APPENDIX .

Integrated Control of Distributed Volume Visualization Through the World-Wide-Web

Cheong S. Ang, M.S.
David C. Martin, M.S.
Michael D. Doyle, Ph.D.

University of California, San Francisco
Library and Center for Knowledge Management
530 Parnassus Avenue
San Francisco, California 94143-0840
contact: doyle@ckm.ucsf.edu

Abstract

The World-Wide-Web (W3) signals the entré of the digital library with the goal of providing immediate and ubiquitous access to digital information of any type from data repositories located throughout the world. The web's development enables not only effective access for the generic user, but also more efficient and timely information exchange among scientists and researchers. We have extended the repertoire of the web to include access to three-dimensional volume data sets with integrated control of a distributed client-server volume visualization system. This paper provides a brief background on the World-Wide-Web, an overview of the extensions necessary to support new data types and a description of the distributed visualization system.

1. Introduction

Advanced scanning devices, such as magnetic resonance imaging (MRI) and computer tomography (CT), have been widely used in the fields of medicine, quality assurance and meteorology [Pommert, Zandt, Hibbard]. The need to visualize resulting data has given rise to a wide variety of volume visualization techniques and computer graphics research groups have implemented a number of systems to provide volume visualization (e.g. AVS, ApE, Sunvision Voxel and 3D Viewnix)[Gerleg, Mercurio, Varde Wettering]. Previously these systems have depended on specialized graphics hardware for rendering and significant secondary storage for the data. The expense of these requirements has limited the ability of researchers to exchange findings. To overcome the barrier of cost and provide additional means for researchers to exchange and examine three-dimensional volume data, we have implemented a distributed volume visualization tool for general purpose hardware and integrated that visualization service with the distributed hypermedia [Flanders, Broering, Kiong, Robison, Story] system provided by the World-Wide-Web [Nickerson].

Our distributed volume visualization tool, VIS, utilizes a pool of general purpose workstations to generate three dimensional representations of volume data. The VIS tool provides integrated load-balancing across any number of heterogenous UNIX™ workstations (e.g. SGI, Sun, DEC, etc...) [Giertsen] taking advantage of the unused cycles that are generally available in academic and research environments. In addition, VIS supports specialized graphics hardware (e.g. the RealityEngine from Silicon Graphics) when available for real-time visualization.

Distributing information that includes volume data requires the integration of visualization with a document delivery mechanism. We have integrated VIS and volume data into the W3, taking advantage of the client-server architecture of W3 and its ability to access hypertext documents stored anywhere on the Internet [Obraszka, Nickersen]. We have enhanced the capabilities of the most popular W3 client, Mosaic [Andressen] from the National Center for Supercomputer Applications (NCSA), to support volume data and have defined an inter-client protocol for communication between VIS and Mosaic for volume visualization.

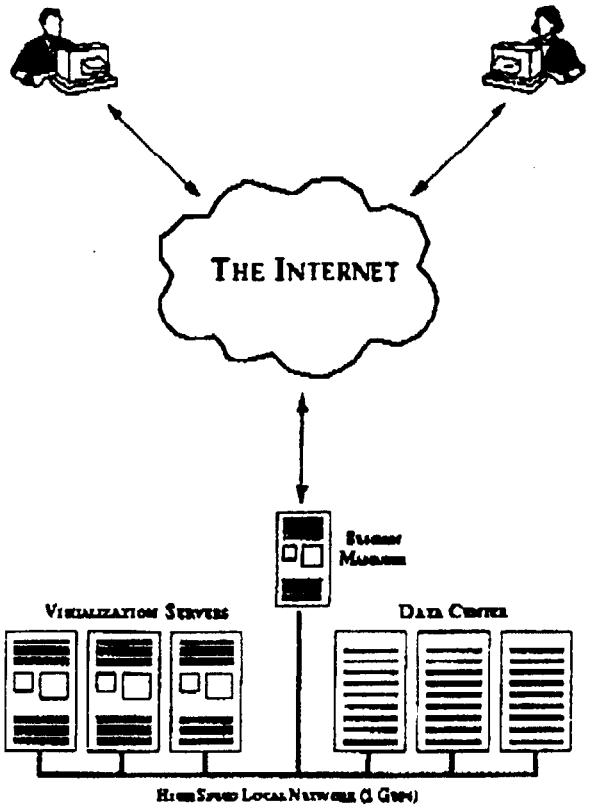

Figure 1: VIS client/server model.

