GPU-Based High-Quality Volume Rendering For Virtual Environments

Andrea Kratz[*] Markus Hadwiger[†] Anton Fuhrmann[‡] Rainer Splechtna[§] Katja Bühler[¶]

VRVis Research Center

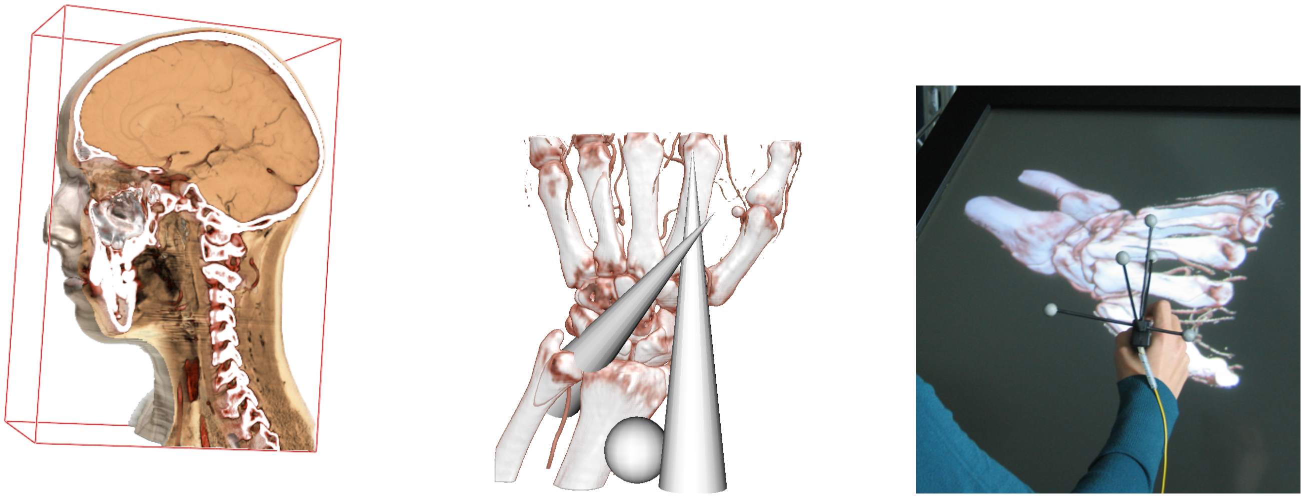

Figure 1: We seamlessly integrate perspective high-quality direct volume rendering into a polygonal virtual environment (center) by calculating pixel accurate intersections in image space. This offers an intuitive interaction using direct manipulation widgets like the clipping cube (left). The right image shows a direct volume rendering of a CT scan of a human hand, projected onto a stereoscopic display. Interaction with such a dataset of size 256 x 256 x 128 and an image resolution of 1024 x 768 can be performed at interactive frame rates.

Abstract

In medicine the knowledge of relative placement and size of anatomical structures is indispensable, especially in surgery and treatment planning. Therefore, the visualization of medical data sets acquired by Computer Tomography (CT) and Magnetic Resonance Imaging (MRI) has become more and more important because it offers a three dimensional view on the underlying data - even of inner structures. A virtual environment additionally enables a stereoscopic, and therefore natural, view. Further, input devices with six degrees of freedom provide an intuitive exploration of the data. Our application is very flexible with respect to different input and output devices, because having two equally immersive virtual environments, the user will always feel more immersed in the one more suitable to his requirements. The virtual environment used in this work is Studierstube, an enhanced virtual reality system. For volume rendering, we employ GPU based ray-casting because of its superior image quality and efficiency in comparison to slice-based approaches.

The main focus of this paper is the seamless integration of perspective high-quality direct volume rendering into virtual reality. We will present a system that realizes pixel accurate intersections between polygonal geometry and volumetric data. Intuitive interaction methods are provided at interactive frame rates including volume clipping and exploration of the dataset's interior. As main application areas we propose education and pre-operative planning.

Keywords: volume visualization, virtual reality, augmented reality

*e-mail: kratz@vrvis.at
†e-mail: msh@vrvis.at
‡e-mail:fuhrmann@vrvis.at
§e-mail:splechtna@vrvis.at
¶e-mail:buehler@vrvis.at

1 Introduction

The main concern of medical visualization is to aid clinicians in their daily work, for example for simulating unusual interventions or training a surgery. Collaborative environments could encourage multidisciplinary work because it assists the communication between colleagues from different fields.

