

Remote 3D Visualization using Image-Streaming Techniques

Klaus Engel

Ove Sommer

Christian Ernst

Thomas Ertl

Computer Graphics Group
Universität Erlangen-Nürnberg, Germany *

Abstract

The visualization of large scale data sets is still one of the biggest challenges in scientific visualization. In the field of web-based visualization the handling of such data sets raises new questions: How to distribute the visualization task onto client and server machines? How to achieve interactive refresh rates? How to reduce network load? How to adapt the visualization to the network bandwidth? How to extend single user visualization applications into collaborative visualization tools?

In this paper we present a new approach for web-based visualization using image streaming techniques. A framework enables the remote control of an OpenInventor application. Images generated by a high-end visualization server are streamed through the network to visualization clients using a special video-streaming codec. The visualization on the server is controlled by CORBA (Common Object Request Broker Architecture) requests generated on the client machines. CORBA provides access to the visualization server from a large variety of client architectures - an issue which is especially important in the field of the heterogeneous World Wide Web. We present a web-based teleradiology system which is based on the proposed techniques.

1 Introduction and Related Work

The World Wide Web provides access to a huge amount of scientific data. New algorithms and applications have to be developed in order to visualize such data on the large variety of clients on the World Wide Web. Data sets from typical scientific applications are growing fast. For example data volumes from 3D medical imaging like CT are approaching sizes of 512^3 which amounts to more than 100 million of voxel cells. Special features of high-end graphics machines are trying to handle such huge data sets by relocating visualization tasks directly into hardware. For example expensive interpolation calculations are taken over by 3D texture mapping hardware.

In the past years several approaches for scientific visualization on the web were investigated. One of the first progressive applications for volume visualization was presented by Lippert et. al [9]. Their system is based on the local reconstruction of wavelet-compressed data, but since it works in the Fourier domain it is restricted to X-ray like images.

Driven mainly by the game industry, the cost of 3D graphics hardware with texture mapping support is continuously decreasing over the past few years. This development has induced that the role of the client for distributed scientific visualization has been emphasized in recent work.

Hendin introduced a VRML-based volume visualization tool [7], which uses three stacks of perpendicular slices. The VRML-plugin is controlled by a Java applet using the External Authoring Interface

(EAI) [10]. We extended the approach of Hendin in [3]. We presented a web-based, multi-user visualization tool for medical volume datasets which makes heavy use of new rendering capabilities of client machines. Since it is completely written in Java, it can be executed on a standard web-browser. The Virtual Reality Modeling Language (VRML) [5, 2] is used to visualize medical volume datasets using texture mapping, which enables the use of fast 3D graphics acceleration hardware of client systems - especially bilinear filtering using texture mapping hardware. We introduced techniques for fast volume clipping, collaborative work and data size reduction. However the application requires volume data to be transferred to all clients participating in a visualization session. Besides the visualization quality is limited due to the bilinear interpolation of the volume data.

Ma and Patten introduced a web based volume visualization system called DiVision, which allows the user to explore remote volumetric datasets using a web browser [11]. The system computes images on a visualization server, which are transferred to the client and inserted into a graph. The graph represents the relationship of all images which the user has rendered so far. The edges of the graph show the change of rendering parameters between two images.

Despite of the fact that the gap in computation power between high-end workstations and low-end clients became smaller over the past years, high-end servers will probably always have special features that will enable them to visualize large scale data sets faster and with better image quality than standard PCs.

In order to make those capabilities available to a large variety of clients on the World Wide Web we propose the streaming of images, generated by a high-end visualization server, through a network connection to clients only used as display and control devices. Visualization parameters (e.g. viewpoint, mapping parameters) can be controlled on the clients side and are sent to the visualization server using CORBA method invocations. The use of CORBA warrants access to the visualization server from a large variety of client applications, thus enabling to be platform-independent on the client side. We developed a framework, that allows to adapt an OpenInventor application for remote control. developed direct volume visualization application which make heavy use of special 3D texture mapping hardware of a high-end graphics machine was adapted using our framework [12].

Our framework allows the sharing of expensive hardware in local area and wide area networks. Additionally the framework provides new possibilities for collaborative work and distance education. Multiple users at different locations can join a visualization session and work on a large scale dataset which is rendered on a high-end graphics server. On the client side only a standard PC is needed to join the session.

In the following section we will describe the general architecture of our framework. The image-streaming codec is outlined in Section 3. Section 4 explains the exemplary teleradiology system we built using the proposed techniques. Results are given in section 5. We will conclude the paper with some remarks on future activities.

