through an improved understanding of complex data that typically have numerous time-dependent interrelated variables distributed through three-dimensional space. Through the

illusion of motion created by a sequence of images, any property of an object can be animated. Typically these properties include position and orientation, but also color, reflectance, transparency, texture, shape, *etc.,* as well as the position, color, intensity and other properties of the light sources.

In the IBM Visualization System, "key-framing" was used to animate the position and orientation of a 'virtual' camera. Camera positions were specified in Cartesian (x,y,z) coordinates and given frame numbers. The software used a double interpolation method with B-splines to determine the camera coordinates for each frame between two key-frames so that the camera arrived at the control point at the specified time. The spline functions were used to guarantee continuity of position, velocity, and acceleration throughout an object's path. Numerous features of the entire visualization system are evident in the video which was composed from RaySpray ray traced images and the CVRTI interactive graphics system.

Volume potentials and vector flow To view the potential distributions originating within the volume of the torso model we extracted isosurfaces from the data by locating a set of points with the same scalar value and joined them with triangular facets. We either displayed a series of color-coded equipotential surfaces as an animation, or selected three or four surfaces to be displayed together in a single image.

The most informative way to display current flow in the complete torso model was to follow current loops by first defining a start or "seed" point in the volume, locating the tetrahedron which contained the point, and then determining the local vector of the electric current. Through the use of a second-order Runge-Kutta integration method, a path which follows that current loop could be traced through space by controlling the step based on cell dimensions and current density strength. This is a direct analogy of particle tracing techniques used in computational fluid dynamics.

Figure **??** show an example of a rendered view of potential distributions on the anterior (right-hand image) and posterior (left-hand image) surfaces of a human torso model. In the left foreground is superimposed a single ECG lead indicating the time instant from which the potentials originate. These data were computed using a forward solution based on the finite element method, from voltages measured on the epicardial surface of the heart during open-chest surgery[**?**, **?**].

5. CONCLUSIONS

This suite of programs has considerably enhanced our ability to construct realistic, three-dimensional, geometric models and visualize simulation results. In retrospect, it is difficult to imagine how one would manage models of this size and complexity without the assistance of sophisticated graphics tools. The fact that we have developed the code ourselves also ensures that further development can, and no doubt will, occur as needs dictate. And although the design and implementation reflect our specific needs, many problems of this type exist in our field and related areas in which three-dimensional mesh construction is required.

ACKNOWLEDGEMENTS

We would like to thank Dr. Dennis Parker for his invaluable assistance in obtaining the nuclear magnetic images and Mr. Jim Cobb for assistance in the visualization of the model geometry. This research was supported in part by awards from the Nora Eccles Treadwell Foundation and the Richard A. and Nora Eccles Harrison Fund for Cardiovascular Research, by the Heart and Stroke Foundation of Canada, by the

Whitaker Foundation, and by a grant for computer time from the Utah Supercomputing Institute, which is funded by the State of Utah and the IBM Corporation.

References

[1] R.S. MacLeod, M.J. Gardner, R.G. MacDonald, M.A. Henderson, R.M. Miller, and B.M. Horáček. Validation of an electrocardiographic inverse solution using percutanteous transluminal coronary angioplasty. In *Proceedings of the IEEE Engineering in Medicine and Biology Society 12th Annual International Conference*, pages 533–534. IEEE Press, 1990.

[2] R.S. MacLeod, C.R. Johnson, and P.R. Ershler. Construction of an inhomogeneous model of the human torso for use in computational electrocardiography. In *Proceedings of the IEEE Engineering in Medicine and Biology Society 13th Annual International Conference*, pages 688–689. IEEE Press, 1991.

[3] P.R. Ershler, R.L. Lux, L.S. Green, G.R. Caputo, and D. Parker. Determination of 3-Dimensional torso, heart, and electrode geometries from magnetic resonance images. In *Proceedings of the IEEE Engineering in Medicine and Biology Society 10th Annual International Conference*, pages 121–122. IEEE, 1988.

[4] C.L. Lawson. Software for C1 surface interpolation. In *Mathematical Software II*, pages 161–194. Academic Press, New York, 1977.

[5] D.T. Lee and B.J. Schachter. Two algorithms for constructing a Delaunay triangulation. *Int. J. Comp. Inf. Sci.*, 9:219–242, 1980.

[6] B.A. Lewis and J.S. Robinson. Triangularization of planar regions with applications. *Comp. J*, 21:324–332, 1978.

[7] H. Christiansen. MOSAIC triangulation algorithm. In *MOVIE.BYU Program Manual*. Engineering Computer Graphics Laboratory, Brigham Young University, Provo, Utah, 1987.

[8] I. Vesely, B. Eickmeier, and G. Campbell. Automated 3-D reconstruction of vascular structures from high definition casts. *IEEE Trans Biomed. Eng.*, 38:1123–1129, 1991.

[9] R.R. Mercer, G.M. McCauley, and S. Anjilvel. Approximation of surfaces in a quantitative 3-D reconstruction system. *IEEE Trans Biomed. Eng.*, 37:1136–1146, 1990.

[10] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In F.K. Hwang and D.Z. Du, editors, *Computing in Euclidean Geomtry*. World Scientific, 1992.

[11] A. Bowyer. Computing dirichlet tesselations. *Computer J*, 24:162–166, 1981.

[12] R.M. Gulrajani, F.A. Roberge, and G.E. Mailloux. The forward problem of electrocardiography. In P.W. Macfarlane and T.D. Veitch Lawrie, editors, *Comprehensive Electrocardiology*, pages 197–236. Pergamon Press, Oxford, England, 1989.

[13] R.M. Gulrajani, F.A. Roberge, and P. Savard. The inverse problem of electrocardiography. In P.W. Macfarlane and T.D. Veitch Lawrie, editors, *Comprehensive Electrocardiology*, pages 237–288. Pergamon Press, Oxford, England, 1989.

[14] Y. Rudy and B.J. Messinger-Rapport. The inverse solution in electrocardiography: Solutions in terms of epicardial potentials. *Crit. Rev. Biomed. Eng.*, 16:215–268, 1988.

[15] R.S. MacLeod. *Percutaneous Transluminal Coronary Angioplasty as a Model of Cardiac Ischemia: Clinical and Modelling Studies*. PhD thesis, Dalhousie University, Halifax, N.S., Canada, 1990.

[16] J.L. Helman and L. Hesselink. Visualizing vector field topology in fluid flows. *IEEE Comp. Graph. & Applic.*, 11(3):36–46, 1991.

[17] J.D. Foley, A. van Dam, S.K. Feiner, and Hughes J.F. *Fundamentals of Interactive Computer Graphics*. The Systems Porgramming Series. Addison-Wesley, Reading, MA, USA, 1990.