Interactive frame rates in combination with direct volume rendering and a stereoscopic view is not an easy task, as the whole scene has to be rendered twice. However, a high level of interactivity is absolutely essential when talking about virtual reality (VR) applications because the user should have the feeling of being immersed in the system. Furthermore it is of particular importance that the user is able to manipulate the virtual object in a natural way which is only possible at interactive rates. On the other hand, the image quality in medical applications is also very important because it is the basis for any kind of diagnosis. As the visualization should facilitate the evaluation of relative placement and scale of anatomical structures, a perspective projection is required.

The project presented in this paper integrates with one of our previous projects [8] where software-based volume rendering was integrated into the Studierstube framework. We seamlessly integrate high-quality perspective volume rendering into a polygonal virtual environment (VE) by considering accurate occlusions between the objects although they are created by completely different rendering frameworks. Furthermore, we overcome all the shortcomings of the previous project, namely:

- Software volume rendering with parallel projection
- Absence of correct occlusions
- Artifacts caused by network delay

The structure of this paper is organized as follows. Initially, related work is presented in section 2. Then the two libraries that were combined are introduced, focusing on the volume rendering rendering framework and its features in section 4. Section 5 deals with the integration of volume rendering into the Studierstube framework. Special attention is paid to intersections between opaque polygonal geometry and volumetric data. Next, VR interaction methods and manipulation widgets are presented in section 6. Section 7 deals with a discussion about different input and output devices. Finally, the last two sections focus on a summary of the results achieved (see section 8) and conclude with an outlook into future work in section 9.

2 Related Work

The advantages of a virtual environment (VE) as user interface for visualization applications have been demonstrated previously: Cruz-Neira [4] demonstrated the application of their CAVE virtual environment in scientific visualization. The useful combination of virtual reality and visualization was demonstrated in [23] and in our own adaptation of a commercial visualization system for virtual reality [7]. The impact of VR applications in medicine is discussed in [16]. Especially in the field of volume visualization, the application of direct manipulation widgets [12], geometric objects exhibiting interactive behavior, makes navigation and manipulation intuitive, efficient and informative. The use of 6 degree-of-freedom (6-DoF) input hardware allows effective control of these widgets and stereoscopic display devices deliver the necessary feedback for 3D interaction and ease the interpretation of the volume data. Previous approaches that integrated volume rendering into VR, like [19] and [8] use software approaches for volume rendering and did not yet realized a seamless combination of the different rendering techniques. [13] focus on the design of transfer functions within a VE.

3 Virtual Reality Framework

Studierstube [18] is an enhanced virtual reality (VR) system which we already applied to problems in the area of scientific visualization [7], [6]. It implements basic interaction methods like positioning objects by dragging them with a 6-DoF pen as well as conventional 2D interaction elements, like sliders, dials, and buttons for parameterizations of visualization methods. These purely virtual interface elements are positioned on the Personal Interaction Panel [21], a handheld tablet. Users hold this board in their non-dominant hand while they make adjustments to the interface elements with the same pen they use for 6-DoF interaction. Studierstube supports a wide range of input and output devices and is thus configurable for many different hardware setups. Studierstube relies on polygonal rendering via OpenGL and is based on the high level API Coin3D [11], an object-oriented C++ class library.

4 The HVR framework: high-quality hardware volume rendering

Volume rendering has become more and more popular in computer graphics. Although it is an impressive methodology to explore volumetric data it is a computationally expensive task. Our HVR framework [17] for volume rendering combines high-quality with real-time performance on consumer graphics hardware. It provides custom rendering modes, like iso surface rendering, shaded and unshaded direct volume rendering as well as more sophisticated rendering modes, described in section 4.2. The image quality can be reduced during interaction. Therefore interactivity is also ensured for large datasets. Settings like color, opacity, iso value and modifications in the transfer function can be done in real-time. Perspective rendering is supported with no performance decrease. Viewing the volume from the inside or fly-throughs are also provided.