1.1 The World-Wide-Web

The World-Wide-Web is a combination of a transfer protocol for hyper-text documents (HTTP) and a hyper-text mark-up language (HTML) [Nickersen]. The basic functionality of HTTP allows a client application to request a wide variety of data objects from a server. Objects are identified by a universal resource locator (URL)[Obraczka] that contains information sufficient to both locate and query a remote server. HTML documents are defined by a document type definition (DTD) of the Standard Generalized Mark-up Language (SGML). These documents are returned to W3 clients and are presented to the user. Users are able to interact with the document presentation, following hyper-links that lead to other HTML documents or data objects. The client application may also directly support other Internet services, such as FTP, Gopher, and WAIS, [Andreessen] or may utilize gateways that convert HTTP protocol requests and return HTML documents. In all interactions, however, the user is presented with a common resulting data format (HTML) and all links are accessible via URL's.

1.2 Mosaic

The National Center for Supercomputer Applications (NCSA) has developed one of the most functional and popular World-Wide-Web clients: Mosaic. This client is available via public FTP for the most popular computer interfaces (n.b. Motif, Windows and the Macintosh). Mosaic interprets a majority of the HTML DTD elements and presents the encoded information with page formatting, type-face specification, image display, fill-in forms, and graphical widgets. In addition, Mosaic provides inherent access to FTP, Gopher, WAIS and other network services [Andreessen].

1.3 VIS

VIS is a simple but complete volume visualizer. VIS provides arbitrary three-dimensional transformation (e.g. rotation and scaling), specification of six axial clipping planes (n.b. a cuboid), one arbitrary clipping plane, and control of opacity and intensity. VIS interactively transforms the cuboid, and texture-maps the volume data onto the tranformed geometry. It supports distributed volume rendering [Argrio, Drebin, Kaufman] with run-time selection of computation servers, and isosurface generation (marching cubes)[Lorenson, Levoy] with software Gouraud shading for surface-based model extraction and rendering. It reads NCSA Hierarchical Data Format (HDF) volume data files, and has a graphical interface utility to import volume data stored in other formats.

The rest of the paper is divided into five parts. Section 2 will describe VIS and how it fulfills the role as a W3 visualization tool. Section 3 is devoted to Mosaic, and will include the implementation of its interface with VIS. Section 4 presents the project results. Section 5 is about ongoing and future development, and the last section is conclusions.

2. VIS: A Distributed Volume Visualization Tool

VIS is a highly modular distributed visualization tool, following the principles of client/server architecture (figure 1), and consisting of three cooperating processes: VIS, Panel, and VRServer(s). The VIS module handles the tasks of transformation, texture-mapping, isosurface extraction, Gouraud shading, and manages load distribution in volume rendering. VIS produces images that are drawn either to its own top-level window (when running stand-alone) or to a shared window system buffer (when running as a cooperative process). The Panel module provides a graphical user-interface for all VIS functionality and communicates state changes to VIS. The VRServer processes execute on a heterogenous pool of general purpose workstations and perform volume rendering at the request of the VIS process . The three modules are integrated as shown in figure 3 when cooperating with another process. A simple output window is displayed when no cooperating process is specified.

2.1 Distributed Volume Rendering

Volume rendering algorithms require a significant amount of computational resources. However, these algorithms are excellent candidates for parallelization. VIS distributes the volume rendering among workstations with a "greedy" algorithm that allocates larger portions of the work to faster machines [Bloomer]. VIS segments the task of volume rendering based on scan-lines, with segments sized to balance computational effort versus network transmission time. Each of the user-selected computation servers fetches a segment for rendering via remote procedure calls (RPC), returns results and fetch another segment. The servers effectively compete for segments, with faster servers processing more segments per unit time, ensuring relatively equal load balancing across the pool. Analysis of this distribution algorithm [Giertsen, 93] shows that the performance improvement is a function of both the number of segments and the number of computational servers, with the optimal number of sections increasing directly with the number of available servers. Test results indictate that performance improvement flattens out between 10 to 20 segments distributed across an available pool of four servers. Although this algorithm may not be perfect, it achieves acceptable results.

