Assessment and evaluation of a light field display for medical data visualization

Jonas Nilsson

May 26, 2008

Abstract

This thesis explains how a medical volume render application is constructed on a holographic light field display and how such an system should be evaluated to be useful as a medical display system. The important aspects regarding human depth perception is reviewed and used to form meaningful test scenarios for the light field display.

Thanks to my family and friends in Sweden for showing that I have somewhere to call home, my friends in Sardinia for putting my mind on other things then rendering and displays, my colleagues at CRS4 for keeping up a good spirit and providing knowledge and aperitive, the Mediterranean climate for the blue sky, Sardinia for the mountains and beaches and Sweden for the stability.

Contents

Chapter 1

Introduction

1.1 Context

Volumetric data is an increasingly important source for medical diagnosis. The power of modern computers offers a lot of possibilities of rendering three-dimensional data to enable viewing on the display. The use of three dimensional data in analysis is promising since the human eye is extremely well adapted for determining distances and structures, since we live in a three dimensional world. Displaying the volumetric data in meaningful way requires different kinds of methods than for two-dimensional images. Such methods have been possible to use in real time for the last couple of years thanks to the computing power of modern graphics processing units. However, much of the information is lost in the process of projecting the volumetric data onto a two-dimensional monitor since 2D projection put down a lot of the cues which humans use in depth perception. It is possible to use stereoscopic display systems for this projection and thereby add some more cues to display. But this projection is still two-dimensional, which means that some important cues will still be left out. The mix of different, often contradicting, cues can in worst case cause sickness and other uncomfortable effects (described in [25]).

The data must be displayed as a real-world volume in order to enable a human to fully understand a volume data set. This can be achieved by using volumetric display systems. Most of these are still very experimental and can not display large data sets. The computational load is many times higher on these kinds of displays than a 2D display since a volumetric display requires a rendering of multiple of views for each frame. Displaying a volumetric rendered scene in such a display will increase the load even further. But the increasing computing power of modern desktop PCs have now made it possible to render a volumetric scene using ray casting algorithms and complex shading on a volumetric display with one

high performance PC. This is a promising development to introduce volumetric displays in hospitals and give physicians a better view inside the human body.

The system used in this investigation is a small (30inch) multi projections system that creates a light field of the scene which is displayed to the viewer through a special light bending screen. The viewer will receive a smooth blend of the light field which creates an image of the rendered object as it should appear from the viewers position. This display can be driven by a single modern consumer PC and can be considered as a replacement of a regular LCD or CRT display for certain applications. However, the display is still a complex and expensive system which must perform much better then a regular display to be a worthwhile investment.

The requirements for medical display systems is understandably very high. Radiologists require high resolution and high contrast images to facilitate a correct diagnosis. The requirements for a holographic system should be similarly high and must therefore be evaluated using similar tools as to any other medical display. However the depth perception is not evaluated using these tools as the evaluations is concentrated on two-dimensional xray images. Other tests must therefore be used to determine the quality of the depth cues displayed by the system.

1.2 Objectives

The purpose of this thesis is to determine if the light field display can be used as a medical display system for showing medical volume data in a way which helps doctors make a diagnose. This is a broad objective which must be split into several parts in order to be solved.

The construction of a volume rendering application which can display volume datasets on the light field display must be described. The second objective is to evaluate the displayed image to determine wether the screen's limits will affect image quality severly. The resulting images should be useful for medical diagnosis and should therefore be compareable with images displayed on a regular 2D monitor.

The investigation tries to be as general as possible since this particular display system is just one of many autostereoscopic display systems and a similar system will in the future have higher resultion and use more computing power for the rendering. It is therefore interesting to develop tests in which the display concept can be tested, and not the current technical limitations.

If the light field display can show images without noticeable distortion, perform comparable or better with a other display environment for user tasks and display images in high enough quality to be acceptable in medical environment has it been showed that the light field display is a viable option as a medical display system.

1.2.1 Volume rendering

The first objective is to explain how the display works and how to develop a volume rendering application for the display. The concept of real time volume rendering will be explained thoroughly as volume rendering is an important part of the system. Different techniques and the performance of the display using different techniques will be examined. The display should be able to handle high quality volume rendering when been driven by a single PC in frame rates high enough to provide interactive manipulation of the scene.

1.2.2 The displays limitations

The display system construction sets certain limitations on the viewing angle, resolution and other similar variables which will affect all images displayed on the screen. The display has a different depth resolution than a regular display, since the viewer experiences an image with depth. The minimum depth value representable is therefore an important variable to determine, since it shows the minimum dimensions of small objects that can be rendered. The display uses a complex non-linear projection computed with a fixed viewer position to display images in a limited field of view. Viewer head tracking can be used, but this increases the complexity of the display software and diminishes its usefulness for multiple users. A user outside the optimal viewing position should be able to understand the volume layout, as long as they are within the field of view of the display. Therefore, the error in the projection must be small enough to not degrade image quality. Furthermore, the projection causes other artifacts in certain situations, which will be delt with later on in this thesis. These distortions will be calculated from the projection calculations to give an analytical description and examined empirical to make an analysis of the effect.

1.2.3 Comparison with other displays

An important factor is a comparison with other displays using volume-rendered data as the light field display should serve a replacement for these displays. The light field display should make it simpler and faster to perform tasks with volume rendered images than in traditional systems. This can be tested by performing the same task in different environments and analyzing the results with an established method. The important result in this case is the correctness, since the correctness is crucial to a medical analysis, but the speed of solving the tasks will be important to show how easily the user can understand the rendered images. This display has the possibility to provide certain depth cues which 2D or stereo systems are unable to show. It's therefore important to know which extra cues will be available and

their impact on human vision. It can be proven that these cues are available by empirical analysis of rendered images. Several tests must therefore be constructed to verify different cues and create tasks for the user to perform, which must result in analyzable data to show if the light field display has an advantage over other systems.

1.2.4 Medical display assessment

The last objective of this thesis is a comparison between this light field display and a two dimensional medical display systems, which show X-ray images with high resolution and high contrast. The medical volume rendered by the light field display should also have a high quality to be usable in a clinical environment. It must therefore be proven that the display's resolution, brightness and contrast are sufficient for a medical display system. This can be done by following established standards used for medical display system evaluation and analyzing whether the result is acceptable for a medical display.

1.3 Thesis layout

This thesis is made of the following chapters arranged in the following order.

Background explains the basic functions of the light field display and the concept of volume rendering. This chapter also contains some important factors of human depth perception and the requirements of medical displays which gives a reason for the perception test being done with the display.

Light field display explains which hardware and software is used for driving the display and also which calculations must be applied for the projection. This chapter includes the distortions which occurs when the real world viewer doesn't correspond to the viewer position used in the projection calculations. The chapter contains descriptions of some tests and also how determine if these distortions is within acceptable limits.

Volume rendering explains how volume rendering techniques can be applied on the light field display and also describes some implementations and the resulting performance.

Perceptual evaluation contains the description and the results regarding depth perception on the light field display.

Evaluation in a clinical context contains a description of a test made with the display using tools for medical display evaluation. The chapter also contains a description of a test which uses the real data sets.

Discussion contains ideas for future work.

Chapter 2

Background

2.1 Introduction

This work focuses mainly on three areas of research, volume rendering, autostereoscopic display systems and human depth perception; and the area of medical imaging devices is only partially considered in this study. A lot of work has been done in each area, even in the relative new area of autostereoscopic display systems and real time volume rendering. Therefore, this chapter will give examples of different autostereoscopic and volume displays, the basics and current research in volume rendering, a summary of the important cues humans use for depth perception and a short description of the established requirements on medical display systems.

2.2 Autosterescopic Display system

A common way of displaying a three-dimensional image for a viewer is to show a scene from two different perspectives, one for each eye to mimic the way which human vision works. This can be done in a number of different ways (some concepts are described by Dogson in [7]). Most commonly by showing two different images of the scene rendered from slightly different viewpoints and present the left and right perspective to the viewer's left and right eye. This is commonly done by forcing the viewer to wear glasses which filter out the left image from the right eye and the right from the left eye. The image is then displayed by mounting polarizing, or analglyph (color), filters to make sure that each image is seen only by the appropriate eye. Another possibility to create the illusion of a three dimensional image is LCD shutter glasses (see image 2.1). Glass containing liquid crystal and a polarizing filter has the property that it becomes dark when voltage is applied, but otherwise is transparent. A pair of eyeglasses can be made using this material and connected to a computer video card. The video card alternately dark-

ens over one eye, and then the other, in synchronization with the refresh rate of the monitor, while the monitor alternately displays different perspectives for each eye. At sufficiently high refresh rates, the viewer's visual system does not notice the flickering, each eye receives a different image, and the effect is achieved.

The need for special glasses in these kinds of displays makes it uncomfortable for viewers while the polarized filter absorbs half of all incoming light, making a stereo display dimmer than its non-stereo counterpart. An autostereoscopic dis-

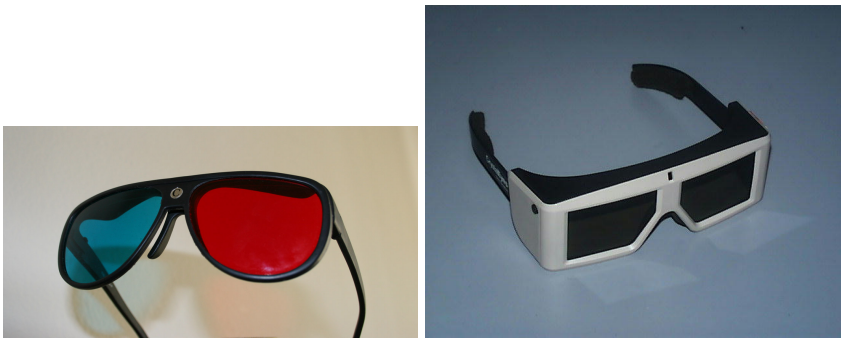

Figure 2.1: **Left image:** Glasses with anaglyph filters. **Right image:** LCD shutter glasses.

play is a type of display which can at least provide a stereo view similar to the stereo glasses, while some can give more cues which enhance the 3D experience even more. The simplest autostereoscopic displays uses two views projected onto a screen which is covered by a parallax barrier in such a way that a user in the correct position see the right view in the right eye and the left in the left eye (see image 2.2). The illusion of 3D is only perfectly perceived from a specific position or zone in front of the screen. A solution to this is to keep track of the users position and adapt the viewing zones accordingly. The tracking can be made with head tracking systems carried by the user or eye tracking system which follows the user's gaze. Unfortunately such a display is useless for multi user environment since the screen only can adapt to one user, and the scene is still only shown through two 2D displays. Although the scene is perceived from a real 3D space will the user still focus his eyes on the surface of the two dimensional screen. As described in the paper by Wann[25] this inconstancy is tiring since our brains have adapted to the real three dimensional world. A prolonged exposure to the contradictory vision cues can cause headaches and sickness. Light field displays are based on a different principle because they are able to project onto the screen a real 3D model. The viewer can explore it visually just like in the real world. The principle is different from the stereo system since these kinds of display will, optimally, use the light field of the whole scene and not only two different perspectives. This can be achieved by capturing multiple perspectives from the scene and

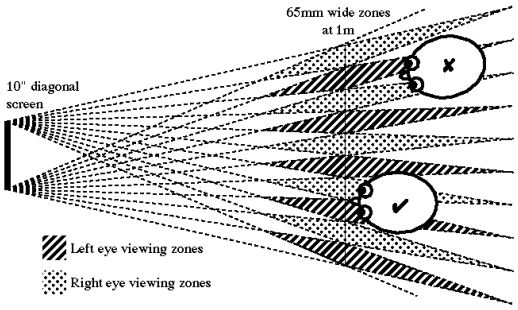

Figure 2.2: **Image**:Parallax barrier providing viewing zones for the left and right eye

displaying these views in such a way that a human watching the scene perceives an object which can be viewed from multiple directions. The multiple views provide a huge challenge in the display construction which has sparked the inventions of many different technologies.

2.2.1 Spatial multiplex

These are similar to a stereo display using a parallax barrier (see image 2.2) but use lenses instead to separate the image into four or more viewing zones. The system can thereby provide more perspectives of the scene. But the resolution is divided by the number of views, as each pixel on the lcd display only can show one color, limiting both the number of views and the resolution of the resulting image. The viewers are forced to view the scene in fixed angles similar to an autostereoscopic lenticular display. Spatial multiplex systems exists in many retail version from many different manufactures.

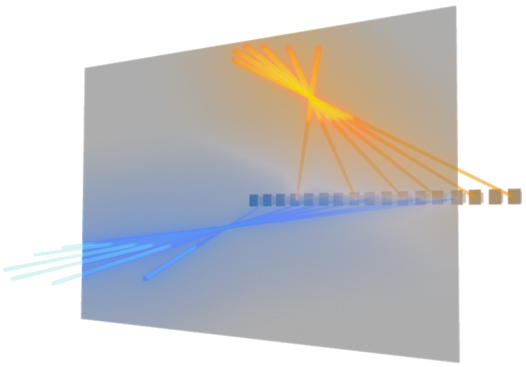

Figure 2.3: **Image**:Multi projector system

2.2.2 Multi projector

A multi projector system simulates the light field of the scene by projecting different views using a set of projectors (see figure 2.3). They form an discrete approximation of the continuous light field which would represent the scene in a real world. The different views are projected onto a screen which blends the light appropriately to give the viewer the experience that they are watching a real-world object. These displays offer far more views than an spatial multiplex as an increasing number of projectors doesn't limit the resolution and gives a better experience of depth. However, a large number of projectors is difficult to be aligned in order to give a correct blended view. Many systems use lenticular screens that magnify different projected images from different viewpoints. The small magnifying lenses on the screen surface create a visible pattern as light is bent from the different projectors. The multi display system used in this thesis has a holographically recorded screen that smoothly bends incoming light and thereby avoid any visible borders.

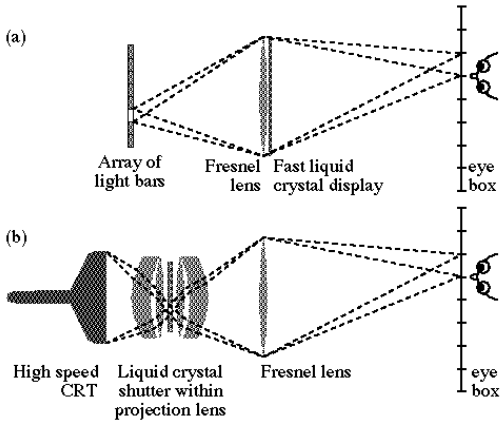

Figure 2.4: **Top**: The principle of a time sequential display. **Bottom**:The Cambridge display system, a time sequential display system.

2.2.3 Time-multiplexed displays

This devices use a single display with high refresh rate projecting the different views while an optical component quickly change to project the displayed view in the appropriate position. An example of this is the Cambridge display [8] which uses an very fast CRT display and a fast liquid crystal element, called shutter. The image from the display is focused with lenses onto the shutter element, this light is then projected onto a Fresnel lens which projects an image of the shutter visible

to the viewer. Dark areas can appear between the different strips which makes the image, as the shutters different strips can be seen, and the sequential displays require a very high speed CRT projector which must be perfectly in sync with the shutter element. The Cambridge display has been developed in a color version using three crt projectors with different filters.

2.2.4 Pure holographic images

The hologram was invented in the 1948 by Dennis Gabor and has been used for holographic still images, often as a copy prevention tool on identity cards and products prone to piracy. A hologram is recorded with both amplitude and the phase of the light wave by interfering the light front from the object with a reference lightfront, usually a laser. The interference pattern is recorded in a special photosensitive material to form the holographic image. The developed image diffracts incoming light by interference fringes on the surface and thereby reconstructs the wavefront of the original object. The holographic image displays a continues light field of the scene which was captured, a capability which would be the goal of display technology. Holographic display systems has been developed, although yet in a experimental state displaying small and grainy images [14][23].

2.2.5 Real Volume displays

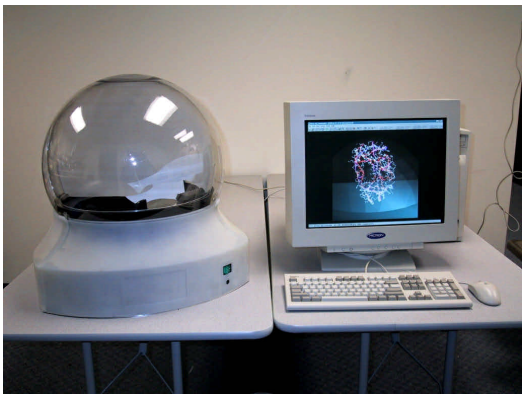

Figure 2.5: **Image**:A volume display developed by Actuality Systems using a small rotating screen.