*Lehrstuhl für Graphische Datenverarbeitung (IMMD9), Universität Erlangen-Nürnberg, Am Weichselgarten 9, 91058 Erlangen, Germany, Email: engel@informatik.uni-erlangen.de, URL: <http://www9.informatik.uni-erlangen.de>

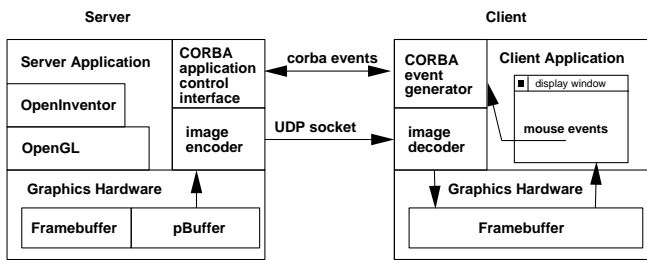


Figure 1: Framework Architecture

2 General Architecture

In order to achieve interactive refresh rates the use of 3D hardware acceleration is essential. OpenGL is the graphics application programming interface (API) for developing portable, interactive 2D and 3D graphics applications. However, it is a low-level API which does not provide extended features for the navigation in 3D scenes and the manipulation of 3D objects. OpenInventor is an object oriented graphics toolkit built on top of OpenGL, which has become a de-facto standard for interactive modeling, rendering and manipulation of 3D scenes [14]. It provides a scene graph programming interface with a wide variety of 3D manipulation capabilities. One of the most important aspects of OpenInventor is the ability to program new objects as extensions to the toolkit. OpenGL extensions which provide access to special hardware features can be used from within these new objects.

We use OpenInventor to render images into the pBuffer, a special protected graphics memory block which allows hardware accelerated off-screen rendering. Images are read from this buffer into main memory, encoded, transferred to the client, decoded and displayed (Fig. 1).

A framework which enables any OpenInventor application to be remotely controlled was developed. The streaming of images and the remote control of the server application are strictly decoupled. They are realized using two different network connections, thus enabling the replacement of the image-transfer module by other more sophisticated streaming modules. Our framework allows the design of collaborative visualization applications as outlined in Fig. 2.

The client application is able to remotely control the server application by invoking different remote methods of a server object. CORBA lets you invoke methods on the server using the high-level language of choice, so the client application can be implemented as a stand-alone x-application or as a Java applet. The client has to provide a drawing area with mouse event handling capabilities for the display of the streamed images.

On the server side OpenInventor provides 3-dimensional manipulator widgets which are integrated into the visualization of the data. Transformations, clipping plane positioning etc. are remotely influenced by picking and dragging manipulator handles. The position and type of mouse-events generated on a client are sent to the visualization server using CORBA requests. Then they are translated into OpenInventor events and handled by the OpenInventor application. Other application specific visualization parameters are manipulated using the graphical user interface of the client application and are sent to the server using a special request method.

3 Image-Streaming

While our setup would allow the transmission of generated images by the off-screen renderer in an uncompressed raw format, we further extend the transmission of images by an image-

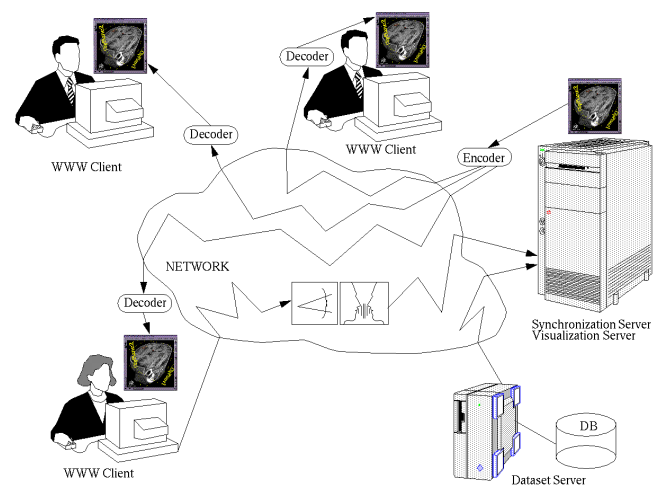


Figure 2: Collaborative work using our framework. Visualization parameters changed by one user are sent to the server and applied to the visualization. Newly created images are encoded, sent to the clients, decoded and displayed.

streaming codec. This codec has been integrated in cooperation with the Telecommunications Institute of the University of Erlangen-Nuremberg.