2.2 Cooperative Visualizaton

The VIS client, together with its volume rendering servers, may be lauched by another application as a visualization server. The two requirements of cooperation are a shared window system buffer for the rendered image and support for a limited number of inter-process messages. VIS and the initiating application communicate via the ToolTalk service, passing message specifying the data object to visualize, options for visualization, and maintaining state regarding image display. The VIS Panel application appears as a new top-level window and allows the user control of the visualization tool.

3. Visualization with Mosaic

We have enhanced the Mosaic W3 browser to support both a three-dimensional data object and communication with VIS as a cooperating application (figure 2). Mosaic provides the user with the ability to locate and browse information available from a wide variety of sources including FTP, WAIS, and Gopher. HTTP servers respond to requests from clients, e.g. Mosaic, and transfer hypertext documents. Those documents may contain text and images as intrinsic elements and may also contain external links to any arbitrary data object (e.g. audio, video, etc...). Mosaic may also communicate with other Internet servers, e.g FTP, either directly – translating request results into HTML on demand – or via a gateway that provides translation services. As a W3 client, Mosaic communicates with the server(s) of interest in response to user actions (e.g. selecting a hyperlink), initiating a connection and requesting the document specified by the URL. The server delivers the file specified in the URL, which may be a HTML document or a variety of multimedia data files (for example, images, audio files, and MPEG movies) and Mosaic uses the predefined SGML DTD for HTML to parse and present the information. Data types not directly supported b Mosaic are displayed via user-specifiable external applications and we have extended that paradigm to both include three-dimensional volume data as well as to integrate the external application more completely with Mosaic.