These kinds of displays differ from the other in that they create a volume of light in the air, which can be looked at in every direction. These displays are so far mostly experimental showing very small translucent images compared to

the size of the display. Devices has been constructed with the use of a rotating mirror combined with a fast projector which projects the images synched with the rotation [16][11]. This design limits the size of the display as the mirror's weight create a physical limit for fast and high precision rotation.

2.2.6 This system

This particular display (presented in [4]) is a Multi Display system, developed by the hungarian company Holografika, which uses an arranged array of micro-display projectors that project light on a special screen. Each projector emits emits light beams toward a subset of the points of the holographic screen. The special holographic surface on the screen enables a directional selective transmission of light beams which result in a smooth blend of the different projected views for the viewer. Similar multi display systems has been constructed in different forms, but the holographically recorded screen creates a fully continuously blend of the views.

The common factor of autostereoscopic display systems is the complexity and cost. They also requires much computing power and specialized software to function. The increased complexity and cost forces the displays performance to be very good to make them a reasonable option to established 2d or stereo systems.

2.3 Volume rendering

The projection of a three dimensional datasets onto a two dimensional display provides a challenge. A two dimensional dataset can easily be mapped to a two dimensional surface, such as a polygon. which makes them easy to work with since the graphics hardware in computers today are constructed to quickly work with points in three dimensions which forms two-dimensional polygons. The points is quickly projected onto the display surface with the use of hardware implemented matrix and vector multiplications. With a dataset spanning a third dimension will the regular projection not work. One dimension have to be discarded during the projection, resulting in a great deal of data loss. The methods for rendering of three dimensional datasets has therefore often been forced to invent new way of using the graphics hardware, stepping outside of the normal pipeline.

2.3.1 Volume acquisition

Magnetic resonance imaging (MRI) uses a powerful magnetic field which cause nucleuses, mostly hydrogen, with a magnetic resonance to algin with the field.

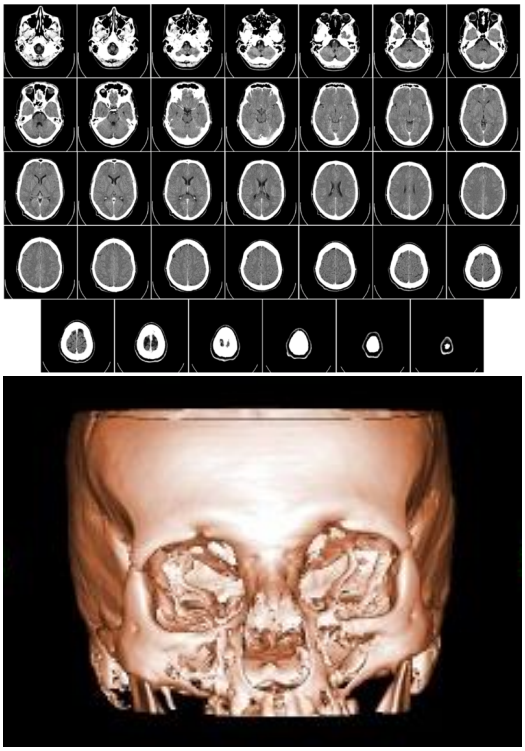

Figure 2.6: **Top**: Slices of ct scanned data of a head from Uppsala University in Sweden. **Bottom**: A volume rendering of the same data.

When a electromagnetic wave of the correct frequency hits the particle will it resonate and release energy which can be measured and reconstructed into images by focus of the magnetic field.

Computed tomography uses series of two dimensional X-ray images by rotating a detector and emitter around the body which is examined. These image slices can then be combined into a three dimensional dataset. The X-rays are absorbed by denser material in the body, but isn't useful to distinguished different kinds of tissue. CT scanning is more common since the equipment is less expensive and faster than MRI.

2.3.2 Sampling methods

The visualization of three-dimensional data is an old problem for computer visualization as multi-dimensional data sets are common. The basic concept is simple: try to translate the volume dataset into two-dimensional polygons which can easily be handled by a regular rendering pipeline. It's possible to generate a polygonal

Figure 2.7: **Image**:An MRI scanner

model from the iso surface of the data set using methods such as the marching cube algorithms or other similar techniques. An other method commonly used is to sample the volume by slicing through it with two-dimensional polygonal planes which then will contain two dimensional samples from the volume. The most natural approach is to follow the light rays trough the volume and calculating the volume's impact, absorbtion and refraction, on the ray as it passes (a technique described by Max [18]). This technique is called direct volume rendering since it uses the volume directly during rendering, unlike the previously described techniques. The simplest way to render the final image is to use a ray casting algorithm. A ray is generated for each desired image pixel. Using a simple camera model, the ray starts at the center of the projection of the camera and passes through the image pixel on the imaginary image plane floating in between the camera and the volume to be rendered. The ray is clipped by the boundaries of the volume in order to save time. Then the ray is sampled at regular intervals throughout the volume. The data is interpolated at each sample point, the transfer function applied to form an RGBA sample, the sample is composited onto the accumulated RGBA of the ray, and the process repeated until the ray exits the volume. This method requires the execution of long loops and is heavily influenced by the size of the render target since one ray is usually cast per pixel. Methods for volume rendering, both using polygonal slices and direct volume rendering is described in the book by Engels et al [9].

2.3.3 Real time volume rendering

The increasing power of computers made it possible to implement real time volume rendering systems. In the beginning on custom made hardware but with the increased development of cheap consumer focused 3D graphics, driven by the in-

creasing interest and profit in computer games, have availability and power of 3D graphics hardware evolved the possibilities of real time volume rendering.

An enormous amount of opportunities opened when graphics cards arrived where the graphics processor unit (GPU) could execute small pieces of custom written software to calculate the position of vertices and the final color of pixel on the screen. The first step toward a programmable GPU was the introduction in 1999 of a configurable rasterization, determining the color of the pixel which will be sent to the framebuffer, and vertex processing, the transformation and projection of the vertices. The configurable architecture was not easy accessible since each vendor used very specific features. It was for example not possible to execute the loops which is required for the direct volume rendering algorithm. Loops and conditional breaks within the loop is functions which a regular CPU easily can handle.

The breakthrough came when the fully programmable GPUs were introduced. Custom code could then be executed, called shader programs, directly by the vertex processor and the fragment processor. Some high level shader languages has been developed to make it easy for development of shader programs using a c-like language. The current languages used are the GLSL language accompanying the OpenGL 2.0 language specification, Nvidia Cg, which is a derivation of the Stanford Shading Language, and HLSL which was introduced in Microsoft DirectX 9.0 sdk and uses a Cg like syntax.

The latest generations of high performance GPUs are able to provide the flexibility and computing power which a ray caster requires (an example is described by Stegmaier [24]) for real time rendering. A source of the current state of real time volume rendering can be found in the book by Engel et al [9]. Since the graphics boards in standard PCs are getting better and more flexible will ray casting be the optimal method because of its highly adaptive implementation.

2.3.4 Volume rendering on multi view system

The implementation of a volume rendering application on a multi view system require powerful hardware since volume rendering is a computationally heavy technique compared to regular polygon rendering. If the display cannot be driven by a single PC will the complexity of a the system limits it's attractiveness compared to simple two-dimensional single PC applications. Research has been done in the specific area of multi projection volume rendering since the single view rendering doesn't take into account the similarities and dependencies between different views in a multi display system [13][12]. Since both multi display system and real time volume rendering is new technologies will more methods be developed, probably requiring new views on the rendering pipeline. The volume is most easily expressed as a volume, not as a projection of two-dimensional images.

With increasing amount of data from CT and MR scanners will development of better visual understanding of the data help physicians to make faster and more reliable diagnosis.

2.4 Human depth perception

Humans perceive the world trough two two-dimensional projections from two different perspectives. Despite this lackluster information can we relatively easy and accurate determine distance and understand the layout of the three dimensional world we perceive. But the human depth perception can not be generalized as a combination of two different view ports of a scene. It's a complex and still not well understood area. The reason for this is that humans uses many different cues from the current perceived scene but also previous knowledge of the world and the objects therein. Humans also uses different cues for different distances. The difference between the images from the left and right eye becomes smaller at greater distances forcing us to depend on other cues. In [15] is James describing which of these cues is to most important and viable to do research on and also which is most useful for different distances. The light field projected on the display is made to be watched at approximately 0.5 meter and will only display a scene with by one volumetric object. The volume will in itself contain many distinct features but will not exist in a context with measurable distance cues such as a landscape or rulers. Because of the limited scene will only a few of the cues be important.

2.4.1 Occlusion

This is simply the effect that reflected light from one object is interrupted by other objects between it and the observer. The effect is implemented in computer graphics by use of z-ordering. If two objects overlap an area on the screen will the pixels from with the lowest z-value, the pixel closes to the camera, be displayed.

2.4.2 Retinal size, Density

Objects closer to the observer will project onto a larger part of the retina than objects further away which implies that volumes closer to the viewer appear less dense then object in the back since the separate particles will be seen further apart. The effect is mimicked in computer graphics by projecting the vertices using a perspective projection matrix.

2.4.3 Motion Parallax

This is the difference of motion between objects in the world as an observer moves through it. Objects further away has a lower speed than objects closer to the observer. This effect can be achieved by a stereo display if the position of the viewers head is tracked and used to adapt the virtual camera position used for rendering. A multiview display allays provide a parallax effect.

2.4.4 Aerial perspective

This is a small but distinct effect that depends of water and particles in the air which interfere with the light from objects far away. Distant mountains can be seen with a foggy appearance. This effect can easily be simulated in virtual environments by simply decreasing the saturation of objects far from the observer. Kersten[17] has shown that an simulated aerial perspective can increase depth perception for a volumetric rendered scene. Although it will decrease the detail of objects far away from the observer.

2.4.5 Accommodation and convergence

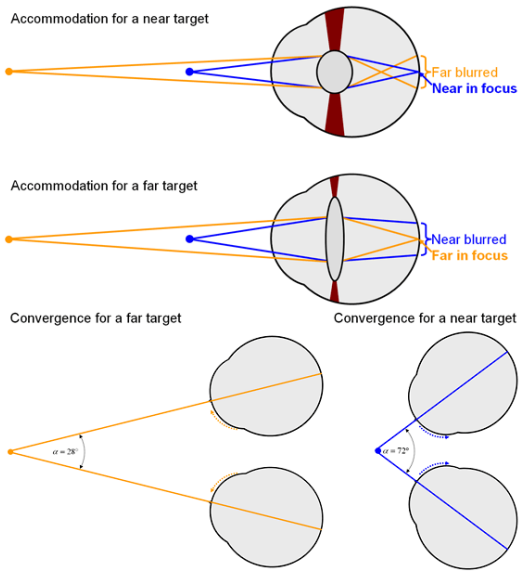

Figure 2.8: **Top**:Accomodation **Bottom**:Convergence

The important difference from an stereo-display system and a volumetric display is the addition of the convergence and accommodation cue. Convergence is the amount each eye need to turn in order to put the object in center of the view

[21], while accommodation is the shape changing capabilities of the lens to keep the object in focus. In a stereo display system will the different projected images not correspond to different positions in the world, since both eyes are looking at the same screen. The multi view display used here reflects the light beams from the projector in such a way that they is perceived to come from an object located in space. [4] But the pixel is still projected upon the surface on the display, which will make the testing of accommodation and convergence interesting.

2.5 Medical imaging devices

Figure 2.9: **Image**:Medical two-dimensional datasets

A typical medical image device shows two-dimensional images, often monochrome xray images, for analysis by physicians. As the light field display used in this thesis will try to replace a medical display should it also be able to display an two-dimensional xray image without quality loss. It is necessary for the display to fulfill requirements for medical displays to be able to use the display in a real clinical situation. A medical display need to have high contrast and resolution to make to probability of missing an important anomaly as low as possible. The American Association of Physicists in Medicine (AAPM) have written an comprehensive assessment for medical display systems which will be used to asses the capabilities of the light field display. AAPM consists of medical imaging experts and organizational affiliates dealing with performance evaluation of electronic display devices, whose purpose is to generate a document that provides guidelines to practicing medical physicists and engineers for performance evaluation of electronic display devices intended for medical use. The assessment is composed of several test patterns used to test different characteristics such as resolution, contrast or luminance. The assessment is described in a long and intricate description

of test to be carried out using equipment for light measurement [22]. Most of the test have a quantitative evaluation using a specific pattern, special equipment and measurement, and a visual evaluation by simply looking at a pattern and determine if some parts of it are displayed as they should be according to some requirements.

Chapter 3

Light field display

3.1 Introduction

A volumetric display increase the complexity and the computational load severely since each scene will be made of several images, often using a non linear projection. The graphics hardware is designed to project three dimensional points onto a two dimensional plane quickly by multiplication of the vertices with a projection matrix. In a regular renderer is the viewer located in the same direction as the camera, and the scene is projected upon a surface in front of the viewer which is then displayed on the monitor. The light field display uses several projectors located in different positions and a screen surface which bends light in different directions. The different projectors have slight differences in light output, color reproduction and pixel alignment. The special holographic material of the display surface cause horizontal light waves to travel straight trough the display, while the vertical rays will scatter in every direction. This attributes presents a huge challenge for rendering to the display and the regular pipeline used by the graphics card have to be adapted in several steps.

To display one image on one projector must the renderer know where the projector is located, where the viewer is located and what attributes the projector have which affect the colors of the output pixels. This images must then be sent in combination with the images for the other projectors to form the light field which the viewer sees. The projection of vertices to the position on the projector screen is non linear because of the screens light-bending ability. This kind of projection can cause problem since the graphics hardware is made for linear projections. Such errors must be investigated to show is they decrease the image quality of the display in a sever way.

This chapter describes how the light field display works and includes all calculations for the projection used and describes how the pipeline is implemented.

The last part of the chapter explore the risks of visual artifacts due to the necessity of a non linear projection and determine if these artifacts has an important impact on image quality.

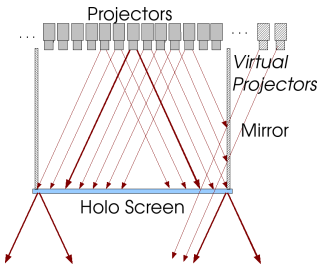

Figure 3.1: **Image**:The light transfer in the horizontal direction

3.2 Concept

The light field display [4][3] uses an arranged array of 96 micro-display projectors which projects light on an special screen. Each projector emits light toward a subset of points (320x240 pixels) on the screen and each point receives light from several projectors. The special holographic surface enables a directional selective transmission of light beams which result in a smooth blend of the different projected views. The surface provides controlled angular light distribution which is precisely set in accordance to screen geometry. The light from the projectors is scattered widely in the vertical direction, making the image possible to view from any vertical position, while transmitting light sharply in the horizontal direction (see image 3.1). The horizontal transmission is designed as wide plateaus and steep Gaussian slopes overlapping in narrow regions which provides a homogenous light distribution resulting in a smooth blend of incoming light.