We shortly present their transmission scheme for Internet video streaming that provides an acceptable video quality over a wide range of connection qualities.

Images generated by the server visualization application are encoded on-the-fly by an encoder module and sent through the network using a UDP socket connection. On the client side the received data is decoded and extracted images are copied into the frame buffer. In order to make efficient use of the available resources, a scheme like this must assume stable and in a multicast scenario identical channel conditions for all receivers, as well as identical resources to perform the channel and video decoding of the received packets. Unfortunately, all these assumptions do not hold for the Internet: Channel conditions are both time- and user-dependent, and users have different network access and computational resources. Therefore scalable video schemes were proposed which produce bitstreams decodable at different bit rates, requiring different computational power.

The Internet video transmission scheme used in this paper is based on spatio-temporal resolution pyramids, first proposed by Uz and Vetterli in 1991 [15, 13]. The idea is shown in Fig. 3 for a two-layer spatio-temporal pyramid. The original video signal is decomposed into a low quality and a high quality layer. The QCIF base layer has half of the temporal resolution of the CIF enhancement layer. The base layer is transmitted by a fully standard compatible H.263 coder. The enhancement layer is encoded by a motion compensated hybrid coder where the usual DCT (Discrete Cosinus Transform) has been replaced by lattice vector quantization to give an improved coding efficiency. Within the low quality layer the original signal is represented at a quarter of its original spatial resolution, and at half of its temporal resolution. The low quality layer can be encoded at a lower bit-rate since it contains fewer samples than the original sequence. Nevertheless the overall amount of samples to encode both layers is increased by 12.5% in this example. The advantages of this overcomplete representation are, that downsampling and interpolation filters can be chosen freely and that motion-compensation can be easily included without suffering from the drift problem in cases where only the lower

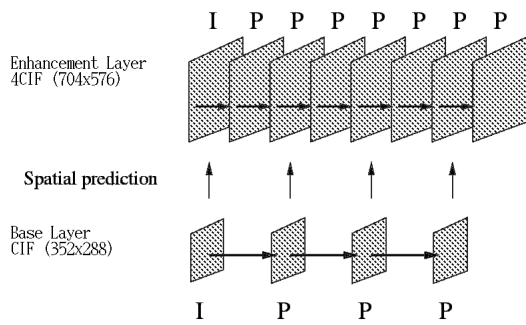


Figure 3: The Group Of Pictures (GOP) structure of the scalable coder. It contains a base layer at CIF resolution (352x288 pixels) complemented by a 4CIF (704x576 pixels) enhancement layer. This GOP comprises eight frames, i.e., in each layer an Intra-frame is transmitted every 8th or 4th frame. Spatial and temporal redundancies are exploited by applying spatial and/or temporal prediction as denoted by vertical and horizontal arrows.

resolution layer can be decoded.

The proposed scheme is especially useful for Internet multicast applications where different users are subject to different channel qualities. It allows the encoding of images stream with two layers at CIF and QCIF resolution at 5 frames per second on a fast workstation in software. This frame rate is sufficient for applications where the render server is even locally not able to produce high frame rates. Higher CPU power will allow to achieve better frame rates in the future. A computation power of a standard PC is needed to decode the image stream. A more detailed description of this approach can be found in [4, 8].

4 A Web-based Teleradiology System

In this section we will describe the web-based teleradiology system (Fig. 4) that we built using the proposed framework. Teleradiology systems aim to reduce distances for the medical community by providing mechanisms for the distribution of medical data over network connections and enabling collaborative work. The World Wide Web has established itself as the communication channel for communities all over the world in the past few years. The use of telemedical applications improves medical care by

- allowing experts from all over the world to participate in the discussion of diagnoses.
- exchanging information with other treating physicians for joint therapy planning.
- sharing expensive hardware and experts.
- giving access to reference databases.
- enabling cooperation with other researchers.
- providing possibilities for distance education and advanced training.