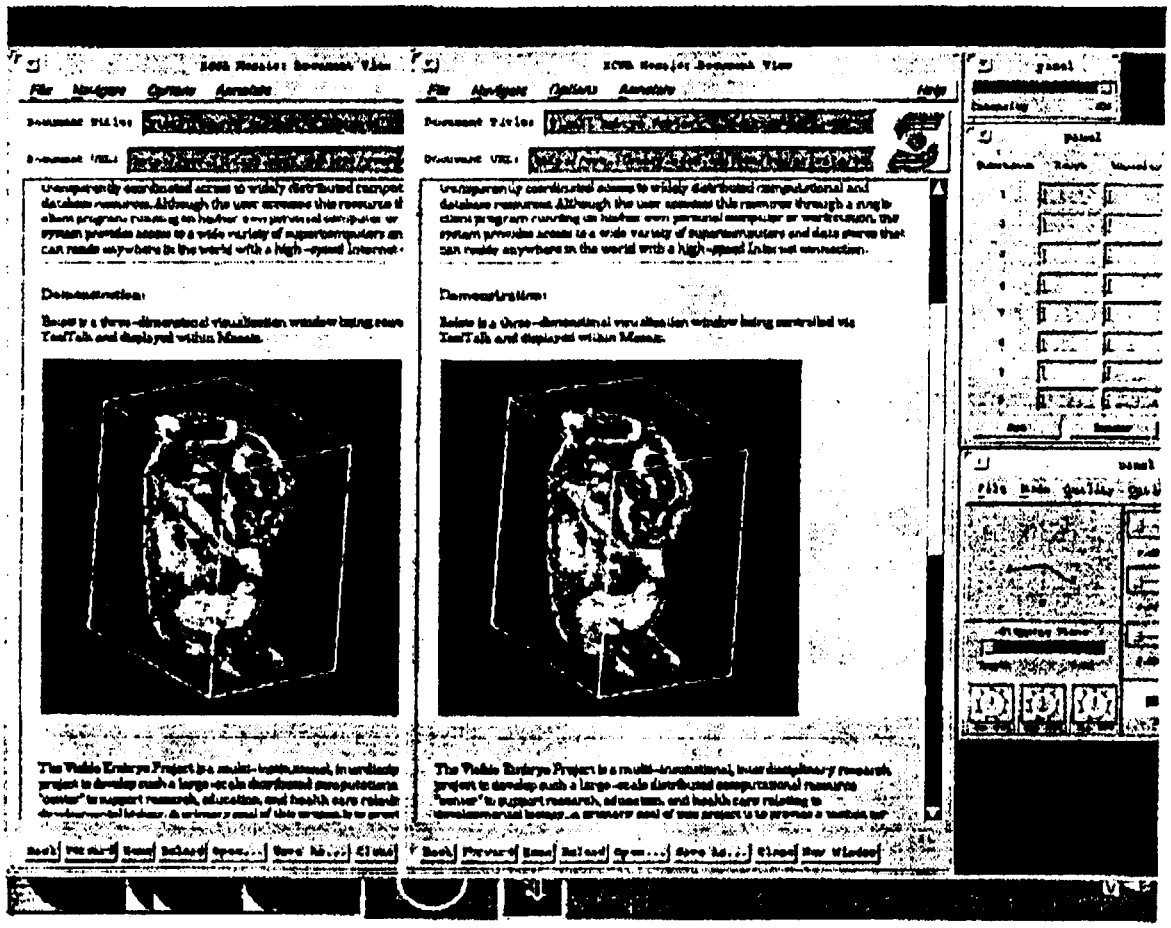

Figure 2: Mosaic utilizing VIS to embed an interactive visualization within an HTML document. Here the data has been rotated 6 degrees in a "cloned" Mosaic window to create a stereo pair.

3.1 Mosaic 3D Image support

We have extended the HTML DTD to support three dimensional data via the introduction of a new SGML element: IMG3D. This element provides information to the presentation system (i.e. Mosaic) about the content that is referenced in the document. The IMG3D element is defined in the HTML DTD as follows:

```
<!ELEMENT    IMG3D EMPTY>
<!ATTLIST    IMG3D VOLUME CDATA #REQUIRED
             WIDTH NUMBER #REQUIRED
             HEIGHT NUMBER #REQUIRED
             IMAGE CDATA #IMPLIED>
```

which is translated as "SGML document instance element tag IMG3D containing no content; three required attributes: volume data set, window width and height; and an optional image". In a HTML document, a 3D image element would be represented as:

```
<IMG3D       VOLUME="http://www.library.ucsf.edu/.../data/Embryo.hdf"
             WIDTH=400
             HEIGHT=400
             IMAGE="http://www.library.ucsf.edu/.../images/Embryo.gif">
```

which may be interpreted as "create a 3D visualization window of width 400 pixels, height 400 pixels, and visualize the data embryo.hdf located at the HTTP server site www.library.ucsf.edu".

3.2 Interface with Mosaic

The VIS/Mosaic software system consists of three elements: VIS, Mosaic, and Panel (which provides GUI for data manipulation). Currently, VIS communicates with Mosaic via ToolTalk™, but the system will work with any inter-client communication protocol. Whe Mosaic interprets the HTML tag IMG3D, it creates a drawing area widget in the document page presentation and creates a shared window system buffer to receive visualization results. In addition, Mosaic launches the VIS application, specifying the location of the data object to render and identifying the shared image buffer. When the VIS process begin execution, it verifies its operating parameters and launches the Panel application that in turn presents the user with the control elements for manipulating the visualization and manages the communication between VIS and Mosaic.

In addition to coordinating communications, the Panel application attempts to balance the rendering load across the selected VRServer(s), sending rendering parameters to the selected server(s), and integrating the resulting image segments(s). Thus the scenario following a user's action on the Panel will be (1) Panel sends rendering requests to VRServer(s) with the parameters from its GUI, (2) Panel fetches the returned image data, then writes it to the pixmap, (3) Panel sends a message to notify Mosaic upon completion, and (4) Mosaic bit-blots the image in the pixmap into its DrawingArea widget. This will be addressed in more details under section 3.2. The configuration of this software system is depicted in figure 3.