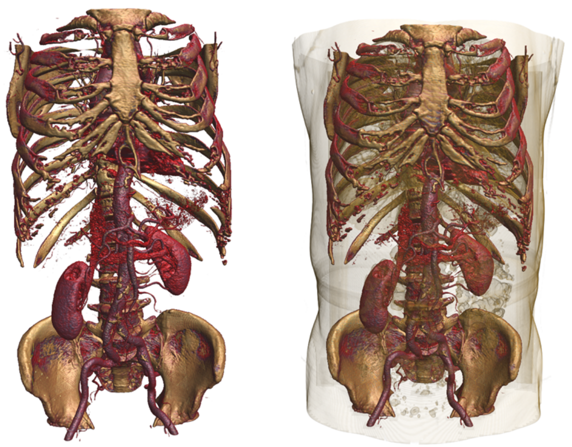

Figure 2: High-quality direct volume rendering of a CT scan of a human torso rendered by the HVR framework using GPU ray-casting. Image size is 512x512x1112.

4.1 GPU Ray-Casting

Ray-casting renders an image of a volume by casting rays from the view point through each pixel, and performing resampling and compositing of successive sample contributions along these rays [15]. Samples are usually taken at equispaced intervals, which are determined by the desired sampling rate. Fast software implementations rely on ray templates to determine the sample locations, which can only be used with orthogonal projection. In contrast to software implementations, GPU ray-casting can easily support perspective projection without decreasing performance [14]. The entire volume is stored in a single 3D texture, and resampling is performed by fetching samples from this texture with trilinear texture filtering. For each pixel, a fragment program steps from sample location to sample location in a loop and performs compositing until the volume is exited or full opacity is reached. In order to be able to step along rays, however, the ray entry positions, directions, and lengths must be computed. The HVR framework performs this ray setup by rasterizing two images [17]. The first image determines volume entry positions by rasterizing the front faces of the geometry that bounds the non-transparent parts of the volume (Figure 3, left).

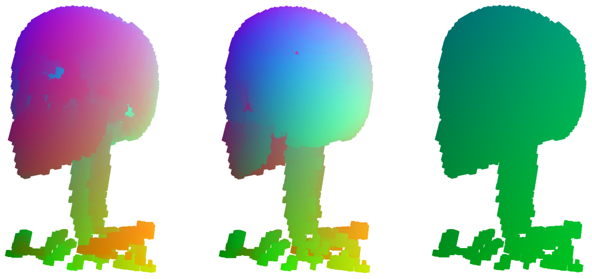

Figure 3: Geometry setup for the ray-casting process. For optimization, the complexity of the bounding box is adapted to the underlying dataset. Front faces are shown in the left image, back faces in the center image and the direction texture in the right image.

The second image is generated by rasterizing the back faces of this bounding geometry (Figure 3, center), but already subtracts the first image that contains the entry positions and thus yields ray direction vectors and lengths (Figure 3, right). Ray directions are stored in the RGB components, and the ray length is stored in the alpha channel. Rasterizing bounding geometry of non-transparent regions of the volume performs *empty space skipping* and improves performance considerably. The compositing fragment program, which performs actual ray-casting, uses the two ray setup images to determine sample locations. The volume is resampled, and the samples are mapped to optical properties via a *transfer function*, which in the simplest case is a 1D texture that contains the emission (RGB) and absorption (A) of light corresponding to the sample's volume density. When the volume is exited as determined by the ray length, or the cumulative absorption reaches 1.0 before (*early ray termination*), ray traversal is terminated and the composited color value is the solution of the volume rendering integral for this ray.

4.2 Enhanced Rendering Modes

4.2.1 Iso-surface shaded Direct Volume Rendering

Although shaded DVR produces images that look very realistic it has to deal with a few significant drawbacks. Volume data often consist of homogeneous areas where the gradient is undefined, which results in noisy images. On the contrary, unshaded DVR visualizes boundaries very well but with no depth perception which makes it difficult to recognize the relative placement of inner structures. The volume rendering framework provides a rendering mode that overcomes these problems by combining shaded iso-surface and unshaded direct volume rendering [17]. This visualization method is of great interest for the use in a virtual environment. It improves the rendering quality significantly but also enables to change the appearance of the volume without modifying the transfer function. The user is able to create meaningful images just by moving a slider on the virtual tablet.

4.2.2 Virtual Endoscopy

The HVR framework is also suited for virtual endoscopy applications [17]. It offers the possibility to explore the dataset's inside, while performing fly-throughs. In a virtual environment this enables the user to move his own head into the virtual one for exploration of inner structures.

5 Integration

In this section the integration of hardware volume rendering into the Studierstube framework is described. It can be separated into two main steps. First, the view of both systems has to be synchronized and the images created by the volume renderer have to be rendered into the context of Studierstube. Second, correct intersections have to be considered to provide the user with a kind of depth perception.