We developed a web-based teleradiology system for computer tomography (CT) data, magnetic resonance imaging (MRI) data and positron emission tomography (PET) data using 3D texture mapping. Recently, the use of 3D texture mapping hardware has become a powerful visualization option for direct volume rendering [1]. The rectilinear volume data is first converted to a 3D texture. Then, a number of planes perpendicular to the viewers' line of

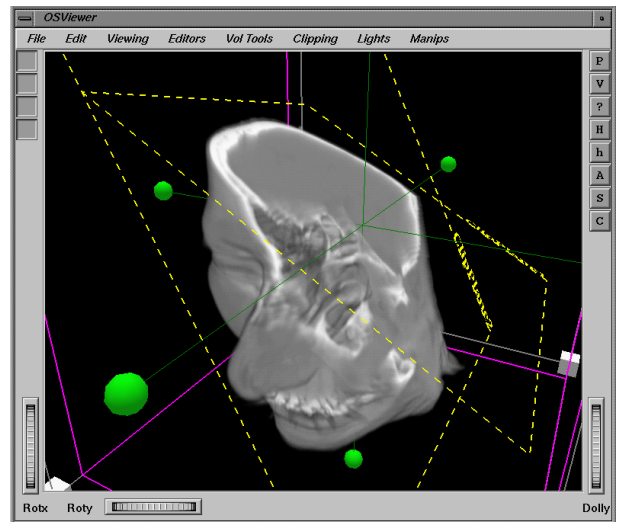


Figure 4: Visualizing a CT scan using the teleradiology system. Two clipping planes are used to show the inside of the head. Note the manipulator handles which are integrated into the visualization (small cubes and spheres).

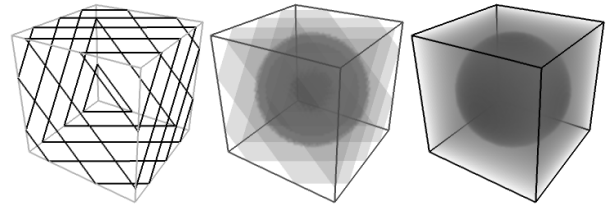


Figure 5: Volume rendering by 3D texture slicing.

sight are clipped against the volume bounding box. The texture coordinates in parametric object space are assigned to each vertex of the clipped polygons. During rasterization the slice that is cut out of the 3D texture according to the texture coordinates is mapped onto the generated fragments. The resulting polygons are projected onto the image plane using adequate blending operation to realize back-to-front or front-to-back compositing (see Figure 5). Since this process is supported by specialized graphics hardware the time it consumes decreases considerably compared to a software implementation. Thus, interactive frame rates can be achieved.

Texture mapping based volume rendering has been completely integrated into the OpenInventor framework in order to obtain the whole flexibility and functionality offered by the toolkit. By introducing a new class the volume renderer is represented as a separate object within the hierarchical structure of the scene graph. This allows convenient application of built-in manipulators, sensors, editors and other predefined classes, methods and features (light sources, anti-aliasing, stereo mode, perspective/orthogonal projection, fly, walk, trackball) [6].

The application enables the remote visualization of radiologic patient data for multiple users. The users are able to share the view onto the visualization of the patient data by synchronizing visualization parameters. A marker can be placed into the visualization to point to a region of interest. Images, created on-the-fly by a high-end visualization server, are streamed through the network to thin clients participating in a visualization session. High quality and fast generation of images are guaranteed by the use of special 3D-texture mapping hardware of a high-end server machine.

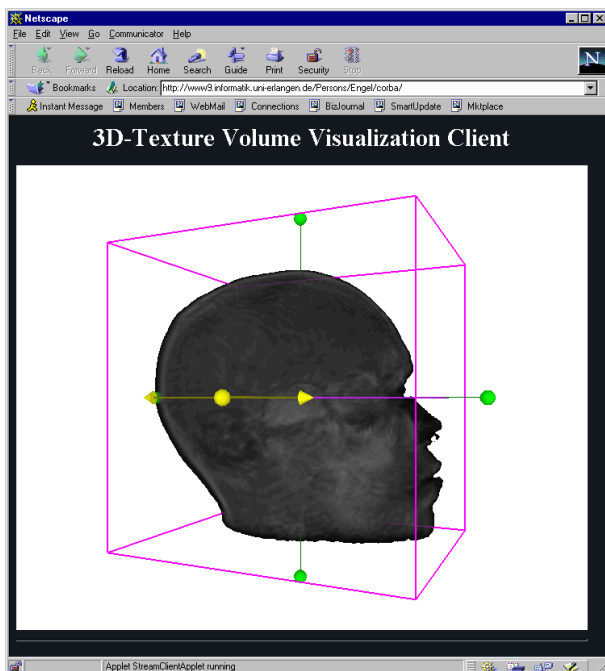


Figure 6: The Java client application running in a web-browser.

	local	raw10	raw100	codec
128x128x64	6.6	0.9/86%	3.5/47%	2.8/58%
256x256x99	3.5	0.9/74%	2.4/31%	2.1/40%
512x512x106	0.9	0.5/44%	0.8/11%	0.8/11%

Table 1: Performance of the application (frames per second) for a small, medium and large data set. A typical sequence of images generated during a visualization session was encoded to calculate the average frame rate. The second number denotes the percentage of the transfer time (including encoding and decoding) respectively to the overall time for displaying one frame.