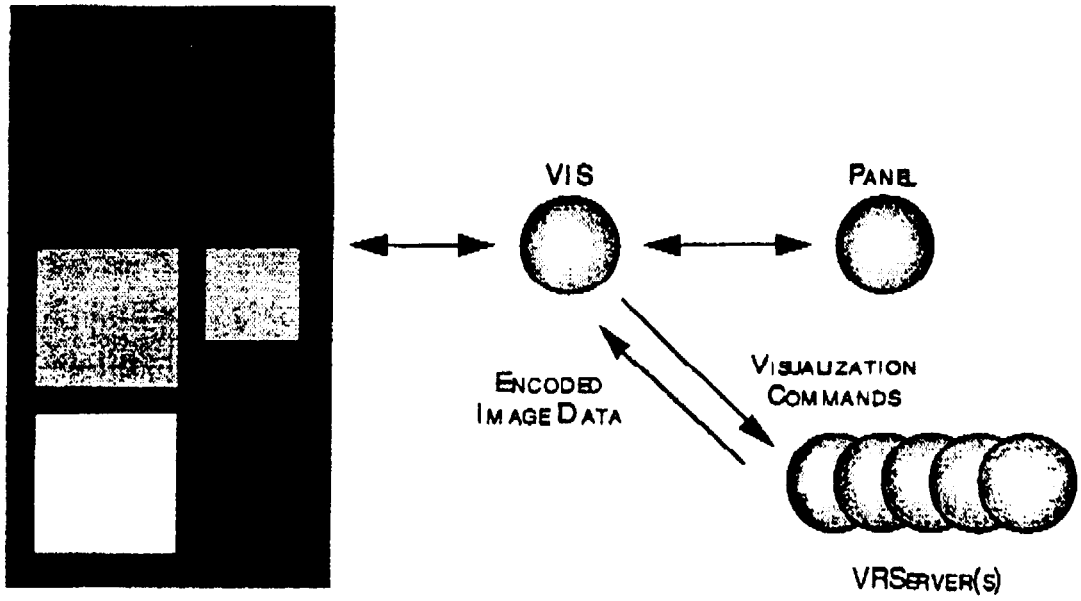

Figure 3: Interprocess communication among Mosaic, VIS and distributed rendering servers.

3.3 Interclient communication

We recognized the minimum set of communication protocols between Mosaic and Panel:

(a) Mosaic notifies Panel when the window id is no longer valid. This happens when the user navigates from the current page to the previous (with BACK button), or to another document through a link.

(b) Panel notifies Mosaic when the scene needs to be updated. This occurs when the user manipulates the data through the Panel's GUI, which may be any actions (rotations, scaling, changes in clipping planes, mode switching, for instance, from texture mapping to volume rendering) that result in rerendering of the picture.

The first case of (a) does not impose any problem since Mosaic can simply terminate VIS (this solution also applies when Mosaic is quitting). The second case is harder to handle because Mosaic caches the page, and in principle, we expect VIS to remain running because it should not be loading the HDF data, which may be quite a few megabytes, again. In the previous case, we simply have Mosaic kill the child process that launched Panel, which may subsequently send a terminate message to the VIS servers. In the latter case, Mosaic will send a message to Panel requesting it to unrealize/unmap its window. Then when the user comes back to the 3D object page, Mosaic may send another message to Panel ordering it to realize the GUI window, and update the window id of the DrawingArea that Panel has been sending message to (ironically Mosaic does not cache the widgets/windows on the cached page).

In (b), Panel will send an update message directly to the DrawingArea window id provided by Mosaic. Upon receiving the redrawing message, the DrawingArea widget calls its registered refreshing function, which simply bit-blots the pixmap Panel updated into the window. The protocols are summarized in Table 1.

MESSAGES	MOSAIC NOTIFIES VIS	VIS RESPONDS TO MOSAIC

Refresh	Compute Image	Done Drawing
PageCaching	Hide Panel	Panel Hidden
BacktoCachedPage	New Window ID	ID Changed, Panel Shown
DestroyPage	Terminate	Terminating

Table 1: Mosaic/VIS IPC communication.

4. Results

The results of the above implementation are very encouraging. The Mosaic/VIS sucessfully allows users to visualize HDF volume datasets from various HTTP server sites. Fig 2 shows a snapshot of the W3 visualizer. Distributing the volume rendering loads results in a remarkable speedup in image computations. However, we did not do any detail performance analysis, because the results of such an analysis would not be reproducable for varying network traffics loads, and fluctuating server workstation loads. The server pool we tested the system consists of heterogenous workstations: a SGI Indigo2 R4400/150MHz, two SGI Indy R4000PC/100MHz, a DEC Alpha 3000/500 with a 133MHz Alpha processor, two Sun SparcStations 10, and two Sun SparcStations 2, which were located arbitrarily on a Ethernet network. To our knowledge this is the first demonstration of the embedding of interactive control of a client/server visualization application within a multimedia document in a distributed hypermedia environment, such as the World Wide Web.