5.1 Image transfer and Viewing

The viewing, modeling and projection matrices are defined by Studierstube. The modelview part is obtained from tracking data, where the movement of the input device defines the modeling transformation and the position of the user's head the viewing matrix. If head tracking is supported by the underlying VR system it is even the head's movement. For this reason, viewing and modeling are separate steps in the Studierstube Framework. Before the whole scene can be drawn, the modelview projection matrix for the HVR framework has to be set accordingly. Therefore the matrices retrieved from the VR framework are multiplied into one single matrix, as shown in equation 1.

$$MV = PEN_{rot} \cdot PEN_{trans} \cdot VM \quad (1)$$

where PEN_{rot} and PEN_{trans} are representing the relative movement of the input device, VM the viewing matrix and MV the modelview matrix. The projection matrix is passed to the renderer in a separate step.

To render the images created by the HVR framework within the scene graph data structure of Studierstube, the volume rendering is embedded into a callback node and performed entirely off-screen. The content of the off-screen buffer is rendered into a dynamic texture and blended back to the screen in the very end.

5.2 Geometry Intersections

For a seamless combination of volume and polygonal rendering, occlusions between volume and OpenGL geometry have to be considered to create a perception of space. Those intersections are essential for an intuitive interaction with the volume which should be as natural as possible. In short, this means that rays either are not permitted to start - before the ray-casting process - or have to be terminated early - as soon as they hit the geometry.

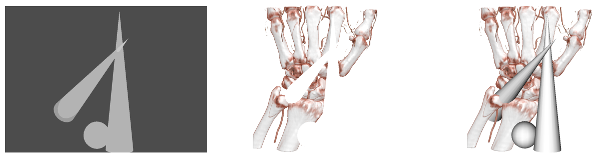

Figure 4: Intersections between volume data and OpenGL geometry exploiting the depth image of the geometry (left). During rendering, rays are stopped at the depth of the corresponding intersection points (center). Blending the geometry image on top of the volume yields the final image including correct intersections (right).

5.2.1 Algorithm Overview

To achieve geometry intersections, the rendering is split into two passes, where the first one renders off-screen producing the final image and the second one writes this image into the framebuffer, so it becomes visible for the user. The idea is to calculate the positions of the scene geometry based on the screen coordinates and the depth of the current fragment. Therefore we have to perform a backprojection by multiplying the $(screen_x, screen_y, depth, 1)^T$ coordinates with the inverse modelview projection matrix, as shown in equation 3. As the $(screen_x, screen_y)^T$ coordinates and the depth are assumed to be in the range of $[0,1]$, they first have to be transformed into normalized device coordinates.

$$\begin{aligned}
\begin{pmatrix} volume_x \\ volume_y \\ volume_z \\ w \end{pmatrix} &= (MVP)^{-1} \cdot NDC \qquad (2) \\
NDC &= \left(\begin{bmatrix} screen_x \\ screen_y \\ depth \\ 1 \end{bmatrix} * 2 - 1 \right) \\
\begin{pmatrix} screen_x \\ screen_y \end{pmatrix} &\in [0,1]^2 \\
NDC &\in [-1,1]^3 \\
depth &\in [0,1]
\end{aligned}$$

The result of the equation above are the positions of the scene geometry in volume coordinates, i.e. in texture coordinates between $[0,1]^3$. The last step is the division by the homogeneous coordinate w, shown in equation 3.

$$\begin{pmatrix} volume_x \\ volume_y \\ volume_z \\ 1 \end{pmatrix} = \begin{pmatrix} volume_x/w \\ volume_y/w \\ volume_z/w \\ 1 \end{pmatrix}, \qquad (3)$$

Now the generation of the direction texture has to be modified by subtracting the front face positions from these values. This yields to a new direction texture that implicitly contains the new stop positions. Figure 5 illustrates the positions of the scene in volume coordinates.

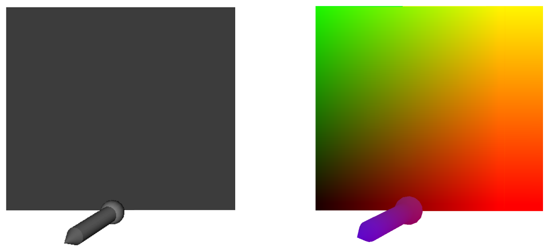

Figure 5: Position of the geometry in volume coordinates. Terminating the rays at those positions leads to a clipped volume, as shown in the center image of figure 4.