3.3 Projection

The method for projection is described in [3] and works as follows. Each projector is located at a fixed position $\mathbf{E} = (Ex, Ey, Ez)$ and projects the image on a rectangular portion $(R_x^+, R_x^-, R_y^+, R_y^-)$ on the screen plane at $z = 0$. The screen transmit light selectively in the vertical direction while scatter it widely in the horizontal direction. Because of this ability must special steps be taken to project a point to the correct position. The light in the x axis from the projectors is transferred in a direction depending on the incoming angel (see image 3.2). The

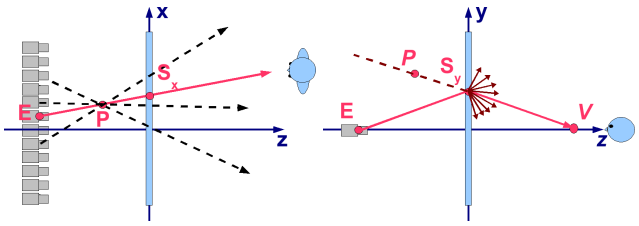

Figure 3.2: **Image**:Light field projection

projection on the x-axis is calculated as the intersection between the ray from the emitter position going through the point at the screen plane located at $z = 0$.

$$Ray : \begin{bmatrix} e_x \\ e_z \end{bmatrix} + t * \begin{bmatrix} e_x - p_x \\ e_z - p_z \end{bmatrix} \tag{3.1}$$

$$Plane : \mathbf{P} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0 \tag{3.2}$$

Substituting the P with the ray

$$\left(\begin{bmatrix} e_x \\ e_z \end{bmatrix} + t \begin{bmatrix} e_x - p_x \\ e_z - p_z \end{bmatrix} \right) \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0$$

$$t = -\frac{\begin{bmatrix} e_x \\ e_z \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}}{\begin{bmatrix} e_x - p_x \\ e_z - p_z \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}}$$

$$t = -\frac{e_z}{e_z - p_z}$$

Inserting t in the ray equation:

$$\mathbf{Px} = \begin{bmatrix} e_x \\ e_z \end{bmatrix} - \frac{e_z}{e_z - p_z} \begin{bmatrix} e_x - p_x \\ e_z - p_z \end{bmatrix}$$

$$\mathbf{Px} = \begin{bmatrix} e_x \\ e_z \end{bmatrix} - e_z \begin{bmatrix} \frac{e_x - p_x}{e_z - p_z} \\ 1 \end{bmatrix} \tag{3.3}$$

It is necessary to know the viewers height and distance from the display to calculate the projected y position since the screen scatters light uniformly in the horizontal direction (see image 3.2). The screen position in the y-axis will be determined by the plane-ray intersection between the viewer position and the screen. Which gives a similar expression to the x position, but exchanging the projector with the viewer:

$$\mathbf{Py} = \begin{bmatrix} v_y \\ v_z \end{bmatrix} - v_z \begin{bmatrix} \frac{v_y - p_y}{v_z - p_z} \\ 1 \end{bmatrix} \tag{3.4}$$

These calculations results in a position **P** on the screen (since both equation results in the z position 0). But further calculations are needed to project the coordinates in the image rectangle of the projector. A orthographic projection matrix **S** is applied to map the x and y coordinate to the image rectangle of the projector.

$$\mathbf{S} = \begin{bmatrix} \frac{2}{R_x^+ - R_x^-} & 0 & 0 & -\frac{R_x^+ + R_x^-}{R_x^+ - R_x^-} \\ 0 & \frac{2}{R_y^+ - R_y^-} & 0 & -\frac{R_y^+ + R_y^-}{R_y^+ - R_y^-} \\ 0 & 0 & -\frac{2}{far-near} & -\frac{far-near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

The previous computed point **P** on the screen is subtracted with the emitter position **E** to get the relative position, and multiplied by the projection matrix to form the projected point **H**. The projection will not use the far or $near$ clip planes as the coordinates P and E already are located at the plane $z = 0$.

$$\begin{aligned} \mathbf{H} &= \mathbf{S} * (\mathbf{P} - \mathbf{E}) \\ \mathbf{H} &= \begin{bmatrix} \frac{(Pn_x - E_x) * 2 - (R_x^+ - R_x^-)}{(R_x^+ - R_x^-)} \\ \frac{(Pn_y - E_y) * 2 - (R_y^+ - R_y^-)}{(R_y^+ - R_y^-)} \\ 0 \\ 1 \end{bmatrix} \end{aligned} \tag{3.6}$$

The z position will be needed by OpenGL to use object order culling and transparency during rendering. Since the depth from the current projection is zero must a new depth be calculated with an object order counted from the viewers point of view instead of the projector. The depth is negated and divided by the viewer to form a value between -1 and 1 with values larger near the viewer and smaller further away (see figure 3.2).

$$H_z = -\frac{P_z}{V_z} \tag{3.7}$$

3.4 Pipeline

The display is fed by a DVI channel working at 75hz with a resolution of $1280 * 1024$ pixels. Each frame is made up of 16 $320x240$ images for 16 projectors which id's are color coded in an header at the top of the image (see image 3.3). Since the display is made up of 96 projectors will it need 6 of these framelets to show one complete 3D image. Rendering a total of $1280 * 1024 * 6$ pixels per scene.

Each of the frameslets start a loop through its associated projectors and renders a scene using the current projector's attributes onto a frame buffer object. This makes it possible to use most regular shader and OpenGL calls and the rendering

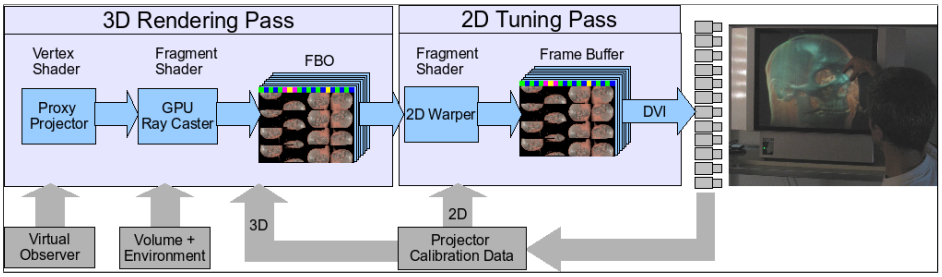

Figure 3.3: **Image**:Display pipeline

loop of a normal OpenGL scene can easily be extended to work with this system by using a vertex shader which can implement the special transform calculations instead of OpenGL's projection matrix. Any fragment shader can then be used with the projected vertices to calculate the colors for the pixels. When 16 scenes has been rendered is the frame rendered to the frame buffer using a fragment shader to add the color coded ids and correct the colors and slight geometrical distortions for each projector according to a pre computed per-pixel lookup table. The full 3d image is then created by the display using the scenes in the frame buffer.

Figure 3.4: **Image**:A volume rendered to the display. The structure of the surface which bends the light can be seen in the closeup.

3.5 Projection distortions

The screen construction and the special rendering pipeline will create new circumstances which must be looked at. The non linear projection is not compatible with the linear computations in the graphics hardware which cause slight errors when relying on functions such as interpolation. The projection is using a fixed viewer position for calculations, which gives viewer outside of the optimal posi-

tion a slightly distorted image. These kinds of limitations will now be examined to see if they cause any fatal errors in the displayed image.

3.5.1 Depth dependent resolution

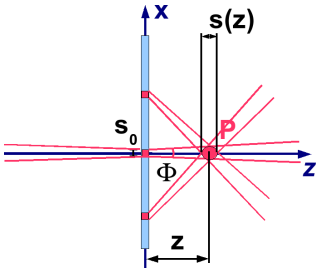

Figure 3.5: **Image**:Point size

The resolution of the image viewed by the observer will not be the same as the resolution on the surface of the screen, since every light beam leaving the screen has a finite angle Φ and every point is made of several crossing light beams. The smallest point size displayable will, because of this factor, depend on the distance of the point to the surface of the screen. It's important to know this relationship during rendering to display the image inside the field of depth of the display, i e the maximum distance at which objects are faithfully reconstructed [3]. This relationship can be seen in the image 3.5 and described as:

$$s(z) = s_0 + 2\,\|z\|\tan\left(\frac{\Phi}{2}\right) \tag{3.8}$$

Where s_0 is the point size on the screen surface and Φ is the angular spread of the screen in the horizontal direction. This screen has a point size of 1.25 mm and an angular spread of 0.8 degrees.

3.5.2 Non linear projection

The display cause long lines between vertices to appear curved since the projection is non linear while the hardware interpolation between the vertices attributes is linear in the rendering pipeline. This problem can be solved by tessellation of very sparse models but should not be a problem in most circumstances. The addition of a few vertices will never affect performance severely but will require additional pre processing of the geometry. The programs used in this thesis uses models generated during program execution and not loaded preconstructed models. Therefore can a simple loop be included in the modeling code whenever two

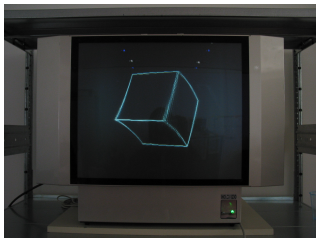

Figure 3.6: **Image**:Projection error on a wireframed rendering off a cube. The lines is perceived bent.

```
float sx = (vertex2[0]-vertex[0]) / nr_of_segments;
float sy = (vertex2[1]-vertex[1]) / nr_of_segments;
float sz = (vertex2[2]-vertex[2]) / nr_of_segments;
float x = vertex[0];
float y = vertex[1];
float z = vertex[2];
glBegin(GL_LINE_STRIP);
for(int i = 0; i<grid+1; i++) {
  glVertex3f(x,y,z);
  x += sx;
  y += sy;
  z += sz;
}
glEnd();
```

Figure 3.7: Division of a line between two vertices.

vertices is placed, dividing the line in a number of segments. It might be possible to calculate the error for a segment length and use a threshold to determine when a segment should be split, but this will depend on the final projection of the vertices making it necessary to compute the length each frame. It is more effective to calculate the max length of a segment projected on screen when the segment is closest to the viewer.

3.5.3 Depth of field

The screen surface has a physical limit, in the horizontal direction, of how close a viewer can watch the screen and still see light from all projectors. This is dependent on the width of the screen and the spreading θ angle of screen surface. $N = \frac{L}{2}\cot(\frac{\theta}{2})$ The closest viewing distance is $536.4mm$ for this screen, where $L = 500mm$ and $\theta = 50$. The point size is dependent on the distance to the screen surface as explained above. It is possible to calculate boundaries, height and depth, for a scene using a desired point size to preserve resolution. This is done by calculating the depth of the desired size using trigonometry and then use

Figure 3.8: **Image**:Out of range projection

the closest viewing distance N and the screens height (400 mm).

$$D = \frac{s_z - s_0}{\Phi}$$
$$H = \frac{(N-D)ScreenHeight}{N}$$

The limited field of view will prevent the display to handle clipping as well as a regular displays since the projectors display a volume in space and not a projection from the viewers position as in a regular single or stereo rendering. The projection causes depth (z) values below or above certain values to be rendered outside the screen surface depending on the position of the emitting projector, causing a inconsistency in the image as some projectors no longer contribute to the final image since their projection is outside the screen. The position of a point in the y direction is calculated with the vertex y value and z value, the viewer y and z value. While the x position is dependent of the emitter position as well as the z value of the vertex. This creates a complex border to determine which points is inside the viewing volume. If the border is known can the rendered volume be adapted accordingly to prevent breaks in the images. Points located behind the projectors is of course also impossible to display, making the display only usable for small volumes. It's therefore necessary to adopt the volume to fit inside the limited volume of the display. The volume of the scene should be tested to see if some of it's vertices gets projected of-screen and then take appropriate action. A solution is to make the program prevent rendering of a vertices outside the limits, or clamping every projected vertex inside the depth of field. Or simply prevent transformations which project the image of-screen.

A framework for resampling and antialiasing for multiview displays has been proposed by Zwicker[26]. The method transforms the scene to fit inside the display using complex filter and signal processing.

3.5.4 Distortion

The viewers position in the y and z direction is a fixed value in the projection calculation, even though the display encourage the viewer to move around and look at the scene from several direction. This will result in a slight distortion of the displayed image when the viewer is away from the position used during the projection calculations. The distortion is only present in the horizontal direction, the screens y coordinate, as the x coordinate depend on the projector positions. A large distortion will increase the risk of misinterpretation of the projected image, since the viewer will be changing position to be able to observe the scene from different viewpoints. The projection calculate the point on the screen for a viewer $\hat{V}$ as:

$$\hat{S_y} = \hat{V_y} - \hat{V_z} \cdot \frac{\hat{V_y} - P_y}{\hat{V_z} - P_z} \tag{3.9}$$

While the real viewer is located at position V. Which makes the screen position error, if the fixed viewers y position is 0:

$$\begin{aligned} \Delta S_y &= \|\hat{S_y} - S_y\| \\ &= \|(-V_y + \frac{V_z}{V_z - P_z}(V_y - P_y) + \frac{\hat{V_z} P_y}{\hat{V_z} - P_z}\| \end{aligned} \tag{3.10}$$

The distortion can be calculated as an angle ϵ_θ which depend of the screen position error ΔS_y ad the viewers distance from the screen.

$$\epsilon_\theta = \arctan(\frac{\Delta S_y}{V_z}) \tag{3.11}$$

It is impossible to evaluate all x and y coordinates for calculating ΔS_y, but a limitation can be set since the screen has a limited displayable area, as explained in the previous segment. The x and y coordinates are only interesting in their extreme positions as these are the points furthest from the viewer and will have most influence on the error. These positions can be calculated for a desired spatial resolution as described in the previous segment. With $P_y = H$ and $P_z = D$ can the maximum angular distortion for a viewers position be calculated.

$$\begin{aligned} \Delta S_y &= \|\hat{S_y} - S_y\| \\ &= \|(-V_y + \frac{V_z}{V_z - D}(V_y - H) + \frac{\hat{V_z} H}{\hat{V_z} - D}\| \end{aligned} \tag{3.12}$$

A plot of the distortion angle, made with Octave [1],using a fix viewer distance of $y = 0$ and $z = 1000mm$ and a desired point size of 2.5 mm can be seen in 3.10.

[1] http://www.gnu.org/software/octave/

The viewers z position, in mm, is represented on the x axis while y axis represent the viewers y position. The z axis represent the distortion angle, regardless of direction. This plot show that the distortion for large z-distances settling down at 5 m while the distortion is very big for distances smaller than the fixed viewer position. The distortion for changes in the y position is large for viewers close to the display, but big z values even out the differences.

It is difficult to know how this distortion will be perceived by the viewer. A visual test is therefore carried out to see wether the distortion will be of a noticeable magnitude.

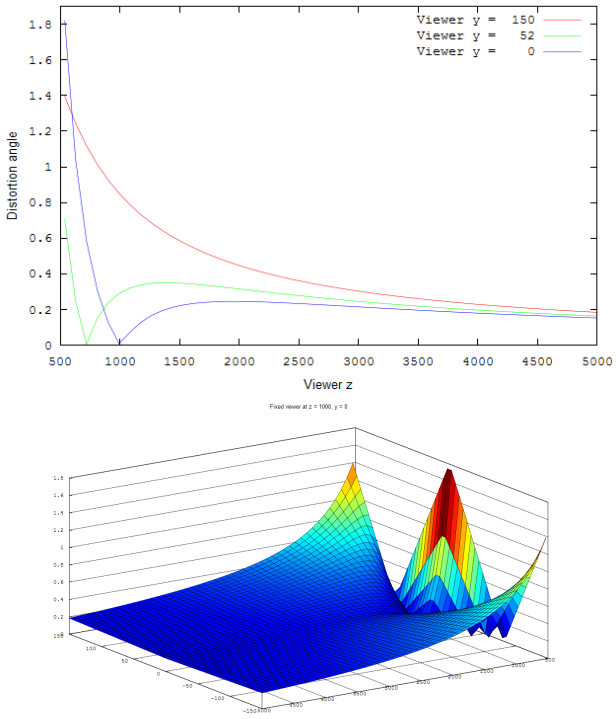

Figure 3.9: **Top**:Graph of the calculated distortion depending on three different viewer distance. **Bottom** Plot of the distortion angle depending on the viewers z and y position.

3.5.5 Test program

This program is used to show the distortion caused by the fixed viewer position in the projection calculations. The program renders the front sides of two wire-framed cubes to the light field display. One of the cubes, rendered in white, use a constant viewer parameter for the projection while the other, rendered in yellow,

uses a variable value which can be changed trough a graphical interface. The cubes are rendered as lines drawn between vertices to clearly show how their borders are projected. This setup enables a clear view of the distortion which the fixed viewer position gives by observing the difference between the yellow and white lines. The lines in the cubes are divided by many vertices and rendered as several OpenGL lines to make them look straight on the display, as explained above in the section about non linear projection. The distortion in the z axis make the object narrower or wider as the vertices are moved out or in from the center. The distortion for the y axis results in a translation of the object as the vertices adapt for a viewer watching the screen from a higher or lower position.

