Volume Rendering on Stereo Display with Virtual Object Manipulation

Julius Adorf, Daniel Berglund, Oleg Kravchuk *

DD2257 Visualization
KTH Royal Institute of Technology, Stockholm

May 27, 2013

Abstract

This article presents a GPU-based interactive volume rendering application that projects 3D scans such as obtained from a computer tomography onto a display. A user wearing anaglyph glasses can perceive the virtual object on a standard screen in depth. Stereo is also supported for a 4K screen. A configurable transfer function allows the user to highlight different parts of the virtual object. The user can control the object pose with a single hand in front of a OpenNI-compliant depth camera.

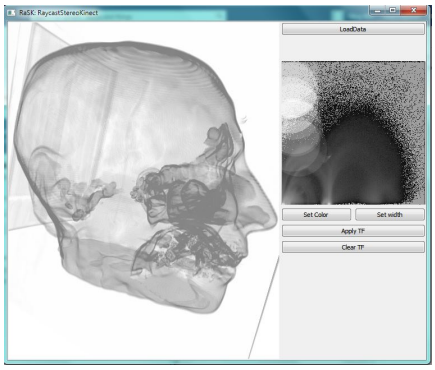

Figure 1: The application window shows the virtual object on the left. The user may customize the transfer function on the right panel.

1 Introduction

First, purely two-dimensional visualization methods are weak at the visualization of data that is inherently three-dimensional. For example, a human looking at single slices of a computer tomography scan might have difficulties to perceive depth. It is probably much more natural for the user to see the sliced data rendered as a volume.

Second, in some applications the user needs to be able to rotate and move a virtual object in three-dimensional space. Often, a mouse is used as an input method. However, it might be beneficial if the user can control the pose of the virtual object in real three-dimensional space using hand gestures.

2 Related work

The idea of manipulating the pose of a virtual object with a single hand is inspired by a touchscreen interface developed by Fujitsu [1]. The volume rendering implementation works on similar principles as described in a tutorial on Real-Time Volume Graphics at Eurographics 2006 [2].

*Names in alphabetic order.

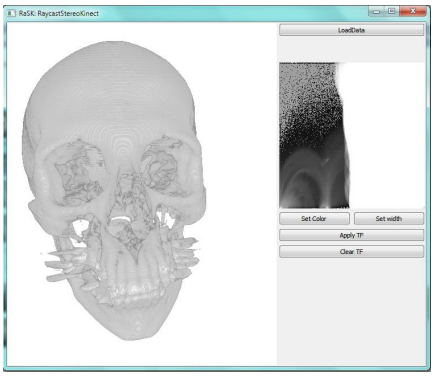

Figure 2: The transfer function can be used to highlight the bone structure of the head of a human. The colors can be customized by the user.

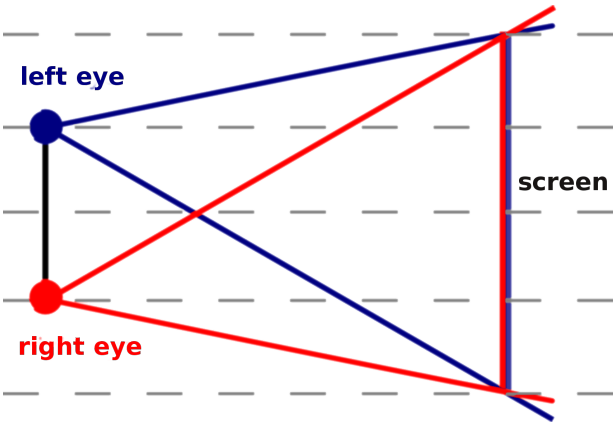

Figure 3: The cameras for each eye are chosen to have parallel axes. Perspective projection is used to obtain the screen projections.

3 Volume rendering

Volume rendering is a crucial step in this application. We obtain a rendered volume from the data slices by means of interpolation and GPU-based ray-casting targeted for real-time applications. In the implementation, we use OpenGL [3] for rendering and Qt [4] for the graphical user interface. The rendering happens within the unit cube. The start points on the front face of the cube and the exit points on the backface are computed first. These determine the direction of the rays.

A transfer function that can be graphically edited with a brush enables the user to highlight certain parts. Figure 1 and Figure 2 show the same object with different transfer functions.

4 Stereo display

The user can perceive the depth of the virtual object, which is rendered to two separate images for display in stereo. The two images are obtained from the virtual object by *parallel axis asymmetric frustum perspective projection* (see Figure 3).

The classic anaglyph 3D technique is used for standard screens. The images for each eye are color-coded. In order to perceive depth, the user has to wear anaglyph glasses.

The Visualization Lab at Royal Institute of Technology in Stockholm has a high-resolution (4K) screen illuminated by two digital projectors. These two projectors can be used to create a 3D effect. The user needs to wear polarizing glasses. Each projector is controlled by a node running a client application. A server application on another node renders the images and sends them to the clients for display. Our client-server solution uses Zeromq [5].

5 Virtual object manipulation

The user can control the position and the orientation of the virtual object either via the mouse or via gestures with a single hand in front of a depth camera.

Waving a hand initiates interaction with the virtual object. The initial position of the hand serves as a reference position. Moving one's hand closer to the camera corresponds to zooming, whereas moving one's hand in vertically or sideways controls rotation around two axes.

Any OpenNI-compliant depth camera is supported

Figure 4: The user controls position and orientation of the virtual object by moving one hand in front of a depth camera.

since our system acquires depth data through the OpenNI [6] platform. We tested the system successfully with both an ASUS Xtion Pro camera (see Figure 4 and a Microsoft Kinect camera. The NiTE 2 [7] body tracker provides estimates of the hand position. In order to make the object pose less susceptible to noise and shaking hands, we compute the weighted moving average of a sequence of hand positions. This venhances user experience.

6 Conclusion

Great efforts have been taken to create a system that renders data slices as a volumetric object and displays this virtual object on either a standard screen or on a specialized high-resolution screen with two projectors via client-server communication. The visualization can be interactively modified by either changing the transfer function or by changing the pose of the object. Zoom and rotation can be controlled by both mouse and single-hand gestures in front of a RGB-D camera.

Even though the whole system is working, we can think of various improvements. The stereo display on the high-resolution screen can be made faster. The initialization gesture for hand tracking should be more convenient. Judging from the feedback we got, it might be worth experimenting with different mappings from hand gestures to rotation and zooming activities.

The project was technically challenging because many subsystems needed to be implemented or integrated in order to make all these features possible. Techniques and knowledge from various fields were required: computer graphics, computer vision, signal processing, parallel computing, to mention some of them. In summary, the project would not have been possible with so little time at hand without combining the skills of engineering students with different backgrounds.

References

[1] Stan Schroeder. `http://mashable.com/2013/04/16/fujitsu-paper-touchscreen`. Accessed on 2013-05-23.

[2] Markus Hadwiger. `http://www.cg.informatik.uni-siegen.de/data/Tutorials/EG2006/RTVG04_GPU_Raycasting.pdf`. Accessed on 2013-05-27.

[3] OpenGL. `http://www.opengl.org`. Accessed on 2013-05-23.

[4] Qt. `http://qt-project.org`. Accessed on 2013-05-23.

[5] ZeroMQ. `http://www.zeromq.org`. Accessed on 2013-05-23.

[6] OpenNI. `http://www.openni.org`. Accessed on 2013-05-23.

[7] NiTE. `http://www.openni.org/files/nite`. Accessed on 2013-05-23.