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
San Francisco, California 94143-0840

The World-Wide-Web (WWW) has created a new paradigm for online information retrieval by 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 capabilities 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 these new data types and a description of an implementation of this approach in a WWW-compliant 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, VandeWettering]. Previously these systems have depended upon specialized graphics hardware for rendering and significant local 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 to 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, we have further 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 heterogeneous 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 WWW, taking advantage of the client-server architecture of WWW and its ability to access hypertext documents stored anywhere on the Internet [Obraszka, Nickersen]. We have enhanced the capabilities of the most popular WWW client, Mosaic [Andreessen] 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. It should be noted that other types of interactive applications could be "embedded" within HTML documents as well. Our approach can be generalized to allow the implementation of object linking and embedding over the Internet, similar to the features the OLE 2.0 provides users of Microsoft Windows on an individual machine.

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

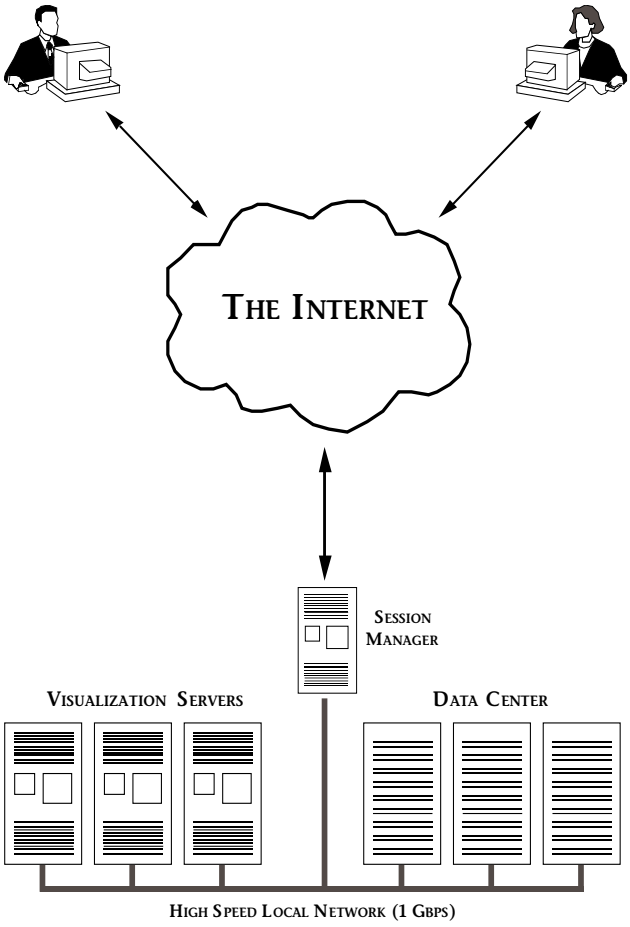

Figure 1: VIS client/server model.

(DTD) of the Standard Generalized Mark-up Language (SGML). These documents are returned to WWW 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 (Motif, Windows and 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 transformed 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.

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 collectively 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 messages specifying the data object to visualize as well as 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 WWW browser to support both a three-dimensional data object and communication with VIS as a coopezrating application (figure 2). HTTP servers respond to requests from clients, e.g. Mosaic, by transferring hypertext documents to the client. 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 WWW 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 by 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 applications more completely with Mosaic.

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: EMBED. This element provides information to the presentation system (i.e. Mosaic) about the content that is referenced in the document. The EMBED element is defined in the HTML DTD as shown in Example 1, which is translated as “SGML document instance element tag EMBED containing no content; four required attributes: TYPE, the type of the external application, in the MIME-type format; HREF, the location/URL of the datafile; WIDTH, the window width and, HEIGHT, the window height. The TYPE attribute give this specification the flexibility to accomodate different types of external applications. In a HTML document, a 3D image element would be represented as shown in Example 2, which may be interpreted as “create a drawing-area window of width 400 pixels, height 400 pixels, and use the application associated to hdf/volume MIME content-type to 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. Currently, the VIS application communicates with Mosaic via ToolTalk™, but the system will work with any interclient communication protocol. When Mosaic interprets the HTML tag EMBED, it creates a drawing area widget in the document page presentation and requests a shared buffer or pixmap from the windowing system to receive visualization results. In addition, Mosaic launches the Panel process, specifying the location of the data object to render and identifying the shared image buffer. The Panel process begins execution by first verifying its operating parameters, then launching the VIS process. The Panel process also presents the user with the control elements for data manipulation and manages the communication between the whole VIS application and Mosaic.