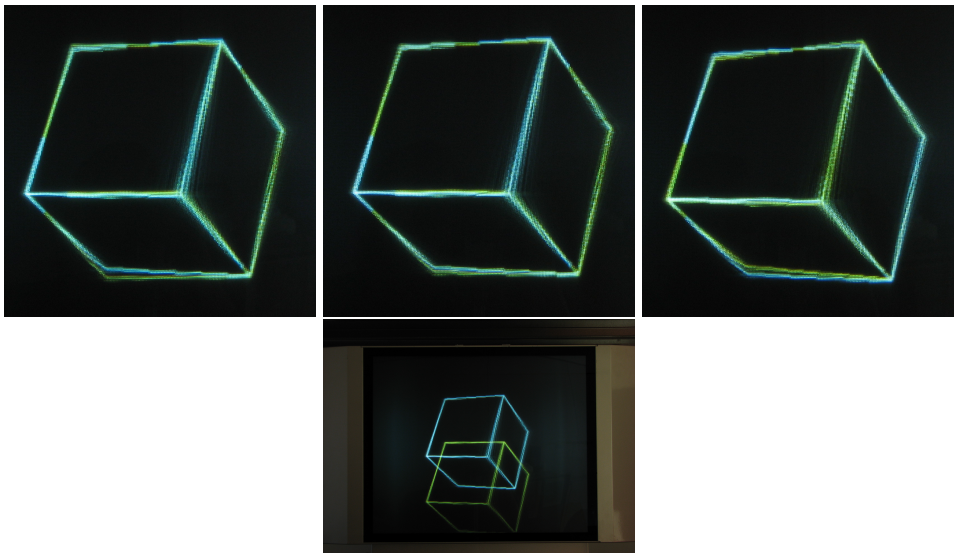

Figure 3.10: **The first three**:Distortion display test. White cube is rendered with a viewer at 1m z distance. Yellow cube is rendered with a viewer at 10m, 2m and 500cm z distance. **The fourth image**: A viewer with y position 150 cause the scene to be rotated, but otherwise identical.

3.6 Discussion

The distortion is nearly non noticeable unless the viewer is close to the screen. But the limits are big enough to prevent the user from experience much distortion during regular use. The biggest distortion is in the y axis of the scene, but since most users won't differ more then 10 centimeters in height from the fix viewer position will this usually not be an issues. The distortion in the y axis is not a problem since it won't change the appearance of the object, just the position. The

biggest issue is the limited field of depth which requires the rendered scene to have a size inside the limits of the viewing volume. But this is a problem that could be solved with a software implemementation, unlike the projection distortion which is a limit of the screen design.

The distortions will mostly be non noticeable for the small scaled volume data sets used in the scenarios explored in this thesis. But a scene which is larger than the display volume is not easily viewable with the current renderer.

Chapter 4

Volume Rendering

4.1 Introduction

A ray casting volume render is more complex to implement using the light field display system since the images are projected from the back while the user sees an image from the front. The number of images which makes up a scene will make the renderer considerably more computationally heavy. But when the ray position has been established can many established methods be used to calculate the color. This chapter describes how the ray casting is implemented while also describing the rendering methods used in the implemented software.

4.2 The volume rendering integral

A volume data set can be considered a collection of particles located in space with a varying density. These particles will only be visible if they change the light in any way, a completely transparent volume has no data and can therefore not be visualized. A ray of light can be absorbed by the particles, as fog or clouds, or the particles can emit light, as glowing soot from a fire. They can also scatter light by chancing direction of the incoming rays, but for now is only the simpler absorbtion-emission model considered. This model is widely used since it provides a good approximation for most visualizations while decreasing the amount of computations necessary.

When light passes trough a area of the volume will it be affected by these properties and the number of particles in that particular area. Each light ray (not photons since this is an approximation and not a simulation) with a radiance (I) that pass trough the volume to the user will be affected by the particles in a thin cylinder, considered to be small enough that the volume properties won't change in the breadth direction. The cylinder can be divided into small slices along its

length, each with a length Δs and a cross area E. If the density of particles is ρ will the number of particles in the slice be $N = \rho * \Delta s * E$. If each particle has an area of A and the Δs is close to zero, so no particles can overlap, will the amount of light to be occluded by the slice be equal to the fraction of the area which is occluded by particles $A * N/E = A * \rho * \Delta s$. The amount of light absorbed by the slice per unit length is described as $\tau = A * \rho$. The particles can also emit light with an intensity of L per unit area. This cause the total light emission from the slice to be $L * A * N * Es = L * A * \rho * \Delta s * E$ and the light emitted per unit area to be $L * A * N = L * A * \rho * \Delta s$. This gives an expression for the amount of change in the light rays intensity for one slice with the amount of intensity added equals to $L * A * N = L * A * \rho * \Delta s$ or $L * \tau$ while the amount of light absorbed equals to $I * \tau$. Which gives differential equation for the amount of change in the intensity $\frac{dI}{ds}$ for the ray trough the volume:

$$\frac{dI}{ds} = L(s)\tau(s) - I(s)\tau(s)$$

The radiance I is not wave length dependent in this expression, which means that no colors can be calculated. To solve this must the wave length dependent radiances be calculated, but usually only for a few wavelength bands (red, green and blue). More complex models can include change in wavelength, and wavelength dependent scattering, which is not included here. To form an expression for the whole light ray trough the medium must this differential equation, which can be done by multiplying both sides with:

$$e^{-\int_0^s \tau(t)dt}$$

and integrate the expression over the ray which starts at s_0 at the back of the volume and ends at the eye of the viewer at D. This results in the following integral:

$$I(D) = I_o e^{-\int_{s_0}^{D} \kappa(t)dt} + \int_{s_0}^{D} q(s) e^{-\int_{s_0}^{D} \kappa(t)dt} ds$$

Which is commonly known as the volume rendering integral. The internal integral with the expression for transparency can be exchanged with an expression for the transparency between two points (s_1 and s_2).

$$T(s_1, s_2) = e^{-\int_{s1}^{s_2} \kappa(t)dt}$$

The integral is then expressed as:

$$I(D) = I_o T(s_0, D) + \int_{s_0}^{D} q(s) T(s_0, D) ds$$

But the dataset is not a continuous function and the complexity of the integral makes it almost always impossible to solve analytically. A numerical method is therefore applied to find an approximation for the integrals. A common concept is to split the integration domain (the ray) into several smaller segments ($s_0 ... s_n$) and then sum them together. While this gives a simpler expression will it still be unusable in the implementation of a volume render software. The equation is therefore split into operations which can be executed in sequence. The iteration can go either from the viewer to the back of the volume or from the back to the viewer but still arriving at the same result since they are both a representation of the same volume integral. The first method is called front-to-back and is divided in two equations per iteration, one for the color c and one for transparency α:

$$c = c + (1 - \alpha) * c_{next}$$
$$\alpha = \alpha + (1 - \alpha) * \alpha_{next}$$

While the back-to-front method only requires one equation per iteration:

$$c = (1 - \alpha_{next}) * c_{next} + c$$

The front to back method is better when searching for surfaces as the value first encountered will be at the viewer, which makes it possible to abort the loop if the value cover values behind it. The back to front method don't requires any accumulated transparency, and can be a better approximation of light spread in the volume since it starts with the light ray and not the viewer. For a deeper understanding of the algorithm recommends [9] and [18] where the scattering property is included. The volume rendering integral provides a physically based sampling of the volume, but its implementation in real time computer graphics has not been possible until the last couple of years.

4.3 Ray casting on a light field display

The common method of employing ray casting is to create a proxy geometry, most commonly a cube which is used as a container for the data set. The texture value of the data set can be accessed by mapping each position in the cube with a position in the data set. Either by using three dimensional texture coordinates for each vertex or using a unit cube, where each side has length 1, in which case the vertex positions will be equal to texture coordinates. The later method is used in the current implementation, as the renderer creates a unit cube using quads and use a transform matrix to scale the cube to a chosen size. The ray casting algorithm is used during the fragment shader step to determine the resulting color of the proxy geometry's sides by sampling of the data set which the geometry contains. One

ray is cast for each fragment and the resulting color is send to the frame buffer. It's not difficult to calculate the direction of the ray in a regular volume renderer since a matrix multiplication with the inverse of the view and projection matrices saved by OpenGL will return the position of the current fragment. The ray used is calculated as the ray connecting the position of the fragment and the position of the camera, since the camera and viewer position is the same when rendering to a regular monitor. The procedure is different for the light field display since the projection is done from the projector, while viewer perceives a combined scene from all projectors as described in [3] and chapter 3. The ray direction needs to be calculated from the viewer's viewpoint, and the rays start position will be calculated by using the position of the fragment on the screen, since multiple projectors will project to the same point. The start position of the ray can easily be extracted by storing the unprojected position **S** from the vertex shader in a texture coordinate which can be used by the fragment shader. The direction of the ray is a three dimensional vector pointing from the viewers position and into the volume. The z component is therefore known to be -1 while the x component depend on the current projector position and the y component depend on the viewer position. This results in a direction vector (before normalization):

$$dir = \begin{bmatrix} \frac{S_x - E_x}{E_z} \\ \frac{S_y - V_y}{V_z} \\ -1 \end{bmatrix}$$

Two points are calculated along the ray which slice through the cube. The first one is the point located at position S, the position of the fragment in the world before projection to the display, multiplied by the inverse model view matrix. This matrix will transform the point back to it's initial position before scaling, resulting in an vector with components between 0 and 1. Equal to the coordinates in the stored texture. The inverse model view matrix is automatically calculated by OpenGl using the model view matrix which was activated for the current scene and directly accessible in the fragment shader. The other point is located in direction dir from point S which can be found by following the ray expressed by $(S + dir) * invmodelview$.

The ray will cut the cube in two positions, one at the front of the cube and one in the back. The front point is already known while the back point can be found by evaluating where the ray cuts the cube. The renderer is only concerned with values inside the cube which make it reasonable to only evaluate the ray for these values. An important optimization here is that it is now known how long the ray is and with a constant step size will it be known how many iterations the algorithm has to execute between these points. This is valuable since the stop criteria otherwise had to be tested in every iteration. The ray casting loop can now

simply be applied in the fragment shader by stepping along the ray and applying the volume rendering technique of choice for a pre determined number of loops (see code 4.1).

```
float3 ray = ray_info.start;
float4 c = float4(0,0,0,0);
for(int i=0; i<(ray_info.length/step_size);i++) {
    c +=  ...compositing...
ray += (step_size)*ray_info.dir;
}
```

Figure 4.1: Raycasting code

4.4 Transfer function

The data set does not represent colors directly since it contains one dimensional values representing different levels of density, or magnetic resistance for MRI. Each of these density values need to represent colors in order for the image to be possible to render. This is done by applying a transfer function which taken a one-dimensional input and returns a color value, commonly described with a red, green, blue and alpha component. Such a function can be represented analytically as a mathematical function which is evaluated in the shader when determining color values. Unfortunately will this prevent easy customization to fit the color values to the data set and the users needs. A common solution is to represent each density value as a texture coordinate in a one dimensional texture which contains RGBA colors which is easily editable and effective to use in the rendering pipeline 4.2. The goal is to give relevant color values to different parts of the volume by giving the user the possibility of interactively changing the color map. The tuning of a transfer function is a complex process since it depends on the values for the current data set. Methods exists to help the user choose a transfer function, or even generate one automatically (some are explored in the book by Engel [9]). But the differences between data sets make the process difficult and time consuming.

The transfer function could be applied directly to the data set and thereby creating a data set of colors, a method called pre-classification. But since the mapping from one dimensional values to colors include a interpolation step will the final color values encountered trough rendering be interpolated values of previously interpolated values. This will not be the same a an interpolated value from the data set applied to the transfer function directly and the extra interpolation decrease quality of the colors. A more common approach is post-classification which use the values encountered during rendering as texture coordinates in the transfer function to sample the color value (see code 4.3). The final color value is

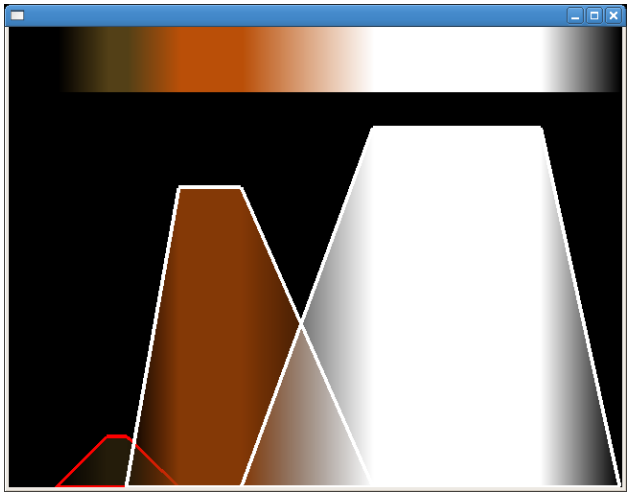

Figure 4.2: **Image**:Transfer function

therefore a color from the transfer function at the coordinate interpolated between the value from the data set which will generally result in a smoother image. It is useful to enable an interactive change of the transfer function, as mentioned above, which is not easily achievable with a pre-classification approach since the whole data set needs to be updated. Post-classification is therefore the most used method of applying a transfer function. Pre-classification can be useful for particular segmented data sets which shouldn't be interpolated, but is not explored in this thesis. [9]

```
float4 tf1D(float3 ray) {
  return tex1D(transfer_function,
               (tex3D(volume_texture,ray).r));
}
```

Figure 4.3: Transfer function code

This implementation use interactive editing of the transfer function, which provides the user with the immediate feedback necessary to change the transfer function to display the wanted details in the current data set. The transfer function used contains 256 RGBA colorvalues values in a one dimensional texture, which quickly can load and update in real time. The size is adapted to 8-bit datasets as it represent the 256 possible values an 8-bit dataset can include. The transfer function should be enlarged if higher quality, for example 16-bit, data sets should be used.

4.4.1 Pre integrated transfer function

The transfer function is often made of segments with steep slopes since such functions easily can be described by some values which simplifies the editing process of the transfer function. This look may cause high frequency components and discontinuities in the image of certain values as the mapping goes from one segment to another. A solution to this problem is to always decrease the step size of the rendering and therefore make smaller steps in the transfer function. This significantly increase the number of loops to be made for each pixel and the approach will not be viable for very high frequency transfer functions. A better solution, described by Engel [10], is to integrated the whole transfer function and using the integral when finding values. The integral between two values on the transfer function will have considered every value even if the patch is full of spikes and discontinuities. An integral can be approximated discretely by a summation if the sampling is small enough. This integral function can then be used as a two-dimensional lookup table for the values during rendering. One sample is taken from the current position and one from the position behind it. The value extracted from the table will be a color representing an approximation of the integral between the previous and the current value (see code 4.4).

A pre-integrated transfer function provides a fast way of enhancing image quality while only require one extra texture lookup per previous texture lookup in the two-dimensional transfer function. The method do require a pre computation of the integral, but since the transfer function is only one dimensional and usually not very big can the integral usually be computed in real time, or at least interactively.

This implementation calculates a new integrated two-dimensional look-up table every time the transfer function is changed by the user. The calculations is an implementation of the method described in [9] but adapted since float values are used in the resulting texture instead of bytes. The extra computational load is barely noticeable, compared to use of a one-dimensional function, but could be very intense for larger transfer functions. A transfer function for a 16-bit dataset would require a two dimensional look up table with 4 294 967 296 values compared to the current 65 536, which could provide difficult to calculate in real time. But 8-bit data sets is common enough that this implementation still can be useful.

```
float4 tf2D(float3 p,float3 n)
  return tex2D(int_transfer_function,
               float2(tex3D(volume_texture,p).r
               ,tex3D(volume_texture,n).r));
```

Figure 4.4: Pre integrated transfer function code

4.5 Lighting

The lighting of a scene is important for the perceived realism as it present information about the layout and structure of surfaces. The most common lighting model used in computer graphics is the Blinn-Phong model, which is an evolution of the Phong light model [20]. It requires knowledge of the surface normal, the direction to the viewer and the direction of incoming light. The total lighting is then described with the ambient (lighting independent of light direction), diffuse (lighting dependent on direction) and specular (lighting dependant on light and viewer direction) light contributions. The amount of each contribution is determined by the attributes of the light and the material of the point, three values which describes the reflective attributes of the material and the luminary attributes of the light. In this implementation is all the volume considered to have the same material, and thereby is only the lighting attributes used for light calculations.