We developed a native client X-Windows application using the C++ programming language and a Java applet. The Java applet is currently only able to receive uncompressed image data but enables access to the capabilities of the volume visualization application from a large variety of clients using a standard web-browser (Fig. 6). Pull-down menus, dialogs and other features of the stand-alone volume visualization application were reproduced on the client side. The direct interaction with the volume is done as usually by dragging manipulator handles which are integrated into the visualization of the volume data.

5 Results

In this section we show results for the proposed techniques and the exemplary teleradiology system. On the server side all tests were run on a SGI Octane MXE equipped with two 250 MHz R10000 processors and 1024 MB main memory. A SGI O2 workstation with 195 MHz R10000 processor and 128 MB main memory was serving as the client system. Both machines were linked via a 10 MBit and a 100 MBit Ethernet network connection.

First we analyze the frame rates for different images transfer methods over a 10 and 100MBit Ethernet connection for a small, medium and large dataset. Method *local* refers to local compu-

	rotation	translation
raw	1188.6	1188.6
GZIP	63.7	63.6
codec/low	6.1	5.2
codec/medium	12.7	10.6
codec/high	16.1	14.9

Table 2: Average image size (Kbytes) for different encoding methods. A sequence of 40 images was encoded while performing a volume rotation and a volume translation parallel to the image plane.

tation and display of the image sequence. Methods *raw10* and *raw100* refer to the raw transmissions of image data in rgb format over a 10 and 100 MBit connection whereas method *codec* refer to the image transmission using the streaming codec. The images were rendered at 4CIF resolution (704x576 pixels) and transferred in full resolution in method *raw10* and *raw100*. In method *codec* the images were encoded using two layers at CIF (352x288) and QCIF (176x144) resolutions. A frame at full resolution is transferred when the manipulation of the volume ends. This scheme was chosen because we could not achieve satisfying encoding rates at 4CIF resolution. A network bandwidth of 10 MBit is far sufficient for the transfer of the image stream. Nearly the same frame rate can be achieved for network connections with much smaller bandwidth.

Table 1 shows, that for the 100 MBit network connection the best frame rates were achieved using the raw transmission of image data. This is due to the fact that the encoding and decoding of images was the limiting factor in this scenario - not the network bandwidth. The local computation shows only a small performance advantage for large data sets because most of the time was spent for the rendering of the volume data. Using the 10 MBit network connection the network bandwidth becomes the limiting factor and higher frame rates can be achieved using the image streaming codec.

Table 2 compares the average image size using the different proposed compression techniques for two typical manipulations of the volume. Method *raw* refers to the uncompressed encoding of image data. The images size was 704 x 576 in RGB format which results in a raw image size of 1 216 512 bytes. Method *GZIP* refers to the encoding of single frames using the GZIP compression. The methods *codec/low*, *codec/medium* and *codec/high* refer to the proposed codec compression using a low, medium and high bit rate.

Obviously, concerning the average image size method *codec* with all bit rate settings is ahead of all other methods due to the interdependent encoding of the images and the lossy compression. This encoding method involves a reduction of image quality which may not be acceptable to medical applications. However it is possible to accept the loss of image quality when manipulating the volume and send a complete frame with full quality as soon as the manipulation is stopped. Due to the use of motion vectors for the encoding of images the codec performs better for the translation of the volume parallel to the image plane than for the rotation of the volume.

Method *GZIP* reaches good compression rates and is especially interesting using a platform-independent client. Java is able to decode GZIP streams directly through a socket connection using built-in native code very effectively.

Method *raw* requires a very high network bandwidth, usually only available in local area networks. Due to the omitting of an encoder and decoder module on the server and the client side this method is interesting for resource sharing in a local area network when a high bandwidth is available and best image quality and frame rate should be achieved.

6 Conclusions and Future Work

We have presented an image-streaming framework for the remote visualization of large scale data sets. The framework is integrated into OpenInventor. Visualization parameters of images produced by a high end visualization server are controlled using CORBA requests generated on client machines.