The VIS process, on the other hand, serves as a rendering engine. It executes the visualization commands from the Panel process, integrates the image data segments from various VRServers, and presents the complete array of image data to the Panel.

Thus the scenario following a user’s action on the Panel will be (1) Panel issues visualization commands to the VIS rendering engine, (2) VIS sends rendering requests to

Figure 2: VIS embeded within Mosaic for interactive visualization in a HTML document.

VRServer(s), then gathers the resulting image segments, (3) Panel fetches the returned image data, then writes it to the pixmap, (4) Panel notifies Mosaic upon completion, and (5) Mosaic bit-blots the pixmap contents into its corresponding DrawingArea widget. The interprocess communication issue will be addressed in more details under section 3.3. The configuration of this software system is depicted in figure 3.

3.3 Interclient communication

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

(a) Messages from Mosaic to a Panel process include the following:

(i) ExitNotify - requesting the Panel to terminate itself.when Mosaic exits.

(ii) MapNotify - requesting the Panel to map itself to the screen when the HTML document containing the DrawingArea corresponding to the above panel is visible.

(iii) UnmapNotify - requesting the Panel to unmap/iconify itself when the HTML page containing the DrawingArea corresponding to the above Panel is cached.

(b) Messages from a Panel process to Mosaic may be one of the following:

(i) RefreshNotify - informing Mosaic of an update in the shared pixmap, and requesting Mosaic to update the correspinding DrawingArea.

(ii) PanelStartNotify - informing Mosaic the Panel is started successfully, and ready to receive messages.

(iii) PanelExitNotify - informing Mosaic the Panel is exiting, and Mosaic should not send any more messages to the Panel.

We have packaged the above protocols and all the required messaging functions into a library. Modification of an existing external application merely involves registration of the external application's messaging window (the window to receive Mosaic's messages), installation of callback functions corresponding to

the messages from Mosaic, and addition of message-sending routine invocations. The protocol is summarized in Table 1.

Messages	Descriptions
ExitNotify	Mosaic exiting
MapNotify	DrawingArea visible
UnmapNotify	DrawingArea cache
RefreshNotify	DrawingArea update
PanelStartNotify	Panel starting
PanelExitNotify	Panel exiting

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 WWW visualizer. Distributing the volume rendering loads results in a remarkable speedup in image computations. Our performance ananlysis with a homogeneous pool of Sun SPARCstation 2's on a relatively calm network produced reasonable results (Figures 4a, 4b, and 4c. Three trials per plot). The time-versus-number-of-workstations curve decreases as more servers participate, and plateaus when the number of SPARCstations is 11 in the case of 256x256 image (9 for 192x192 image, and 7 for 128x128 image). The speed increases at the plateaus are very significant: about 10 times for the 256x256 image, 8 times for the 192x192 image, and 5 times for the 128x128 image. The outcomes suggest that performance improvement is a function of the number of volume rendering servers. Furthermore, the optimal number of workstations and the speed increase are larger when the image size is bigger. This is in complete agreement with Giertsen's analysis. We have also successfully tested the software system in an environment consisting 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 an 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:

```
<!ELEMENT    EMBED   EMPTY>
<!ATTLIST    EMBED
             HREF    %URL    #REQUIRED
             TYPE    CDATA   #REQUIRED
             WIDTH   NUMBER  #REQUIRED
             HEIGHT  NUMBER  #REQUIRED>
```

Example 1: SGML definition for EMBED element.

```
<EMBED
   HREF="http://<host>/…/Embryo.hdf">
   TYPE="hdf/volume"
   WIDTH=400
   HEIGHT=400>
```

Example 2: EMBED element usage.