The ambient light is considered an approximation of light reflected from the environment, although the ambient light in the phong model is nothing more then an increase of the brightness. Resulting in a lower contrast. But it helps to make points visible even if they have no direct light.

```
ambient = k_a * l_a;
```

The diffuse light contribution is calculated by use of the dot product of the normal vector and the light direction vector. For simplicity can the light be considered to be so far away that it's rays will be considered parallel. This simplicity make it possible to only use the light direction, not the light position, and thereby only compute the position once per fragment instead of once per step in the loop. The light direction is found by multiplying the inverse of the model view matrix with the untransformed light direction. This transforms the light vector to a vector in the untransformed space of the volume. Thereby will the light direction change depending on the rotation of the bounding geometry. A direction from the z direction, similar to a flashlight held by the viewer, will be calculated as follows:

```
float3 L = normalize(mul(glstate.matrix.inverse.modelview[0],
                         float4(0.0f, 0.0f , 1.0f, 0.0f)).xyz);
```

The diffuse contribution is then calculated as:

```
diffuse = k_d * l_d * dot(L,N);
```

The specular contribution is more complex as it requires knowledge of the viewers position. This direction is already calculated in the raycasting shader since the ray is cast from the viewer's position. In the original method by Phong is the reflection direction of the light needed to calculate the specular highlights but is replaced by the vector halfway between the light direction and viewer direction in the Blinn-Phong method, which simplifies the calculations.

```
float3 H = normalize(L + (-ray_direction) );
```

The specular contribution is the most complex variable and is determined with the use of the Phong constant. This gives an description of the shininess or smoothness of the surface and visually will increase or decrease the size of the specular highlight.

```
specular = k_s * l_s * pow(dot(H,N),k_s);
```

The total light contribution in the point will then be calculated by adding the three contributions together and multiplying the resulting value with the color of the point.

```
new_color = color * (ambient + diffuse + specular);
```

4.6 Gradient

The gradient ∇f is a vector which represent the direction of change of the values in the dataset. The gradient is a analytical expression where each component (x,y,z) represent the derivative in the corresponding direction of the function f, this means that some approximation has to been done to calculate the discrete gradients in the dataset. A common method in volume graphics is the central difference scheme. This method requires two samples per component on opposite sides of the point where the gradient is estimated with a distance h between the samples. The gradient estimation vector is then calculated as:

$$s1 = \begin{pmatrix} f(p+(h,0,0)) \\ f(p+(0,h,0)) \\ f(p+(0,0,h)) \end{pmatrix}, s2 = \begin{pmatrix} f(p-(h,0,0)) \\ f(p-(0,h,0)) \\ f(p-(0,0,h)) \end{pmatrix}$$
$$\nabla f \approx \frac{s1-s2}{2h}$$

The gradient is easily converted to a normal representing the orientation of a surface by simply normalize it to keep the length at zero. The gradient can be calculated in a preprocessing step and stored along with the data-values for the volume.

This method consumes a lot of memory if the data set is big since each vector requires three values each. An other method is to calculate the gradient in the shader for each step in the loop, which increases the amount of work for each loop but also increase the quality of the gradient since the preprocessing method will use an interpolation of the gradient stored in the data set. A gradient calculation during rendering makes it possible to use the classified data, values mapped to the transfer function, in the calculation. The transfer function can map different values to the same color and the surface will therefore change accordingly.

```
sample1.x = tex1D(transfer_function,
                  tex3D(volume_texture,ray+float3(H,0,0)).r).a;
sample2.x = tex1D(transfer_function,
                  tex3D(volume_texture,ray-float3(H,0,0)).r).a;
sample1.y = tex1D(transfer_function,
                  tex3D(volume_texture,ray+float3(0,H,0)).r).a;
sample2.y = tex1D(transfer_function,
                  tex3D(volume_texture,ray-float3(0,H,0)).r).a;
sample1.z = tex1D(transfer_function,
                  tex3D(volume_texture,ray+float3(0,0,H)).r).a;
sample2.z = tex1D(transfer_function,
                  tex3D(volume_texture,ray-float3(0,0,H)).r).a;
float3 gradient = (sample1 - sample2)/(2*H);
float3 N = normalize(gradient);
```

Figure 4.5: Gradient calculation in shader

Some parts of the data set may have areas where values doesn't differ much from one segment to another which result in a gradient with a small magnitude. The normal derived from this gradient is not well defined. A surface represented by several such normals is not clearly defined and the lighting calculated can therefore be varying and noisy. It's possible to define a surfaceness factor, for example the length of the gradient or similar and adapt the lighting calculations according to this to minimize the impact of ill defined normals.

4.7 Stochastic jittering

The constant step size of the ray casting can cause image artifacts since each sample will be taken from the same depth. This cause the image to seem divided in slices since every ray in every step sample a value from the same plane. A simple solution is to move each ray a small random distance before the traversing begins. The method is simple and breaks the patterns of the layered image, but increase noise and discontinuities. Random number generation is not complectly implemented in the shader language since the computations isn't supported on all graphics boards and can be computational heavy. The random numbers needed for the implementation can instead be pre computed and stored in a texture which is

sent to the shader. Each fragment can then use the random number from the texture at the position. This method will save the time needed for computation and also provide each fragment with the same random value each frame, giving the image a consistent look. The noise could be better adapted to the screen by using a texture map with the same size as the screen resolution, but the current implementation use a small texture and the position on the screen for texture lookup. A small random texture is enough to break the layered pattern of the ray caster.

```
ray += ray_info.dir*tex2D(rand_texture,IN.world_screen_position.xy).x*step_size;
```

Figure 4.6: Stochastic jittering

4.8 Compositing Techniques

There exists a multitude of different composition techniques to use with ray casting. Some simulates the real physical properties of light transfer, such as the dvr algorithm, while other try to create an image that enhance the important features of the dataset. Some common methods have been implemented and is described in the following section. The code is meant to be placed inside to ray-casting loop described above.

4.8.1 Simulated xray

This method sums all values on the current ray without taking the alpha value into consideration, which makes the projection similar to a two-dimensional x-ray image. The method is simple but doesn't use the possibilities of the three dimensional data set. It will not discard any data because of small alpha values which can make it interesting for special dataset. The method is order independent since it weights each point on the ray equally disregard of its position.

```
c+= color_function(ray) * weight;
```

Figure 4.7: Xray rendering rendering

4.8.2 Maximum intensity projection (MIP)

The interesting areas in a lot of medical data sets are often the ares with the highest values. Blood vessels is enhanced with contrast agents before ct scanning which makes them absorb more radiation resulting in sharp difference between the vessels and the tissue. The MIP algorithm will step trough the whole ray and compare

each value with the previously highest one encountered. When the loop is finished is the value returned with the transfer function to give the final color of the fragment (see code 4.8). This is an depth oblivious technique since a fragment value always result in the highest value from the ray no matter if the ray is found in the back or the front of the ray. The method is simple and fast and is only useful for visualizing special datasets where the interesting areas have the highest values.

```
if(tex3D(volume_texture,ray).r>max) {
    max = tex3D(volume_texture,ray).r;
    c = color_function(ray);
}
```

Figure 4.8: MIP rendering

4.8.3 Direct volume rendering

The current implementation use the front to back composition algorithm since this makes it possible to break the loop when the accumulated alpha is high enough. The requirement of an alpha buffer is irrelevant since the accumulated color is storing an alpha value in its fourth component, causing no extra memory overhang. A back to front composition is therefore not meaningful in the current context. For each loop will the new color value be picked and added to the accumulated color weighted by the accumulated alpha value, and the same is made for the accumulated alpha. The loop will break if the accumulated alpha value has reached 0.95 percent, since further color accumulations will be insignificant small.

```
next_c = color_function(ray)
c.rgb += (1-c.a)*next_c.rgb;
c.a += (1-c.a)*next_c.a;
if(a>0.95) break;
```

Figure 4.9: DVR rendering

4.8.4 ISO surface

An iso surface is a surface defined by all points in the volume of a certain value. Each ray is traversed until the iso-value is found and a regular phong lighting equation is applied to calculate the color using the normalized gradient in the point. Since the ray advance with discrete step is it unlikely that it ever will end up exactly in the point with a value of the iso-surface. If the value of the data set in a point encountered by the ray must the value of the iso-surface exist somewhere

between the rays current and previous positions. The surface is then calculated by searching for the iso value along the line between the current and previous ray positions using by using a recursive search for the value on the line.

```
if(tex3D(volume_texture,ray)>iso_v) {
float3 x0 = ray - (step_size)*ray_info.dir;
float3 x1 = ray;
float3 mid = (x0+x1)*0.5;
for(int i=0;i<4;i++) {
   (tex3D(volume_texture,mid).a>iso_v)
    ? (x1 = mid) : (x0 = mid);
    mid = (x0+x1)*0.5;
}
...shading...
break;
}
```

Figure 4.10: ISO surface rendering

4.9 Data

The data used by the program is downloaded from an volume data repository [1] where many real world acquisitions and simulated data sets can be found. Many of these has been used in other work which provides a way of making comparisons of the image quality and rendering performance with the techniques used in this program. The files read by the program is uncompressed unsigned bytes with one value in every point. The files is in RAW format and therefore only contain the data values with no information on the layout or the size of the data set. Therefore must the size in each dimension of the data set be known when reading and loading the data to a texture. Otherwise will the program try to read from unallocated memory or miss values. The dimensions is stored, for each data set, in small files which just contains the three integer values for the dimensions. The files, with the extension .inf, have to be created manually as no dimensional data can be extracted from the RAW file format. This method is chosen since the data sets dimensions is a constant value which doesn't need updates. The program contains functions which allow the user to switch dataset during runtime by use of a simple file browser. But it is only possible to load RAW files that are accompanied by an .inf file. Only one data set is kept in memory at a time since the memory requirements of gradients and data values is very high. This forces recalculations of gradients if a dataset is reloaded which prevents fast switches between datasets. But the work environment of medical dataset viewers rarely has to handle more then one data set at a time. A more advance program would use data in the DICOM [19] format since this is a well establish format for medical images. But this will not change

the performance or image quality of the rendering and will therefore be out of the scope for this application.

4.10 Implementation

The program consists in basic of a renderer class, a control panel class, a volume texture class and the shader-programs used during rendering. The program is written in c++ with Trolltech QT for handling gui and interaction. The code used for rendering to the light field display (holo.h), compiling and handling the shader and other important functions using vectors and matrices (SL library) is part of the software developed and used at the visual computing group (vic) at CRS4.

The renderer is a huge class which handles rendering to the screen and the light field display and loading of the shader-programs. This class contains all code which generate the geometry and the preintegrated transfer function necessary for rendering while also handling the user interaction with the renderer. This includes rotation and zoom using the mouse and keyboard interaction to change the rendering mode. When the user press a mouse button while the cursor is over the rendered window will the renderer increase performance while lowering quality to make the scene more interactive. The step size in the raycasting loop is increased and the level in the mipmap of the volume is increased. This will cause the renderer to perform less iteration and faster texture reads from the smaller mipmaped texture. The resolution of the small scenes drawn to the light field display is also decrease to minimize the amount of fragments and thereby the amount of rays cast.

The control panel creates a graphical user interface to display the transfer function and change it interactively. The transfer function will be send to the renderer, where an integrated lookup table will be calculated each time the user changes any values. Loading and saving of the transfer functions as xml files is possible to prevent the tedious task of configuring all values every time the program is restarted.

The volume class include functions to load a volume data-set from the hard-drive and upload it as a texture to the graphics card. Gradients is calculated from the loaded data set using the central difference scheme similar to the one performed in the shader described above. The texture is stored as an three-dimensional texture with four values per texel (texture point). The first three values, RGB, represent the three components of the gradient in that point while the alpha is used for

the original data value. The texture is created with the OpenGL mipmap function to easily calculate the mipmap levels which will be useful to decrease memory load for certain texture lookups.

The vertex shader program is simply performing the projections described in the projection chapter (chapter 3), and sends these projected values to the fragment program.

The fragment shader program include several composting methods. These can quickly be switched between to compare performance and image quality, although a more optimized shader only would run one method, thereby avoiding unessential if statements and function calls. The methods used is xray, MIP, iso-surface, direct volume rendering (dvr) and a combination of dvr and iso-surfaces. All methods use the pre-integrated two-dimensional transfer function for color lookups. The xray and MIP methods is simply implemented as described above and doesn't use any gradient since no lighting is applied in these methods. The Blinn-Phong lighting algorithm is used for the other methods with a gradient calculated either using the central difference scheme on the data set values or using a pre-calculated gradient stored in the data set.

The regular dvr method described above will calculate the gradient in the shader or extract it from the precomputed data, calculating the lighting and shading for every step in the loop. This requires a high sampling rate to not miss values. A sampling rate lower than the resolution of the dataset will miss many values resulting in a noisy image and lack of details. A high sampling rate, were each step in the loop results in a step with a width of one voxel in the dataset, and lighting calculations using precomputed or shader computed normals will result in a clear and detailed image. The method requires many iterations for the ray to traverse the volume using these small step sizes, but the method is simple with an easy adaptation of the transfer function. The last method implemented in the shader use a combination of ISO surface search and DVR. The idea is to combine the smooth surface representation of the ISO surface method with the volume visualization of the DVR algorithm. The search for an ISO surface is performed in every iteration to decrease possibility of missing a surface. The surface will, in most cases, not exist between the points and the returned value will then be the value closest to the iso surface. The ray position will therefore always be changed to the approximated ISO surface position and calculated from there. This method improves the quality of the ISO surface since all rays passing trough the surface will adopt to the surface. The method isn't really using an ISO surface, since the ray keeps on going after the surface is hit, but more of an ISO volume. But the search for an iso surface is costly and only essential if the iso value is present in

Figure 4.11: **Image**:High sampling (0.5 voxel per step) and low sampling (2 voxels per step) dvr rendering

```
next = next color
if(next.a > 0) {
    find ISO
    move ray close to ISO value
    next = new color from the ray position.
    calculate gradient and normal
    c.rgb+=(1-c.a)*
        ...calculate with phong lighting, diffuse, specular, ambient...
    c.a+=(1-c.a)*next.a; //alpha doesn't get phong lighting
    }
```

Figure 4.12: ISO surface with DVR rendering

the current segment. A simple solution is to add an if clause in the loop to switch the rendering method. If the value is below the ISO-value will the regular DVR algorithm be used without gradient calculations. Thereby is the costly ISO-value search skipped for many segments while finding a visibly nice representation of the surface. This approach cause sudden changes in the lighting between segments

```
next = next color
if(next.a > 0) {
    if(next.a > ISO) {
        find ISO
        move ray to ISO
        }
        next = new color from the ray position.
        ...calculate gradient or lookup gradient...
        c.rgb+=(1-c.a)*
             ...calculate with phong lighting, diffuse, specular, ambient..
        c.a+=(1-c.a)*next.a;
}
```

Figure 4.13: ISO surface with DVR rendering, with if clause

as the ray will suddenly jump to the ISO surface during rendering. If the transfer function and the iso-value isn't adapted correctly to the dataset will the rendered

scene look noisy. A break could be added when an iso surface is found to further reduce calculations although such a solution will make transparent ISO surfaces impossible while also increasing the amount of high frequency noise in the final image. Pre calculated gradients are faster, while the transfer function dependent gradient gives a good lighting on surfaces defined by the transfer function. But the extra texture lookups cause a lot of extra work for each loop and the improved image quality is so small that the pre computed gradients is the better choice. The combined DVR ISO surface method gives a high image quality but requires

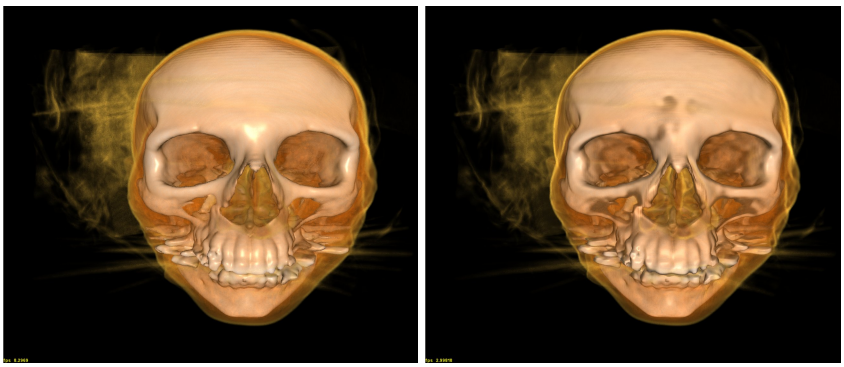

Figure 4.14: **Image**:Renderings of (1)Iso search with dvr and a break and (2)Iso search for all segments with a gradient calculated from the transfer function

a careful optimization of the transfer function and ISO value since the step size is adjusted depending on the ISO value. With a correctly adapted transfer function and ISO-value can the image quality be nearly as good as a DVR rendering using larger step sizes. It gives a clearer representation of surfaces for large step sizes 4.15, but cannot handle transparent surfaces and volumes as good as the ISO method. The best quality and most correct representation of the volume will be rendered with the DVR-method with small step sizes. But the same image quality can be achieved with the ISO DVR method using a correctly adapted transfer function and ISO value with a larger step sizes resulting in faster framerates.