5. Ongoing/Future work

We have begun working on several extensions and improvements on the above software system:

5.1 MPEG Data Compression

The data transferred between the visualization servers and the clients constitue the exact byte streams computed by the servers, packaged in the XDR machine independent format. One way to reduce network transferring time would be to compress the data before delivery. We propose to use the MPEG compression technique, which will not only perform redundancy reduction, but also a quality-adjustable entropy reduction. Furthermore, the MPEG algorithm performs interframe, beside intraframe, compression. Consequently, only the compressed difference between the current and the last frames is shipped to the client.

5.2 Generalized External-Application-to-Mosaic-Document-Page Display Interface

The protocols specified in Table 1 are simple, and general enough to allow most image-producing programs be modified to display in the Mosaic document page. Our undertakings will be to extend the protein database (PDB) displaying program, and the xv 2D image processing program, to be Mosaic PDB visualization server, and Mosaic 2D image processing server.

5.3 Multiple Users

With multiple users, the VIS/Mosaic distributed visualization system will need to manage the server resources, since multiple users utilizing the same computational servers will slow the servers down significantly. The proposed solution is depicted in Fig 4. The server resource manager will allocate servers per VIS client request only if those servers are not overloaded. Otherwise, negotiation between the resource manager and the VIS client will be necessary, and, perhaps the resource manager will allocate less busy alternatives to the client.

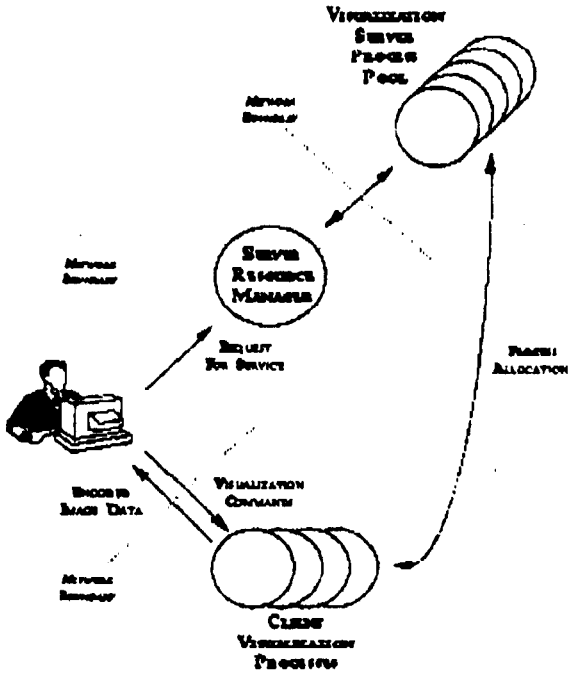

Fig 4: Server Resource Management

5.4 Load Distributing Algorithm

Since the load distributing algorithm in the current VIS implementation is not the most optimal load distribution solution, we expect to see some improvement in the future implementation, which will be using sender-initiated algorithms, described in [Shivaratri].

6. Conclusions

Our accomplishment takes the technology of networked multimedia system (especially the World Wide Web) a step further by proving the possibility of adding new interactive data types to the W^3 servers and clients, and coordinating the execution of the applications that handle them with the W^3 clients. The addition of the 3D volume data object in the form of HDF to the W^3 is welcomed by many medical researchers, for it is now possible for them to view volume datasets without a high-cost workstation, and accessing datasets in the W^3 fashion, through hypertext and hypergraphics links within an HTML page. As for the researchers who would like to share their findings with the world, they merely have to run a W^3 server that serves the HDF files, which in turn, can be easily created with the various import facilities available freely from NCSA.

7. References

Andreessen, Marc, "NCSA Mosaic Technical Summary", FTPed from ftp ncsa uiuc.edu, Revision 2.1, May 8, 1993.