5.2.2 Implementation Details

In the first rendering pass the geometry is rendered into a depth image and a 2D RGBA color texture, respectively. Simultaneously, the clipped volume is rendered into a second 2D RGBA texture. Finally, the geometry image is blended on top of the clipped volume which yields to the final image including correct intersections. For off-screen rendering and separate generation of the different color and depth images, we utilize the EXT_framebuffer_object OpenGL extension [9]. Backprojection and direction texture generation are done in one additional fragment program within the HVR framework (see figure 6).

```
uniform vec2 window_position;
uniform sampler2D depth_texture;
uniform sampler2D   front_faces;

void main()
{
 // compute the homogeneous view-space position
 float4 hviewpos;
 hviewpos.xy = window_position;
 hviewpos.z = tex2D(depth_texture, window_position);
 hviewpos.w = 1.0;
 // transform into normalized device coordinates
 hviewpos = hviewpos * 2.0 - 1.0;
 // back-project to homogeneous volume space
 float4 hvolpos = gl_ModelViewProjectionMatrixInverse * hviewpos;
 // return normalized volume-space position
 return (hvolpos / hvolpos.w);
}
```

Figure 6: GLSL function that calculates the position of the scene geometry in volume coordinates.

6 VR INTERACTION WITH VOLUME DATA

Our main interest is in medical education and neurosurgery. The system should help medical doctors in diagnosis and surgical planning. It could aid in collaborative work among surgeons and radiologists and facilitate communication, even between patients and medical doctors. As a traditional slice view obtained from image acquisition devices, as for instance a CT, could mostly only be interpreted by radiologists, the volume rendered image can also be understood by non-experts. In education, the system should help medical students to understand anatomical structures and their placement. Remembering the requirements of VR, the user should have the possibility to explore the data in the most natural way, so interaction and navigation have to be intuitive. For interaction, the possibility to cut away parts and to look into the volume is very important. Hence we have to provide clipping. The user has to be provided with an immediate feedback during interaction by highlighting the direct manipulation widgets. Furthermore, we should keep clearly in mind that VR also deals with some accuracy problems caused by the tracker device. This could complicate especially the design of transfer functions which is a highly accurate user interaction. Although our optical tracking system is very precise, we leave the design of transfer functions in 2D and provide to load them during runtime. The appearance of the volume within the VE can be modified when using iso-surface shaded direct volume rendering, as described in section 4.2.1, and changing the iso-value by moving a slider. This offers an intuitive way to visualize different features and structures. The basic interaction elements realized in this work are navigation and selection using the input device. The system is controlled using widgets (sliders and buttons) on the Personal Interaction Panel.

6.1 Navigation

The most obvious direct interaction method is navigation. The user is able to move the volume by positioning the input device inside the volume's bounding box and then pressing the primary button on the input device. The object follows all motions and rotations of the input device as long as the user pushes the button.

6.2 Clipping Cube Widget

Clipping means cutting of graphical elements at user-specified lines, curves, axis-aligned and arbitrary aligned planes. By means of clipping, it is decided, which parts of an object (a scene) are within a clipping area and therefore visible - and which are not. We provide axis aligned clipping which is realized in terms of a clipping cube widget. Each face represents one of the six clipping planes. It is represented by a bounding box, with invisible spheres at the cube's corners. When moving the input device into one of the spheres the user is given an immediate visual feedback by highlighting the bounding box, so he knows that he is able to move the clipping planes. This causes a resize of the cube, keeping it axis-aligned.

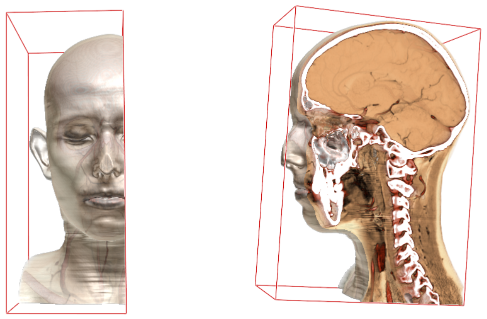
Figure 7: Clipping Cube Widget. Rendering Mode is iso-surface shaded direct volume rendering.

6.3 Lighting Widget

The user is able to change the lights direction and its intensity. A sphere indicates the light source. It can be transformed in the same way as the volume object by moving the input device inside the sphere and pushing the primary button. The distance between the sphere, i.e. light source, and the center of the volumes bounding box defines the intensity. The spheres position defines the light direction.