4.10.1 Problems

The problem with a method using a varying step size is that the alpha contribution for each fragment will be the same, even though the segments will be of different lengths and thereby contain more particles, as explained in the section about dvr. A solution to this problem is to adapt the alpha and color value according to segment length as described in [9] and thereby solve the problem. Unfortunately were the implemention of the alpha and color adaptation not successful for the current method. The ray always move toward the ISO value which make

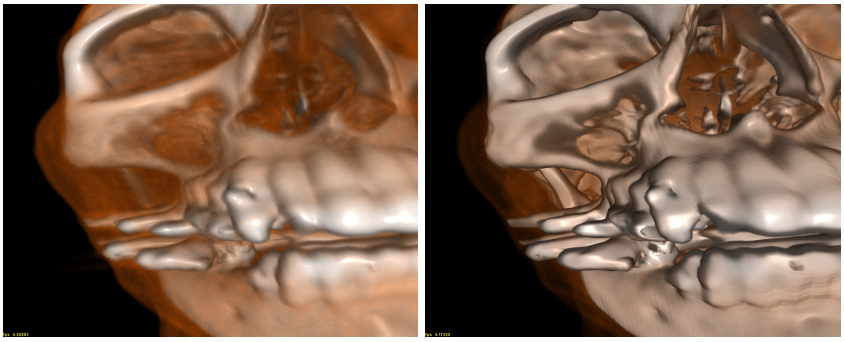

Figure 4.15: **Image**:Closeup on DVR rendering and ISO DVR rendering

some segments very small as the ray passes trough a solid object and jump back toward the surface. The alpha contribution will then be so small that the loop will continue sample values constantly resulting in a saturation of the color for all fragments which rays passes trough a surface. The solution would be to allow a longer, or infinite loop which breaks when the alpha value is high enough, but this will decrease performance too severely. The current method without alpha adaption create nice looking images which doesn't show any noticeable errors in color contribution.

The methods can be further developed by use of the amplitude of the gradient to have a measurement of the "surfaceness" of the iso surface as described in the section about gradient. A method using this approach was developed where the amount of light added depended on the magnitude of the gradient. An other method made an linear interpolation of the position of the ray segment between the ISO surface point and the old ray point. If the gradient was large (a well defined surface is present) was the ray be moved to the ISO surface. It will otherwise stay closer to its old position. These methods gave a small difference in the image appearance, although not in a clear way, while increasing the amount of work to be done per loop. But they could prove useful for certain data sets where gradients are commonly ill defined.

4.10.2 Possible optimization

The current implementation uses the whole dataset as a look up table which force the graphics board to look up a value in the whole 3d texture ever time. Although one texture lookup is very fast will the performance be affected severely by the many texture lookups which is done for each frame when calculating gradients with the two-dimensional pre integrated transfer function. A better approach would be to use the mipmap function when the texture is loaded which creates smaller versions of each texture. A ray segment further a way from the viewer

would not need the full resolution texture to look up texture values. One of the smaller mipmaps would be faster. Unfortunately will the calculations to determine which mipmap to look up increase the computational load. The volume contains a lot of empty space with pixel values which will not contribute to the final image. These points are still looked up by the loop when a ray is passed trough the volume, resulting in unnecessary calculations. An octree could be used to store the min and max value of the different levels. It will then be possible to quickly know if a part of the volume is composed of data that are not useful. This kind of optimization is described in [9] but requires a precalculation of the volume and a search trough the octree during rendering.

4.11 Performance

The display is driven by a single PC running Gentoo Linux with an Athlon64 3300+ processor and a Nvidia 8800 GTX graphics board. Nvidias high-level shading language CG is used for the GPU calculations necessary while the application is written in C++ using Trolltechs QT as a graphical interface.

The scene used for testing, showed in the screenshots, is created from a CT-scanning of a human head dataset downloaded from an volume image repository [1]. The image is 256 values wide in all three dimensions. The scene rendered with a transfer function and ISO-value resulting in the scene displayed below. It contains one iso surface, which is the bones and teeth in the data set and transparent regions for the skin and muscles.

The framerates for the volume rendering to the light field display varies between 0.3 and 2.8 fps. Although its enough to fill the framebuffer and display an image to each projector will the lowest framerates severely affect interaction. Changes in the transfer function and rotation of the scene is not interactive for 0.5 frames per second. The comparable framerates for rendering the same scene to a 1280*1024 pixel big window on a regular monitor varied between 8 (ISO surfaces and dvr with shader gradient calculation) to 18 fps (one iso surface with break from the loop). These are acceptable framerates for interaction and can therefore be used to adapt the scene with transformations and changes in transfer function and then show it in the light field display. The rendering quality was reduced, as described above, when the user use the mouse to transform the scene. This provided an increase of 4-3 fps when the scene was rendered to the light field display, which is an interactive framerate. The step size used for these measurements where 1 voxel, although the ISO-surface search can generate good looking images for up to 4 or 5 voxels per step, depending on data set and transfer function.

[1] http://wwwvis.informatik.uni-stuttgart.de/ engel/pre-integrated/head256.zip

Method	Light field	Desktop
Xray	18	1,4
MIP	15	1,0
ISO surface	18	1,6
DVR	15	0,4
DVR and ISO with if	15	0,87
DVR and ISO with gradient shader	8	0,4

Table 4.1: Framerates

Chapter 5

Perceptual evaluation

5.1 Problem statement

The data sets used in medical application can be very different and will use varying volume rendering techniques in the rendering process. It is not possible to test every possible data set, so the displays performance must be tested in such a way that generalized conclusions might be drawn. This is done by constructing experiments where the viewers will solve tasks in highly specialized environments to enable as much control as possible over which cues are being tested.

The cues occlusion, retinal size and arial perspective is easily achievable when rendering to a two-dimensional display by using regular 3d rendering methods. Culling will remove points with a higher z value (after projection with the view matrix) simulating the occlusion experienced in the real world. By using a perspective view matrix will the projected points move closer together for objects further away, thereby causing these objects to look smaller. The arial perspective is an old computer graphics trick where fragments further away will lose their color and fade in to the background color, this is a cheap way of creating a more realistic environment while keeping the amount of on screen geometry to a minimum. Since these cues are easily mimicked will the test of cues which a two-dimensional single or stereo display can't provide be more important.

Accommodation and convergence is impossible with two-dimensional display, since both eyes look at the same surface. The lack of convergence and accommodation in a stereo display cause unhealthy effects on users for long viewing sessions since their eyes will see two different scenes without rotation or focusing of the eyes as in real life. This multi view display uses a two-dimensional screen for projection, but the eyes of the viewer could potentially rotate to the approximate position they would in real life because of the smooth blend between the projectors. This is very difficult to test since it would require a method of determine the

angle of rotation of the eyes. Something which could be done using appropriate hardware, which is not available for this investigation. The accommodation effect can be possible since the display is using a light field of the scene in the horizontal direction. If the amount of detail received by the eye is enough may the accommodation be perceived in the horizontal direction as light rays from different projectors arrive at different positions in the eye. A camera lens can be used to test this concept as it works similar to a human eye. In the vertical direction will the eyes receive light from one point on the screen surface as the screen scatters light independent of incoming direction.

Motion parallax is possible with a two dimensional display but requires the use of special hardware which can track the user and adjust the camera in the visualization accordingly. This parallax effect can only be experienced for a single user, since the same scene is seen by all observers. The use of special hardware carried by the user for tracking will increase the cost and the complexity of the system.

The evaluation of the display's capability to provide certain cues doesn't show wether these cues will help the user of the display. A comparison with other display systems for solving certain task must therefore be done to get a relevant statistical comparison of the performance of the light field display. Two other systems was available in this experiment, a regular 2D LCD monitor and a stereo display using two projectors which both project onto a screen. The light from each projector is passed through filters while the viewer wears glasses with similar filters. These cause the left image to filtered out from the right eye and vice versa thus creating a stereo view. The tests made with the light field display is performed under different circumstances such as allowing or disallowing movement of the user (disabling motion parallax) or using static or moving scenes.

5.1.1 Tests

Four tests has been carried out to test the display's capabilities and performance. The tests has been designed from ideas used in [16], [17] and [5] which concerns evaluation of volume render techniques and volume display systems.

The first test is an implementation of the test described in [5] which will show if the light field display is better for displaying the volume rendered images compared to a two-dimensional display or a stereo display. The test investigate wether the display system can give an correct understanding of a volume, even though the volume is rendered with object order independent methods.

The second test investigates the possibility that the display can give an accommodation and convergence effect. It will be a big advantage for the display if such an effect can be shown.

The third test tries to rate the motion parallax provided by the display and is determined by how small depth distances can be distinguished. This will highly depend on the resolution of the display.

The fourth test is a more complex test which combines different cues to simulate a more real world problem. This test will simulate a task which isn't uncommon for physician using volume data sets. Since the task must be under strict conditions and provide an analyzable result will it not be possible to use real volume datasets. Instead is a simulated environment with stylized an simple objects used. This test is analyzed statistically more extensively than the previous as more data was available.

The test program is written in c++ using Trolltechs QT as an gui for attribute input for the test, and answer input from the test subject. The program is designed in a modular approach by using a viewer class which handles the rendering of the scene and a control class which sends a structure with the parameters of the current scene to the viewer. It is thereby easy to exchange the rendering loop in the viewer with another code and change content of the structure sent to the viewer to create a new test. The tests provides output of the result to text files and reading a previous saved test configuration using an xml document.

5.2 Stereopsis

5.2.1 Concept

A volume rendered orthogonally with a depth oblivious technique such as Xray or MIP onto a two-dimensional display will be very difficult to understand since the important occlusion cue has been removed from the process. It is not possible to use the size as a cue either since the projection is orthogonal. The image on the screen is then perceived as a rectangle with a randomly varying texture. The extra depth cues provided by the light field display should make the same image appear like areal volume and give the viewer a better understanding of its layout.

A test described in [17] and [5] renders a data set of a rotating cylinder filled with values generated with the Perlin noise algorithm. When such a volume is rendered with Xray or MIP and projected orthogonally will it be impossible to distinguish which direction the cylinder rotates. The test can therefore show whether

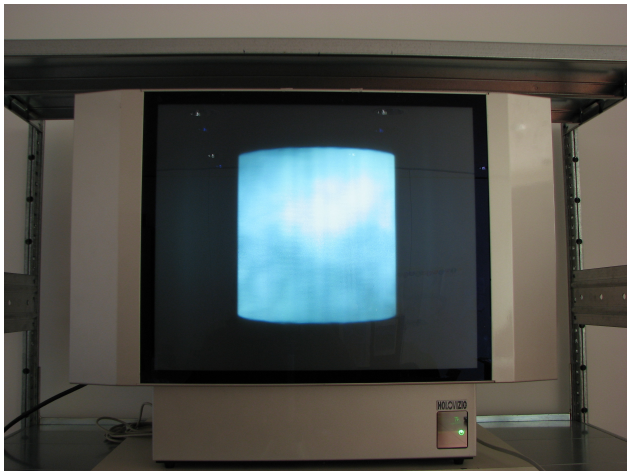

Figure 5.1: **Image**:Rotating cylinder

a display system provides a better understanding of the volume compared to a regular two-dimensional display. The cylinder is generated as a volumetric data set by placing values between 0 and 1 in points inside the described cylinder. The values are generated by the Perlin algorithm from the noise.h c++ file which is available on Ken Perlins homepage [1]. The test program uses the same volume renderer as described in chapter 3 and loads the generated data set as a regular volume dataset. The transfer function in the renderer is not used and the color of the fragments is determined by a white color multiplied by the value from the data set. Both the MIP and Xray method where used in the tests to see if the methods caused any differences in the results. The test was carried out with 5 different subjects were each got 20 trials where the rotating cylinder was shown for one second. The rotation direction, left or right, was determined by a random variable. But the rotations speed where always kept the same. The screen went black when the cylinder had been shown for one second and the subject answered wether the object had rotated to the left or to the right. The test was carried out on the light field display and a regular 2d LCD monitor for both xray and MIP rendering resulting in four separate tests for each subject.

5.2.2 Result

The percentage of correct answers when using the regular monitor was close to 50 in all cases, which implies that the subjects guessed the rotation. The light field displayed gave a much higher amount of correct answers never lower than 70 percent. The rendering method didn't seem important for the results since

[1] http://mrl.nyu.edu/ perlin/doc/oscar.html

Table 5.1: Results for rotating cylinder. Numbers are the percentage of correct answers

Light field display XRAY	 HOLO MIP
100	95
100	75
80	70
75	75
100	100
2D monitor 2D MIP	 2D XRAY
60	40
55	55
50	50
65	70
50	60
50	70
Light field display means:	87
2D display mean:	56.25

both methods gave similar results on both systems. A third test was carried out with the stereo display system described above. This was only carried out with a few persons and didn't result in enough data to draw concrete conclusions. But the percentage of correct answers were the same as for the light field display which should suggest that stereo vision is enough to solve the task. Two separate perspectives gives the cues which were missing from the 2D display. No motion parallax or other cues which the light field display provides were needed, the subjects stayed still during the test. This shows that a three-dimensional volume can be understand easily when rotating around its own axis and viewed with a stereo perspective.

5.3 Convergence and accommodation

5.3.1 Concept

Since the display is able to show a object as a real volume is it possible that the viewer can experience the convergence effect (the eyes rotates to keep the object in focus) and accommodation (the lenses changes shape to keep rays from the same part of the object to arrive at the same point on the retina). This effect might

be visible on vertical lines with separate depth as the lines will be made up of light from different projectors arriving at slightly different position to the viewer. The difficulty with designing a human visual test for this effect is huge since the human eye focuses automatically and quickly. Therefore will a single lens camera be used which can take pictures with different focus settings. The effect will be the same as in the human eye although the camera lens doesn't change shape, only distance to the image plane. Unfortunately will it not be possible to measure the convergence since the camera only use one lens. A similar test was carried out by Jones in the evaluation of their volume display system[16].

The test program renders three wire framed squares at different distances and different sizes to be easily separated. Three different square collections are rendered to show if there is any differences on different locations of the screen.

5.3.2 Results

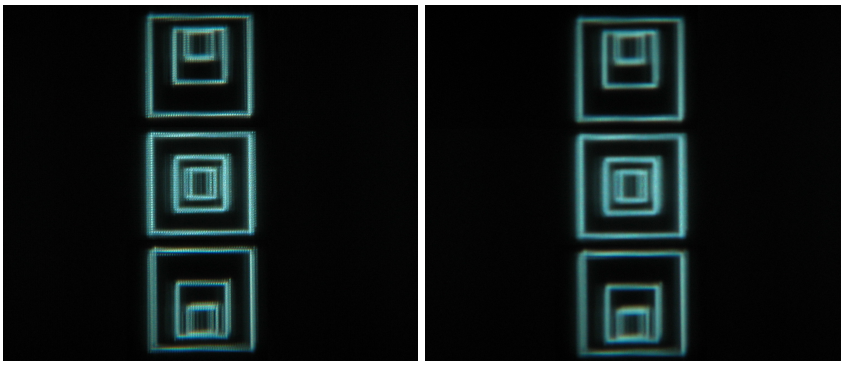

Figure 5.2: **Image**:The test with two different focus settings on the camera.

The pictures were taken with a Canon Powershot 6 Mega-pixel digital camera where the focus distant could be changed by the user. The images show no clear proof of different focus depth in the display which could be because the limited amount of views seems to be to small for the cameras focus resolution. The light rays arriving to the camera is too spread for points rendered outside the display surface. As explained in chapter 3 will the depth position of a point determine its size which makes points outside the screen slightly out of focus (see image 5.2) for any focus setting on the camera. The resolution in a human eye isn't comparable to a digital camera, at least not the standard production model used in this test, and the focal workings of the eye is more precise. It is possible that the display gives a certain accommodation effect but it cannot be proven. But it is more likely that it gives a convergence effect since the eyes are further apart than the small distance of the retina which is needed to give a accommodation effect.