Argiro, V. "Seeing in Volume", Pixel, July/August 1990, 35-39.

Avila, R., Sobierajski, L. and Kaufman A., "Towards a Comprehensive Volume Visualization System", Visualization '92 Proceedings, IEEE Computer Society Press, October 1992, 13-20.

Bloomer, John, "An RPC Case Study: Networked Ray Tracing", Power Programmingwith RPC, O'Reilly & Associates, Inc., Sept 92, 409-451.

Brinkley, J.F., Eno, K., Sundsten, J.W., "Knowledge-based client-server approach to stuctural information retrieval: the Digital Anatomist Browser", Computer methods and Programs in Biomedicine, Vol. 40, No. 2, June 1993, 131-145.

Broering, N. C., "Georgetown University, The Virtual Medical Library," Computers in Libraries, Vol. 13, No. 2, February 1993, 13.

Drebin, R. A., Carpenter, L. and Hanrahan, P., "Volume Rendering", Computer Graphics, Vol. 22, No. 4, August 1988, 64-75.

Flanders, B., "Hypertext Multimedia Software: Bell Atlantic DocuSource", Computers in Libraries, Vol 13, No. 1, January 1993, 35-39.

Gelerg, L., "Volume Rendering in AVS5", AVS Network news, Vol. 1, Issue 4, 11-14.

Giertsen, C. and Petersen, J., "Parallel Volume Rendering on a Network of Workstations", IEEE Computer Graphics and Applications, November 1993, 16-23.

Jäger, M., Osterfeld, U., Ackermann, H, and Hornung, C., "Building a Multimedia ISDN PC", IEEE Computer Graphics and Applications, September 1993, 24-33.

Kaufman, A., Cohen, D., and Yagel, R., "Volume Graphics", Computer, July 1993, 51-64.

Kiong B., and Tan, T., "A hypertext-like approach to navigating through the GCG sequence analysis package", Computer Applications in the Biosciences, Vol. 9, No. 2, 1993, 211-214.

Levoy, M., "Display of Surfaces from Volume Data", IEEE Computer Graphics and Applications, Vol. 8, No. 5, May 1988, 29-37.

Lorensen, W., Cline, H.E., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm", Computer Graphics, Vol. 21, No. 4, July 1987, 163-169.

Mercurio, F., "Khoros", Pixel, March/April 1992, 28-33.

Narayan, S., Sensharrma D., Santori, E.M., Lee, A.A., Sabherwal, A., Toga, A.W., "Animated visualization of a high resolution color three dimensional digital computer model of the whole human head", International Journal of Bio-Medical Computing, Vol 32, No. 1, January 1993, 7-17.

Nickerson, G., "WorldWideWeb Hypertext from CERN", Computers in Libraries, Vol. 12, No. 11, December 1992, 75-77.

Obraczka, K., Danzig, P, and Li, S., "Internet Resource Discovery Services", Computer, Vol. 26, No. 9, September 1993, 8-22.

Pommert, A., Riemer, M., Schiemann, T., Schubert, R., Tiede, U., Hoehne, K-H, "Methods and Applications of Medical 3D-Imaging", SIGGRAPH 93 course notes for volume visulization, 68-97.

Robison, D., "The Changing States of Current Cites: The Evolution of an Electronic Journal", Computers in Libraries, Vol. 13, No. 6, June 1993, 21-26.

Shivaratri, N.G., Krueger, P., and Singhal, M., "Load Distributing for Locally Distributed Systems", Computer, December 1992, 33-44.

Singh, J, Hennessy, J. and Gupta A., "Scaling Parallel Programs for Multiprocessors: Methodology and Examples", Computer, July 1993, 42 49.

Story, G., O'Gorman, L., Fox, D, Schaper, L. and Jagadish, H.V., "The RightPages Image-Based Electronic Library for Alerting and Browsing", Computer, September 1992, 17-26.

VandeWettering, M., "apE 2.0", Pixel, November/December 1990, 30-35.

Woodward, P., "Interactive Scientific Visualization of Fluid Flow", Computer, Vol. 26, No. 10, June 1993, 13-25.

Zandt, W.V., "A New 'Inlook' On Life", UNIX Review, Vol 7, No. 3, March 1989, 52-57.