The introduced techniques were demonstrated by a prototypic web-based teleradiology system which makes heavy use of special 3D texture mapping hardware. There are a number of other important requirements for such a system. In this paper we mainly focused on the high-quality visualization and image transfer codec. Security issues and patient database access issues were not covered.

The introduced application is work in progress. Only the basic functionality of the powerful stand-alone volume visualization application has been made available for remote control. In the future we are going to extend the application to provide the whole functionality of the stand-alone application in the web-based application and extend the multi-user abilities. The multi-user capabilities we proposed are currently not implemented.

Currently only the natively implemented client application supports compressed image-streaming mechanisms. We are currently investigating the possibilities of using compression methods which Java provides (ZIP and GZIP compression) and implementing a decoder for the proposed streaming codec.

Another area for possible future work involves the development of specialized image-streaming codecs for on-the-fly computer rendered images. The presented codec is based on the needs of internet video transmission, which has different characteristics than rendered image sequences. It involves quality loss which may not be acceptable for medical applications.

7 Acknowledgements

We would like to thank the Telecommunications Institute of the University of Erlangen-Nuremberg for their support and for providing the source code to the image-streaming codec.

References

- [1] B. Cabral, N. Cam, and J. Foran. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. In A. Kaufman and W. Krüger, editors, *1994 Symposium on Volume Visualization*, pages 91–98. ACM SIGGRAPH, 1994.
- [2] Rikk Carey and Gavin Bell. *The Annotated VRML 2.0 Reference Manual*. Addison-Wesley Developer Press, 1997.
- [3] K. Engel and T. Ertl. Texture-based Volume Visualization for Multiple Users on the World Wide Web. In *5th Eurographics Workshop on Virtual Environments*, 1999.
- [4] B. Girod, K. W. Stuhlmüller, M. Link, and Horn. U. Packet loss resilient internet video streaming. In *Proc. Symp. on Visual Comm. and Image Proc., SPIE, (San Jose, California)*, 1999.
- [5] Jed Hartman and Josie Wernecke. *The VRML 2.0 Handbook*. Addison Wesley Developers Press, 1996.
- [6] P. Hastreiter, H.K. akmak, and Th. Ertl. Intuitive and Interactive Manipulation of 3D Data Sets by Integrating Texture Mapping Based Volume Rendering into the OpenInventor Class Hierarchy. In K. Spitzer Th. Lehman, I. Scholl, editor, *Bildverarbeitung f"ur die Medizin: Algorithmen, Systeme, Anwendungen*, pages 149–154. Inst. f. Med. Inf. u. Biom. d. RWTH, Aachen, Verl. d. Augustinus Buchhandlung, 1996.
- [7] Ofer Hendin, Nigel John, and Ofer Shochet. Medical Volume Rendering Over the WWW using VRML and JAVA. In *Proceedings of MMVR*, 1997.
- [8] U. Horn, K. W. Stuhlmüller, W. Link, and B. Girod. Robust internet video transmission based on scalable coding and unequal error protection. In *Image Com. Special Issue on Real-time Video over the Internet*, 1999.
- [9] L. Lippert, M.H. Gross, and C. Kurmann. Compression domain volume rendering for distributed environments. In *Proceedings Eurographics '97*, pages C95–C107, 1997.
- [10] C. Marrin. Proposal for a vrmf 2.0 information annex. <http://cosmosoftware.com/developer/moving-worlds/spec/ExternalInterface.html>, 1997.
- [11] James Patten and Kwan-Liu Ma. A Graph Based Approach for Visualizing Volume Rendering Results. In *Proceedings of GI'98 Conference on Computer Graphics and Interactive Techniques*, 1998.
- [12] O. Sommer, A. Dietz, R. Westermann, and T. Ertl. An Interactive Visualization and Navigation Tool for Medical Volume Data. In V. Skala, editor, *Proc. 6th International Conference in Central Europe on Computer Graphics and Visualization '98*, pages 362–371, 1998.
- [13] M.K. Uz, M. Vetterli, and D.J. LeGall. Interpolative multiresolution coding of advanced television with compatible sub-channels. *IEEE Trans. on Circuits and Systems for Video Technology*, pages 86–99, March 1991.
- [14] J. Wernecke. *The Inventor Mentor, Programming Object-Oriented 3D Graphics with OpenInventor*. Addison-Wesley, 1994.
- [15] D. Wilson and M. Ghanbari. Transmission of SNR scalable two-layer MPEG-2 coded video through ATM networks. In *Proceedings of 7th International Workshop on Packet Video*, pages 185–189, Brisbane, Australia, Mar. 1996.