5.4 Depth discrimination

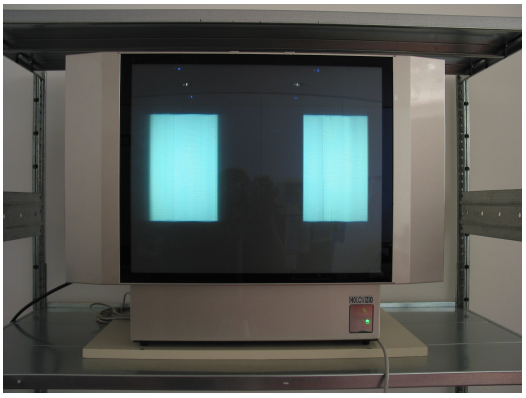

Figure 5.3: **Image**:Depth discrimination test

5.4.1 Concept

The light field display enables horizontal motion parallax since the smooth blend of views from the projectors always show the scene from the corresponding position. The ability to determine depths on the display is a critical factor which must be tested extensively. It is interesting to find the smallest noticeable distance between objects in a scene when the viewer use the motion parallax provided by the multi projector system. A small value will make it easier for the viewer to notice small differences in the scene, although the distance will be different for different people depending on their eyesight. It is preferable to test the displays ability using the cues provided by the display. A rendered scene will provide an occlusion effect of objects drawn in front or in back of other objects because of the depth sorting in the display hardware. The light fields motion parallax is the desired cue to test, which is done by eliminating the other cues.

The test shows two rectangles of the same size on the screen, enough far apart to not overlap each other from any visible direction. The projection on the light field display will cause the rectangles to be rendered in different sizes. A first approach to delete this cue was to add a small random number to the sizes. But the first experiments showed that the different sized rectangles were distracting from other cues. It was obvious that the results were more determined by the randomized sizes than the distance and that the test should be implemented without changes in the rectangle sizes. The small differences in depth between the rectangles would maybe make the difference in size nearly unnoticeable, making the parallax, accommodation and convergence cues much more important. The solu-

tion to eliminate the perceived size difference was to adapt the sizes depending on the distance of the rectangles. This can be done by calculating the angular sizes α and β of the rectangles from the viewers position N and $N + d$ (where d is the distance between the rectangles) and the size of the rectangles a and b. (An approximation since the viewer isn't exactly in front of both rectangles)

$$\begin{aligned}
&\alpha = \tan^{-1}\left(\tfrac{a}{2N}\right) \\
&\beta = \tan^{-1}\left(\tfrac{b}{2(N+d)}\right) \\
&\alpha = \beta \\
&\tan^{-1}\left(\tfrac{b}{2(N+d)}\right) = \tan^{-1}\left(\tfrac{a}{2N}\right) \\
&b = a + \tfrac{a*d}{N}
\end{aligned}$$

By adapting the size b depending on the distance d will the size be eliminated as a cue and the rectangles will be perceived as similar sized which make it possible to carry out the test with only motion parallax as a cue. The test subjects were shown the scene for a decreasingly number of distances, and replied if the left or right rectangle was in front. The test was carried out by 5 subjects.

5.4.2 Result

Distances smaller then 5 cm was difficult to determine and at 1.5 cm was the answers not more correct then guesses (50 percent mean correct answers). But the results show that it was possible to determine small distances even though only motion parallax was the only cue available. A similar test could be carried out with a stereo system using head tracking but unfortunately weren't the time to develop such an application available. The limits is dependant on the resolution and the number of projectors for the current display system as more projectors could fill up the missing views required for the smallest distances.

5.5 Complex structure discrimination

The previous tests has mainly described the display characteristics and image quality. This information is valuable but doesn't provide information of how the display handle in specific task. A more task centric test were therefore developed which could evaluate the value of the display in medical analysis.

5.5.1 Angiography

Angiography is a common invasive procedure for diagnosis of vascular diseases by tracking the shapes and layout of blood vessels of the body. An example is the

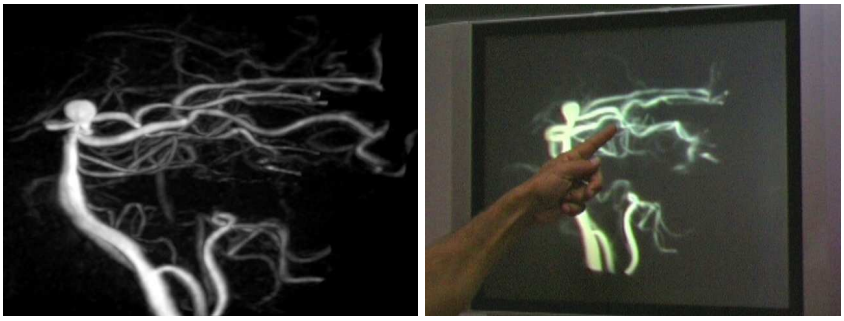

Figure 5.4: **Image**:A volume rendering of an aneurysm. Left: The depth oblivious technique makes the layout difficult to understand. Right: The same scene rendered to light field display is easy for the viewer to understand.

search for an aneurysm, an abnormal widening of a blood vessel usually caused by weakness in the wall of the blood vessel 5.4. These can be present in any part of the body but are most dangerous when present in the brain. The common method of the analysis is to observe 2d-images of 3d-structures, in these cases CT-scans or X-rays of the blood vessels which have been filled with a radiocontrast agent (see image 5.4). The contrast agent absorbs enough radiation to sharply reproduce the blood vessels in the resulting image. The layout of the bloodvessels is a complex tree shaped structure which, for the untrained eye, can be difficult to understand. Especially since the common reproduction of the images is an object order independent MIP-rendering of the volume. Physicians need lot of training and also good understanding of 3D-layout to understand the tree structure when no depth cues are available. The light field display is therefore a promising development to make the diagnosis task easier, especially since it provides the same depth cues for all viewers. Cooperation is a valuable perk of the display. a a test program, which rendered a scene similar to an order independent tree structure, was developed for evaluation of the display capabilities.

5.5.2 Test program

A test program was designed starting from an idea described in [2] which generates a collection of lines shown in a static scene. The program generates a collection of connected lines, branches, which is formed by randomly create vertices in a 3 dimensional space between 0 and 1 (see image 5.5). The number of connected lines in a branch is a random number between some values chosen for the test, usually 1 to 5, and the task for the test subject is to count the number of connected corners in a certain branch. At each vertex is a small cube drawn to enhance the perception of the corners as overlapping lines will make the corners

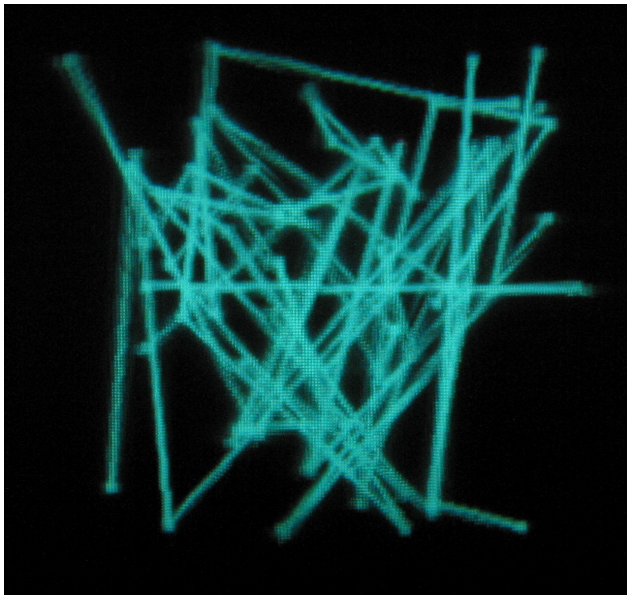

Figure 5.5: **Image**:Collection of branches

non perceivable. The starting vertex of the line which the subject will follow is marked with a red cube to be easily distinguished.

The program uses a cylinder-ray intersection test [6] when placing the vertices to minimize the risk of overlapping lines. All previously generated lines are tested with the new line when a vertex is about to be placed and a line should only be created at a certain distance from other the other lines. This distance is determined by cylinders that are defined between every existing line. If an intersection is found will the vertex be generated with new values until some values are found which doesn't intersect with previous lines. The loop will only run for a limited number of iterations since a solution might not be possible for all vertices using this method, especially for large number of vertices. Each line is split in many small lines when drawn because of the non-linear projection as explained in chapter 5.

The test was made with 13 people, where some had previous experience and deep knowledge of the display system while others had never seen it before. 10 randomly generated scenes with 20 branches and one to five lines per branch where displayed to the subjects. Each subject viewed a different scene as a new scene were generated for every test. The randomness of the test is chosen to cover as many different circumstances as possible and make it possible the redo tests as no one can learn the patterns. A small problem with the approach is that some scenes might be more difficult to interpreted, which could cause some subjects to score worse then others. The subjects didn't have any time constraint and could move around freely while solving the task to try to overcome the difficulties with

the random scenes.

The same test made on a regular monitor, for comparability, where the user could rotate the structure with a mouse. The two-dimensional display has a slightly higher resolution than the light field display while the rotation will make it possible to rotate the scene 360 degrees giving a much more complete view of the structure. The both methods cannot be considered equal, but they represent two common ways of displaying data.

5.5.3 One-way ANOVA analysis

It is vital to use the collected data from the tests in a relevant way to be able to draw any conclusions. The objective should be to show that the data from the tests using the light field display differs, hopefully in a positive way, from the same test using a two-dimensional display. The goal can thereby also be seen as disproving the hypothesis that the two tests result in data distributed in the same way, which means that the datasets will be composed of randomly sampled variables from distributions using the same mean. This is the null hypothesis (H_0), which if its true shows that the use of different displays will not affect the outcome of the tests. If the null hypothesis can be disproved will we show that the display has an impact on the test outcome.

One technique for hypothesis testing is called analysis of variance (ANOVA), the type of ANOVA focused on here is the simplest one-way ANOVA where the groups are independent. The analyze is done by splitting the variance of all samples into two parts, one using the difference between the value of the sample and the group mean (called ”within group”) and one using the difference between the group mean and the total mean (called ”between groups”). The ratio between these two components is the test statistic, usually denoted the F value in ANOVA. The F value is a value drawn from a F-distribution which makes it possible to calculate a probability value for the achieved F value. The null hypothesis can be rejected if the probability, p ,is sufficiently low (usually $p < 0.05$).

As the name implies will the variances of the result be necessary to perform the analysis. But first must the derivation between the values and the means of the dataset be calculated together. There exist one global mean Y_tot and one mean value for each group g Y_g. Three derivation are calculated for each score Y_i; deviation from global mean ($Y_i - Y_tot$), deviation from group mean ($Y_i - Y_g$) and deviation of the group mean from the global mean ($Y_g - Y_tot$). The variance is the mean of the squared deviation and thereby must first the squared deviation be calculated. As many values is present will the summed squared deviation (SS) be

used instead, which is simply a summation of the squared deviations.

$$SS_{total} = \sum (Y_i - Y_{global})^2$$
$$SS_{between-groups} = \sum (Y_g - Y_{global})^2$$
$$SS_{wihin-groups} = \sum (Y_i - Y_g)^2$$

The squared sums are then used to calculate the variance, commonly refereed to as Mean Squares or MS for short. They are calculated by dividing the SS with the degrees of freedoms as follows:

$$df_{between-groups} = k - 1$$
$$df_{within-groups} = \sum_{i=1}^{k} (n_i - 1)$$

Where k = the number of groups and n_i = the number of values in group i. The MS can then be calculated as:

$$MS_{between-groups} = \frac{SS_{between-groups}}{df_{between-groups}}$$
$$MS_{within-groups} = \frac{SS_{within-groups}}{df_{within-groups}}$$
$$F_{(df_{between-groups}, df_{within-groups})} = \frac{MS_{between-groups}}{MS_{within-groups}}$$

The resulting F value represents a value picked from a F distribution with the degrees of freedom $(df_{between-groups}, df_{within-groups})$. The probability of picking this value is calculated or taken from an F-distribution table, a statistical calculator or calculated by the formulas for an F-distribution.

Since the results from the tests was saved in a custom file format and no statistical software was easily accessible came the solution to implement the ANOVA analysis in a custom written c++ program which could quickly read all results from the tests, calculate the means, the summed squares and the resulting F value. The implementation of a probability calculation for the F-distribution was not necessary since user friendly calculators could be find on the internet, hosted by reliable sources (for example http//davidmlane.com/hyperstat/F_table.html). The ANOVA analysis program read all result files created by the test program, calculated the mean values for the chosen variable for each group and a global mean. A problem surfaced when some of the first tests didn't include a variable for time since they used a older version of the test software that didn't use a timer. It was necessary to use all tests but disregard the one without time values since a value of zero will afflict the calculations. This made it possible to use time as a variable in the ANOVA calculation program. Unfortunately does such exceptions limits the flexibility of the program as it is highly specialized for this particular data set, but is enough for the needs of the experiment.

Light field display mean:	7.05
2D with mouse Mean	7.18182
Global mean	7.11591
df Numerator	1
df Denominator	29
F	0.0451025
p from table	0.83330

Table 5.2: Number of right answers

Light field display Mean:	6.15
2D with mouse mean	4.27273
Global mean	5.21136
df Numerator	1
df Denominator	29
F	1.73418
p from table	0.19819

Table 5.3: Size of error

5.5.4 Result

The results where processed by the c++ program described above using the ANOVA analysis and three different results were analyzed.

The number of right answers for the test was very similar between the two different environment (see table 5.2). This is measured as the number of correctly answered trials for one test. A probability of 0.83 shows that the null hypothesis probably is correct. There's no important difference between the two methods.

The size of the error is the difference between the number of nodes guessed by the user and the number of nodes that exists in the branch. The 2d with mouse approach has a few less errors, which could be regarded as it's easier to make a mistake on the light field display and follow the wrong path (see table 5.3). A probability of 0.19 is low but not enough to discard the null hypothesis. But it shows that there might exist a difference between the two methods. It might be more difficult to follow some paths which span the whole volume when the light field display is used as the viewer has no view of the back of the scene. This makes it easy to follow the wrong path and thereby give a big error. As long as the vertices is separated in the horizontal direction will the task be easy since the subject can use the motion parallax to see different parts of the scene. The

Light field display mean:	143.962
2D with mouse mean	208.358
Global mean	176.16
df Numerator	1
df Denominator	17
F	2.48479
p from table	0.13338

Table 5.4: Time summation

difficulty increases substantially when the nodes are located in the same vertical plane. Small distances between nodes is also a problem because of the limited resolution of the display. It is possible to explore the whole volume with mouse input and therefore easier to see the structure from all directions. Thereby might it be easier to not follow wrong paths.

Time summation is simply measured as the time from the started until the user input the last answer. Some of the performed test didn't have any time measurement as this was a feature later implemented in the experiment program and these subjects were therefore discarded from the ANOVA analysis. The probability of 0.13 shows that some different might exist between the two groups (see table 5.4). The task seems to be faster on the light field display. A reason for this might be the lack of input devices since usage of the mouse will require more time to rotate and observe the scene while the user of the light field display quickly can change viewpoint of the scene.

5.6 Discussion

These tests shows that the light field display gives an increased understanding of volume images since it can provide a motion parallax for all viewers of the display. It was not possible to determine wether the display could give a accommodation and convergence cue, although it might be present. The display showed an advantage in following pattern since a viewer can look around the scene without rotating it.

There exists a lot of possibilities to make different perception tests, but it also requires a lot of knowledge of the human visual system and depth perception.

Chapter 6

Evaluation in clinical context

6.1 Introduction

Medical displays are evaluated using guidelines from the American Association of Physicists in Medicine (AAPM) Task Group 18. Displays need to fulfill certain criteria to prevent artifacts and lack of visible details which could potentially lead to wrong diagnosis and harm for patients. The guidelines include visual, quantitative and advance testing methods. The characteristics, which is tested with specially designed test patterns, include reflection, geometric distortion, luminance resolution and display artifacts. The qualitative tests isn't possible to perform for this thesis as no equipment for measurement of light was available, but the guidelines feature a visual evaluation using the test patterns which will be enough to draw some conclusions. The light field display's construction and rendering pipeline make it very difficult for the display to be compared to regular two-dimensional medical monitors. The light field display has a relative low resolution compared to medical displays. Which commonly work in resolution 1600X1200 or even 2048x2560 pixels.