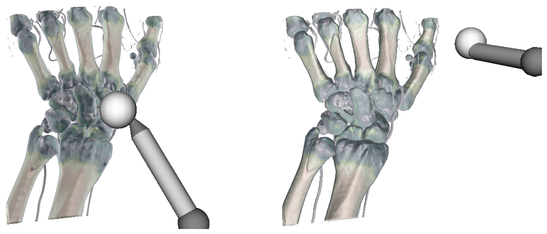
Figure 8: Lighting Widget. Rendering Mode is iso-surface shaded direct volume rendering.

6.4 Slice View Widget

The slices are provided in a 3D texture by the volume renderer. To display one slice, the appropriate texture coordinates have to be calculated. The rendering of the slices is realized on the virtual tablet. By moving a slider on a three dimensional coordinate system the user is able to create a cut-away view of the volume by viewing the corresponding slice (figure 9, right image), simultaneously.

6.5 Fly-throughs

As mentioned in section 4.2.2 the HVR framework is suited for virtual endoscopy. For our application this means that the user is given two different views on the dataset. We provide a traditional view on the outside of the volume and additionally its exploration from the inside.

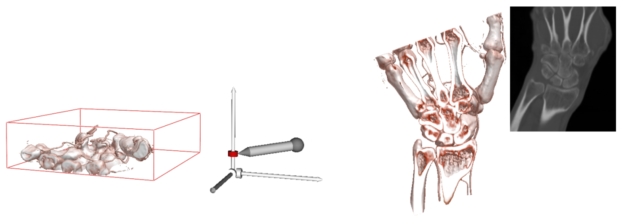
Figure 9: Slice View Widget. By moving a slider on a three dimensional coordinate system the user is able to see the according slice. Rendering Mode is iso-surface shaded direct volume rendering.

7 Hardware Setup

Our prototype system 10(a) consists of a swivel mounted Barco Baron Table [2] with shutter glasses synchronized by an infrared emitter, an ART Optical Tracking System [1] with three cameras and a single PC running our hybrid application. The PC is equipped with an AMD Athlon 64 3800 CPU and a NVIDIA GeForce Quadro FX 3400/4400 graphics card. The optical tracking system delivers data with an error well below 1mm, which allows precise interaction using our pen and pad combination. Tracking markers - small reflective spheres - are mounted on the Baron Table, the shutter glasses, and the handheld tablet, and allow wireless tracking of head and interaction devices. Tracking the table allows swiveling it during operation without losing registration, an operation which allows to adapt the working volume to different tasks. We apply our full

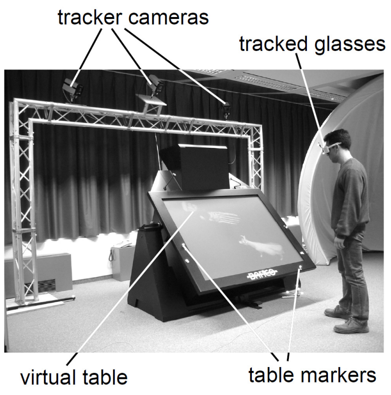

Figure 10: Hardware Setup

set-up for educational purposes and in pre-operative planning. For intra-operative applications, we rely on simpler hard- and software combinations: Since most surgeons do not want to wear a pair of tracked shutterglasses or use our pen and pad interaction - even if it was possible to guarantee sterility of the devices - we have to fall back to what is essentially a desktop virtual environment. We use a SeeReal [22] autostereoscopic monitor (see figure 11) and a mouse interface, remotely controlled by a nurse or - during intra-operative consulting - a radiologist. The monitor guarantees a stereoscopic display with contrast high enough to work in a surgical environment and displays the volumetric data as visualized in the pre-operative planning phase on the full set-up. The surgeon does not have to wear any glasses, which would impair his field-of-view and dim the view of the operation field. The monitor can be used with a fixed viewing zone, or with a tracking device, which re-adjusts the viewing according to the surgeon's position. This works best when the surgeon wears a retro-reflective marker on his head, which can easily be fixed to a sterile cap.

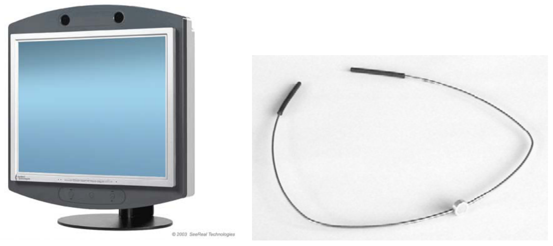

Figure 11: SeeReal auto stereoscopic monitor (left) and reflector spot (right). Image taken from [22].