5.1 MPEG Data Compression

The data transferred between the visualization servers and the clients consists of 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

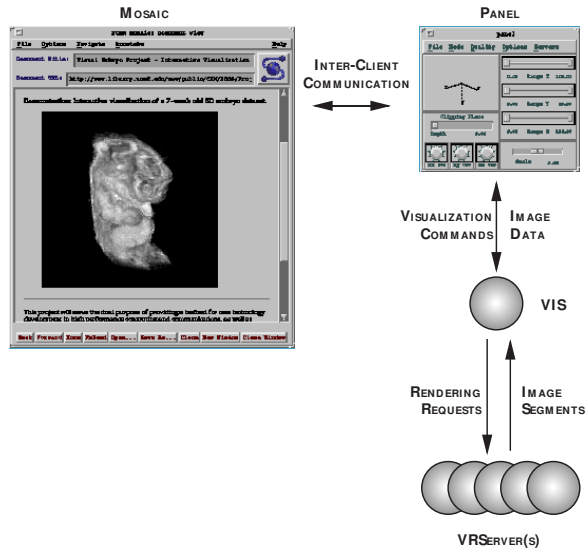

Figure 3: Communication among Mosaic, VIS and distributed rendering servers.

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. We have successfully incorporated an in-house CAD model rendering program into Mosaic. Our next undertakings will be to extend the protein database (PDB) displaying program, and the xv 2D image processing program, to create a Mosaic PDB visualization server, and a 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 5. The server resource manager will allocate servers per VIS client request only if those servers are not