Some displays uses a higher color range of 10-bit instead of 8-bits, which this light field display use, per pixel (actually 8-bit per color component, but a grayscale image have the same value in every component) to represent gray scale pixels. Which makes it possible to represent 1024 shades of grey instead of the 256 possible in the light field display. The multi projection system creates a slight variation in the brightness over the display surface because of the slight different attributes of the projectors, although some of this is taking into account for in the rendering pipeline.

It's thereby improbable that the light field display will perform as good as a two-dimensional display dedicated to displaying grayscale images. But the visual tests can provide information in which areas the light field display lacks most in

quality. These test will show the performance of this particular display and will not be a generalization for other systems with the same technology. Other systems may have brighter projectors with higher resolution which would provide a clearer and brighter projection.

6.2 Implementation

The test-patterns are 1024x1024 or 2048x2048 pixels big 16-bit png images. The 16-bit gray-scale tiff format of the images is not supported in Trolltech QT which is used for reading the image files. Therefore has the images been converted to 8-bit png format before loading. The 16-bit format is not commonly supported by image editors and the conversion from 16 to 8 bit is a process which is handled differently by different software. This caused a lot of problem and the conversion to 8 bits result in a loss of data. An other method would have been to read the 16-bit values directly and use these as the OpenGL texture since OpenGL has support for 16-bit texture formats. This would decrease the error from conversions but make the image loading process more complex. Unfortunately didn't the time exist to implement such a solution and redo the evaluation.

These images are loaded by the program and used with OpenGL as a texture for a rendered rectangle. The rectangle is positioned at z coordinate zero to be rendered exactly on the screen of the light field display. This forces a resampling of the image to display it on the light field display, since it uses several projectors with resolutions of 320x240 pixels. Which makes it impossible to color a screen pixel directly unlike a regular monitor where every pixel drawn corresponds to a pixel on screen (if the screens native resolution is used).

The tests was performed by loading the chosen 8-bit image as a texture and correcting the gamma on the graphics card to adapt the rendered image to the brightness of the light field display. Each image used had some attributes described in [22] which could be evaluated by observing the rendered image carefully.

6.3 TG18 test patterns for display assessment

6.3.1 TG18-QC

The QC pattern is a multi purpose pattern displaying different patterns and luminance values. It is used in this examination to evaluate distortion, contrast and frequency response. The contrast can be evaluated by observing the 16 squares of different illuminance values arranged around the middle of the image. All of

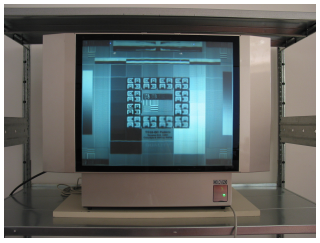

Figure 6.1: **Image**:TG18-QC

these should be able to distinguish, but the three whitest squares in the rendered image were not distinguishable. A small distortion was visible on the long lines in the image. Although the lines appeared straight were slight distortions sometimes visible, an error which might be an caused by small differences between the projectors. The frequency response is determined by the visibility of the line pair patterns in the center of the image. The high frequency patterns are not displayed correctly but the horizonal pattern looked more distorted than the vertical. This can be a cause of the horizontal distribution of projectors. The lines in the vertical pattern is a combination of lines from different projectors projected slightly differently and blended by the screen. The horizontal lines is projected on the same position since the screen only scatter these rays (as explained in chapter 3). The image projected will, as explained above, be a combination of low resolution versions of the big 1024x1024 test pattern. The combination of different projectors makes the image smoother but still in a low resolution, thus downsampling the horizontal lines and thereby losing the high frequency data.

6.3.2 TG18-BR

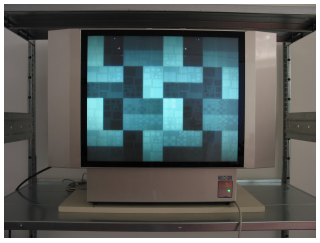

Figure 6.2: **Image**:TG18-BR

The BR pattern is a collection of differently sized squares with different luminance. The collection of squares are located in areas with different luminance and the evaluation is performed by observing wether some illuminance combinations are harder to distinguish than others. The smallest squares were not visible on the

light field display because of the limited resolution. But most squares were clearly visible and although the darkest were less visible than the bright ones. Although the medium luminance areas between squares were not visible at all.

6.3.3 TG18-LP H V

Figure 6.3: **Image**:TG18-LP, the horizontal version. Left: original size, Right: zoomed

This pattern display high resolution low contrast images using three different levels of luminance (10, 50 and 89 percent of white). The pattern consists of lines, either vertical or horizontal which span the whole image width or height and the evaluation is simply to see wether all lines are visible in all the different luminance levels. The horizontal lines were visible for all luminance levels although the lowest level (10 percent) required a close examination of the screen to distinguish the line pattern. The vertical pattern were not visible for any of the contrasts levels. This is an unexpected result since the vertical high resolution pattern was clearly visible in the QC pattern. An explanation could be that the contrast difference between the lines were much smaller in this pattern which could be worsen by the blending on pixels in the horizontal direction. Vertical lines are then blended and blurred slightly and will be invisible if the contrast between the lines isn't high enough.

6.3.4 TG18-UN

This pattern consists of a homogenous luminance value which is used to show the light distribution of the display. An optimal display should show the pattern with the same luminance in every point. The luminance is uneven when rendered to the light field display. Two large bright areas are visible to the left and right border of the display. This is an effect of the mirrored projectors which contribute with twice the amount of light (see chapter 3). An LCD projector always emit

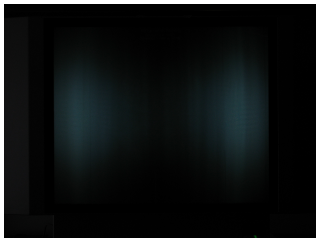

Figure 6.4: **Image**:TG18-UN

some light since the LCD filter cannot stop all light from the projectors lamp. This creates an uneven light distribution for low luminance values in homogenous scenes. The effect decrease for higher luminance values since the pixel colors will be adapted in the fragment shader used by the renderer (see chapter 3).

6.3.5 TG18-CH

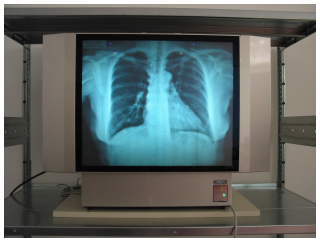

Figure 6.5: **Image**:TG18-CH

This test pattern is an xray image of the chest of human. A trained physician who is used to work with these kinds of images can easily see if the displayed image is shown incorrectly on the screen as there's features and small details that should be easy to spot. Unfortunately were no trained physician available for the test, but the pattern description includes some features which should be evaluated. The vascular pattern in the lung and aorta should be clearly visible with the heart and the diaphragm. The overall sharpness and contrast of the image should also be evaluated as with the previous patterns.

The displayed image on the light field display was clearly represented, although the sharpness of features were limited, similar to previous patterns. The contrast seemed a bit lacking which clearly shows that the color reproduction is limited. The white areas of the image where hard to tell apart which made it difficult to spot the aorta. It should be seen in front of the spine but was not clearly visible. The vascular pattern of the lungs were clearly visible but the heart was

difficult to distinguish. The diaphragm separating the thorax and the abdomen could be seen just below the lungs.

The limits of using an 8-bit converted image can clearly be seen in this test. A lot of contrast have disappeared during the conversion. It could be possible to render the image in 16-bit and use some tonemapping technique to downsample it to the 8-bit which the display can show. But it is difficult for this display to challenge displays constructed to show grayscale images.

6.3.6 TG18-KN

Figure 6.6: **Image**:TG18-KN

This test pattern is an xray image of a knee, and the procedure is similar to the chest xray in that some features should be evaluated. The reproduction of soft and bone tissue is the primary feature of this image. The contrast difference between the bone and soft tissue is very high, making the clearly visible in the knee. But the soft tissue is barely visible in some places because of the low luminance values. An other important aspect is the trabecular details inside the bone, which should be seen as the lines and spots on the bone. These can not clearly be seen because of the limited resolution.

6.4 Result

This display cannot show two-dimensional images in a high resolution correctly because of the multi projector system. The general contrast and brightness of the images were usually acceptable, but a regular 2D LCD monitor can easily challenge the light field display in both contrast and brightness while offering a much higher resolution. This shows that this current display cannot replace a 2D display system for showing 2D images. A version using brighter projectors with higher resolution could probably show a 2D image better. But the complex projection make the result difficult to predict.

Chapter 7

Conclusion and discussion

7.1 Conclusion

The software developed for this thesis can load volume datasets and render them using raycasting with several different techniques applied to display high quality images on the light field display. The software use a pre-integrated version of the transfer function, calculated in every update of the transfer function, to increase image quality. No empty space leaping or other advanced methods were used to increase performance. The framerate was high enough to provide interactive, but not real time, possibilities for the user to adapt the transfer function and change view of the scene.

The light field display seems promising for medical visualization of volumetric images. Several tests were made which focused on exploring the perception of three dimensional scenes on the light field display. These showed that the display gives the observer an strong feeling of the depth in the scene. Primary using the motion parallax provided by the multiple projector system. A big advantage with this system compared to stereo displays is that the motion parallax will be provided to all users, inside the display view boundaries, that observes the scene. Thereby presenting many opportunities for collaborative tasks. It couldn't be proved that the display provided a convergence or accommodation cue, although it might be present but too small to be recognized by the camera used in the test. Users solved a task, designed to simulate a real world data set, quicker, but not more correctly, with the light field display than with the use of a regular 2D display system since they could navigate the scene without rotating it.

The display constructions cause some small artifacts and limits, but they are not a big problem since they showed to not effect image quality in any significant way in rendering volume datasets. The light field display cannot replace a 2D monitor for displaying 2D images since the screen pixels won't correspond

to the pixel sent to a projector but a combination of several pixels from several projectors, were each has a small resolution of 320x240. The biggest problem is therefore the limited resolution of the projectors, which is simply a matter of scale. The performance of mini projectors will steadily increase, especially with the led based projectors. A new display using the same architecture but brighter projectors with a higher resolution could potentially solve the problems of the current system, although an increased resolution will require more computing power to fill all the pixels. But since the graphics hardware also steadily increases in performance should the extra pixels be of no worry.

The display can therefore be considered a viable option for displaying medical volume datasets, but not as an replacement for 2D-displays since the current performance isn't high enough to properly display high quality 2D images.

7.2 Future studies

The volume rendering application developed for this thesis is far from optimized and a lot of performance can be gained by optimization of its functions. The display pipeline simply renders one whole scene for each projector, a technique which is straight forward but doesn't take into account the dependencies between different scenes.

The perception tests made in this thesis is based upon previous tests which are not developed for this kind of display. The tests used came from other evaluations which evaluated volume displays, and not multi projector systems such as this. If multi view systems becomes common will the possibility to compare these devices be valuable. Similar to the test patterns which can be shown on 2D display systems could something similar be used for 3D displays. Like a a volumetric test pattern which could present an easily comparable image. The goal should be to have generalized tests which could evaluate multi view and volumetric display systems using standardized methods similar to the medical display evaluation available today. This goal requires more knowledge of perceptual psychology and the workings of human depth understanding. The simulated task centric test is interesting but can only give an approximation of the display performance. It would have been valuable to perform real medical tasks on the display, using real data sets and letting trained physicians solve the task.

The future will present an increasing computing power to help the visualization of the human body where the goal would be to visualize a human being in way which makes it as simple and natural to explore the body in great detail on the inside just as on the outside. The display technology is a big part in this goal as volumetric displays will present images in a way which makes them understandable and natural for human eyes. With better technology and deeper understanding

of the human visual system will this goals be met.

Bibliography

[1] http://www.volvis.org/.

[2] ACM. *An Evaluation of Depth Perception on Volumetric Displays*, New York, NY, USA, 2006.

[3] Marco Agus, Enrico Gobbetti, Jos Antonio Iglesias Guitin, Fabio Marton, and Giovanni Pintore. Gpu accelerated direct volume rendering on an interactive light field display. *EUROGRAPHICS*, 27(2), 2008.

[4] Tibor Balogh, Tams Forgcs, Tibor Agocs, Olivier Balet, Eric Bouvier, Fabio Bettio, Enrico Gobbetti, and Gianluigi Zanetti. A scalable hardware and software system for the holographic display of interactive graphics applications. In *EUROGRAPHICS 2005 Short Papers Proceedings*, Conference Held in Dublin, Ireland, August 2005, 2005.

[5] Christian Boucheny, Georges-Pierre Bonneau, Jacques Droulez, Guillaume Thibault, and Stéphane Ploix. A perceptive evaluation of volume rendering techniques. In *ACM Symposium on Applied Perception in Graphics and Visualization*. ACM, ACM SIGGRAPH, 2007.

[6] Joseph M. Cychosz and Jr. Warren N. Waggenspack. Intersecting a ray with a cylinder. In *Graphics gems IV*, pages 356–365. Academic Press Professional, Inc., San Diego, CA, USA, 1994.

[7] N. A. Dodgson. Autostereoscopic 3d display. *Computer*, 38(8):31–36, 2005.

[8] N.A. Dodgson, J.R. Moore, S.R. Lang, G. Martin, and P. Canepa. A 50 time-multiplexed autostereoscopic display. *Proc. SPIE–Stereoscopic Displays & Applications XI*, 3957, 2000.

[9] K. Engel. *Real-Time Volume Graphics*. AK Peters, Ltd., 2006.

[10] K. Engel, M. Kraus, and T. Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. *Proceedings of the ACM*

SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware, pages 9–16, 2001.

[11] G. Favalora, R. Dorval, D. Hall, and J. Napoli. Volumetric three-dimensional display system with rasterization hardware. In *Stereoscopic Displays and Virtual Reality Systems VII*, volume 4297 of *SPIE Proceedings*, pages 227–235, 2001.

[12] Michael Halle. Multiple viewpoint rendering. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 243–254, New York, NY, USA, 1998. ACM Press.

[13] T. Hübner and R. Pajarola. SINGLE-PASS MULTI-VIEW VOLUME RENDERING.

[14] M. Huebschman, B. Munjuluri, and H.R. Garner. Dynamic holographic 3-d image projection. *Optics Ex-press*, 11:437–445, 2003.

[15] Peter M. Vishton James E. Cutting. Perceiving layout and knowing distances: The integration, relative potency, and contextual use of different information about depth. In *Handbook of perception and cognition*, volume Vol5 Perception of space and motion, pages 66–117. Academic Press, 1995.

[16] A. Jones, I. McDowall, H. Yamada, M. Bolas, and P. Debevec. Rendering for an interactive 360 light field display. *ACM Transactions on Graphics (TOG)*, 26(3), 2007.

[17] Marta A. Kersten, A. James Stewart, Niko Troje, , and Randy Ellis. Enhancing depth perception in translucent volumes. *IEEE Transactions on Visualization and Computer Graphics Journal*, 12(6):1117–1123, September-October 2006.

[18] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[19] P. Mildenberger, M. Eichelberg, and E. Martin. Introduction to the DICOM standard. *European Radiology*, 12(4):920–927, 2002.

[20] B.T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[21] N. Qian. Binocular Disparity Review and the Perception of Depth. *Neuron*, 18(3):359–368, 1997.

[22] E. Samei, A. Badano, D. Chakraborty, K. Compton, C. Cornelius, K. Corrigan, M.J. Flynn, B. Hemminger, N. Hangiandreou, J. Johnson, et al. Assessment of display performance for medical imaging systems: Executive summary of AAPM TG18 report. *Medical Physics*, 32:1205, 2005.

[23] M. Stanley, P. Conway, S. Coomber, J. Jones, D. Scat-Tergood, C. Slinger, B. Bannister, C. Brown, W. Crossland, and A. Travis. A novel electro-optic modulator system for the production of dynamic images from giga-pixel computer generated holograms. In *Practical Holography XIV and Holographic Materials VI*, volume 3956 of *SPIE Proceedings*, pages 13–22, 2000.

[24] S. Stegmaier, M. Strengert, T. Klein, and T. Ertl. A Simple and Flexible Volume Rendering Framework for Graphics-Hardware–based Raycasting. *Proceedings of the International Workshop on Volume Graphics*, 5:187–195, 2005.

[25] J.P. Wann, S. Rushton, and M. Mon-Williams. Natural problems for stereoscopic depth perception in virtual environments. *Vision Research*, 35(19):2731–2736, 1995.

[26] Matthias Zwicker, Wojciech Matusik, Fredo Durand, and Hanspeter Pfister. Antialiasing for automultiscopic 3d displays. In *Eurographics Symposium on Rendering 2006*. Eurographics, 2006.