8 Results

We seamlessly integrated high-quality perspective volume rendering into a polygonal VE by considering accurate occlusions between the objects although they are created by completely different rendering frameworks. Furthermore, we overcome all the shortcomings of our previous project [8], namely:

- Parallel Projection
- Absence of correct occlusions
- Artifacts caused by network delay

The table below gives a summarized view of the measured frame rates using the following system:

- single processor PC, AMD Athlon 64 3800
- equipped with a NVIDIA Quadro FX 3400/4400 graphics card
- GPU memory size 256 MB
- running Windows XP
- on a virtual table

Dataset	Rendering Mode	Mono [fps]	Stereo [fps]
Hand	ISO	20 fps	11 fps
Hand	DVR + ISO	17 fps	10 fps

Table 1: Frame rates comparing different render modes (iso surface and combined shaded iso surface and unshaded direct volume rendering) and mono and stereo settings. The frame rates are measured with an image resolution of 1024x768. The hand dataset is of size 256x256x128.

As shown in table 1 we are able to achieve interactive frame rates. Performance in the HVR framework is scalable, so the user is able to reduce the image resolution during interaction (like clipping using the clipping cube widget) and change it again to high resolution to view the clipped volume in its best quality again.

9 Conclusion and Future Work

We have presented the integration of perspective high-quality hardware based volume rendering into a virtual environment. We have achieved a seamless integration of the volumetric data into the polygonal rendering of the VE by calculating pixel accurate intersections of both in image space. Besides the integration, we discussed navigation, interaction and direct manipulation widgets.

The integration of volume rendering into the polygonal rendering of Studierstube was realized by performing the entire volume rendering off-screen. Occlusions between volume and opaque OpenGL geometry were achieved by rendering a depth image of the geometry and terminating the rays in the ray-casting process at exactly that depth in volume coordinates. This extension offers a native combination of volume and polygonal rendering.

Navigation and Interaction are based on the Studierstube API. We provide axis-aligned clipping, intuitive lighting by directly moving the light-source and a traditional slice view. The interaction methods are aided by visual feedback. Although a sophisticated widget for transfer function design, like the one proposed by [12] could be integrated easily into our VR framework, we propose to leave the accurate design on standard desktop PCs. Pre-defined transfer functions and arbitrary datasets can be loaded during runtime, anyway.

Typical VR navigation and interaction methods, like 3D magic lenses [24] or world-in-miniatures [20] turned out not to be useful for this kind of application which should be used in the medical domain. Therefore upcoming work should concentrate on VR interaction methods that fit these special requirements. It will also focus on a hybrid solution that provides a stereoscopic view on a passive stereo screen and a second 2D display providing a standard QT user interface. As input devices we will provide a space mouse that offers 6-DoFs for 3D navigation and a PC mouse to serve the user interface.

Another interesting aspect could be to focus on the evaluation of different 3D rotation techniques. For instance, the work of [3] focuses on 3D rotation especially in the field of medical applications. Their work is based on the question, why medical doctors still prefer to read CT or MRI data slice by slice, although 3D visualizations offer a great potential for medical applications and could ease the clinical work flow. Their assumption is, that one reason for this is due to unintuitive 3D rotation techniques.

We will continue the evaluation of our system in collaboration with our medical partners. Further, we plan a cooperation with the Vienna General Hospital for teaching purposes, because although there is a lot of research in medical visualization all too often these techniques are either not known by clinicians or not accepted. An early introduction to those new technologies in education would familiarize students to them already during their studies which could lead to more acceptance. Furthermore the use in education could aid in validation. Preliminary demonstrations for medical students and neurosurgeons have resulted in positive feedback.

Acknowledgments

This work has been carried out as part of the basic research on medical visualization and the FemInVis project (http://medvis.vrvis.at/projects/feminvis/) at the VRVis Research Center for Virtual Reality and Visualization in Vienna (http://www.vrvis.at), which is funded in part by the KPlus program of the Austrian government. The medical data sets are courtesy of Tiani Medgraph (http://www.tiani.com/) and the Department of Neurosurgery at the Medical University Vienna. The first author also wants to thank Florian Schulze for his valuable comments regarding this paper.