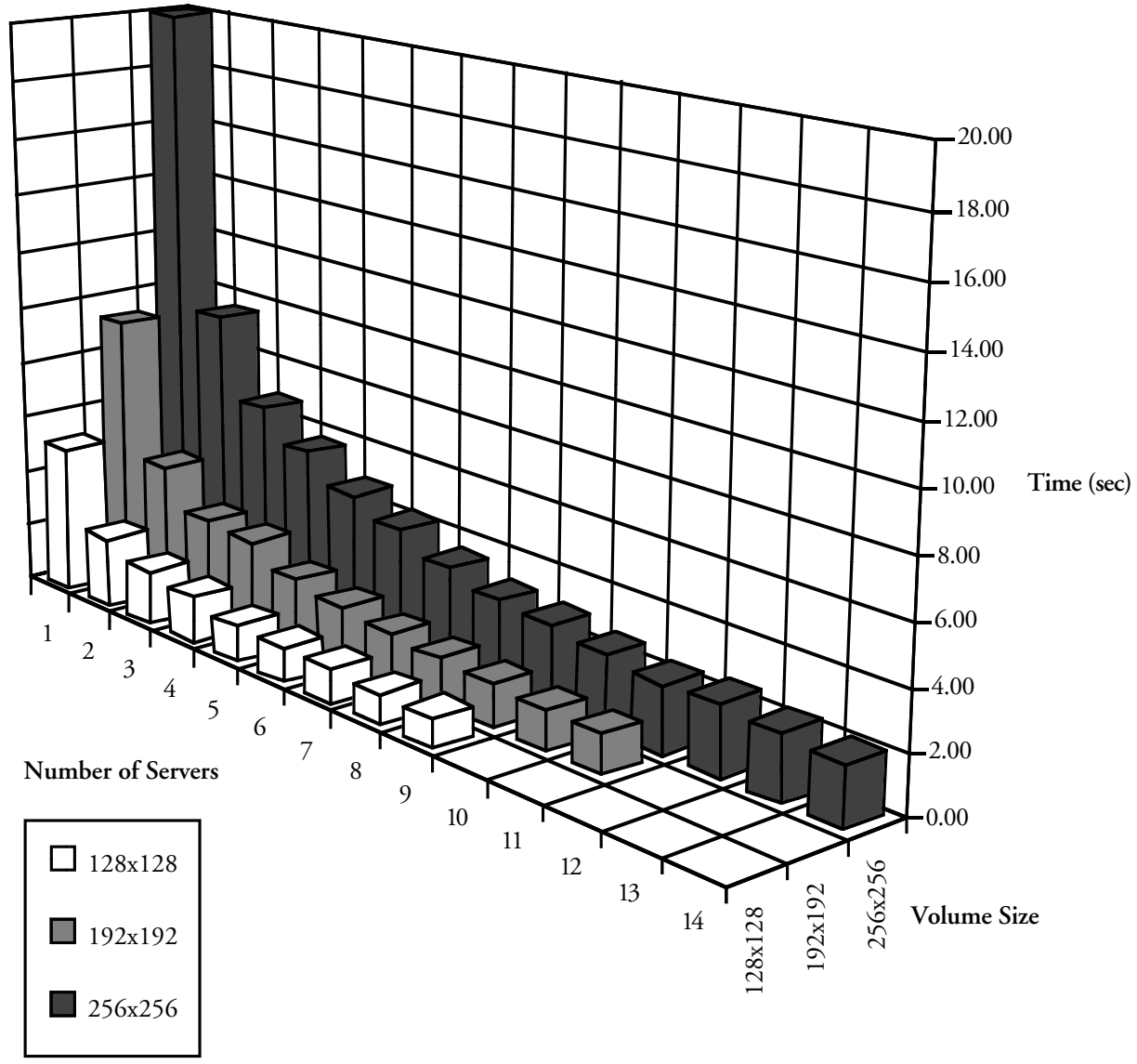

Figure 4: Volume rendering performance for 128^2, 192^2, and 256^2 data sets .

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.

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].

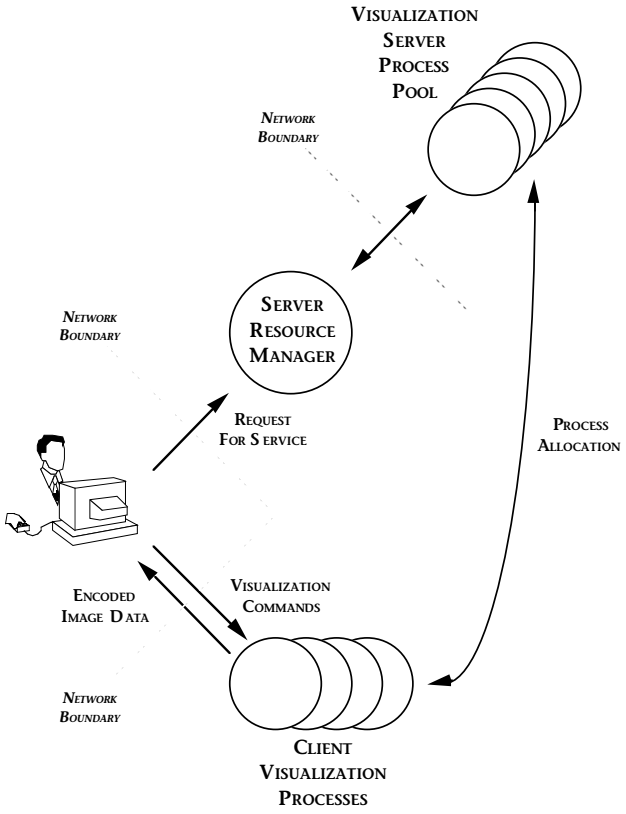

Figure 5: Server Resource Management

6. Conclusions

Our system 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 both the WWW servers and clients. The addition of the 3D volume data object in the form of an HDF file to the WWW has been welcomed by many medical researchers, for it is now possible for them to view volume datasets without a high-cost workstation. Furthermore, these visualizations can be accessed via the WWW, through hypertext and hypergraphics links within an HTML page. Future implementations of this approach using other types of embedded applications will allow the creation of a new paradigm for the online distribution of multimedia information via the Internet.

7. References

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.

Andreessen, M., "NCSA Mosaic Technical Summary", from FTP site ftp.ncsa.uiuc.edu, 8 May 1993.

Bloomer, J., "Power Programming with RPC", O'Reilly & Associate, September 1992, 401-451.

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.