References

[1] Advanced realtime tracking gmbh, germany. a.r.t. infrared tracking system. information brochure, April 2001. See `http://www.ar-tracking.de/`.

[2] Barco, belgium. baron high-performance projection table. information brochure, July 2002. See `http://www.barco.com/`.

[3] Ragnar Bade, Felix Ritter, and Bernhard Preim. Usability comparison of mouse-based interaction techniques for predictable 3d rotation. In *Smart Graphics*, pages 138–150, 2005.

[4] C. Cruz-Neira and others. Scientists in wonderland: A report on visualization applications in the CAVE virtual reality environment. *Proceedings of the IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, pages 59–66, 1993.

[5] Timothy J. Cullip and Ulrich Neumann. Accelerating volume reconstruction with 3d texture hardware. Technical report, Chapel Hill, NC, USA, 1994.

[6] Anton Fuhrmann and Eduard Gröller. Real-time techniques for 3d flow visualization. In *VIS '98: Proceedings of the conference on Visualization '98*, pages 305–312, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[7] Anton Fuhrmann, Helwig Löffelmann, and Dieter Schmalstieg. Collaborative augmented reality: exploring dynamical systems. In *VIS '97: Proceedings of the 8th conference on Visualization '97*, pages 459–ff., Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[8] Anton L. Fuhrmann, Rainer Splechtna, Lukas Mroz, and Helwig Hauser. Distributed software-based volume visualization in a virtual environment. In *Proceedings of the 11th EUROGRAPHICS Workshop on Virtual Environments (EGVE 2005)*, pages 129–140, 2005.

[9] Silicon Graphics. OpenGL Extension Registry, 2005. See `http://www.oss.sgi.com/projects/ogl-sample/registry/`.

[10] Markus Hadwiger, Christian Sigg, Henning Scharsach, Katja Bühler, and Markus Gross. Real-Time Ray-Casting and Advanced Shading of Discrete Isosurfaces. *Computer Graphics Forum*, 24(3):303–312, 2005.

[11] Systems in Motion AS. Coin3d, 2005. See `http://www.coin3d.org`.

[12] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 255–262, 2001.

[13] Joe Kniss, Jürgen P. Schulze, Uwe Wössner, Peter Winkler, Ulrich Lang, and Charles D. Hansen. Medical applications of multi-field volume rendering and VR techniques. In *VisSym*, pages 249–254, 350, 2004.

[14] Jens Krüger and Rüdiger Westermann. Acceleration techniques for GPU-based volume rendering. In *IEEE Visualization*, pages 287–292, 2003.

[15] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, May 1988.

[16] G. Riva. Applications of virtual environments in medicine. *Methods of Informatics in Medicine*, 42(5):524–34, 2003.

[17] Henning Scharsach, Markus Hadwiger, Andre Neubauer, Stefan Wolfsberger, and Katja Bühler. Perspective Isosurface and Direct Volume Rendering for Virtual Endoscopy Applications. *Proceedings of Eurovis/IEEE-VGTC Symposium on Visualization 2006*, 2006.

[18] Dieter Schmalstieg, Anton L. Fuhrmann, Gerd Hesina, Zsolt Szalavári, L. Miguel Encarnação, Michael Gervautz, and Werner Purgathofer. The studierstube augmented reality project. *Presence*, 11(1):33–54, 2002.

[19] Jürgen P. Schulze, Roland Niemeier, and Ulrich Lang. The perspective shear-warp algorithm in a virtual environment. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 207–214, Washington, DC, USA, 2001. IEEE Computer Society.

[20] R. Stoakley, M. Conway, and R. Pausch. Virtual reality on a WIM: Interactive worlds in miniature. In *CHI'95*, pages 265–272. 95.

[21] Zsolt Szalavari and Michael Gervautz. The personal interaction panel - a two-handed interface for augmented reality. *Computer Graphics Forum*, 16(3):335–346, 1997.

[22] SeeReal Technologies. Seereal technologies, 2006. See `http://www.seereal.com/`.

[23] Andries van Dam, Andrew S. Forsberg, David H. Laidlaw, Joseph J. LaViola Jr., and Rosemary Michelle Simpson. Immersive VR for scientific visualization: A progress report. *IEEE Computer Graphics and Applications*, 20(6):26–52, 2000.

[24] John Viega, Matthew J. Conway, George Williams, and Randy Pausch. 3D magic lenses. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, Papers: Information Visualization, pages 51–58, 1996.