# Volume visulization of medical scans in virtual reality

- Master thesis - Vision, Graphics & Interactive Systems - Medialogy -

Project Report by
Mads Bang Hoffensetz & Christian Nygaard Daugbjerg

Aalborg University
Media Technology
Electronics and IT

This report is set with the document markup language LaTeX, originally develop by Leslie Lamport and based on Donald Knuth's TeX. The body text is set with *Computer Modern* pt 11, designed by Donald Knuth. The document was written with the use of `www.overleaf.com`, an online collaborative writing and publishing system for LaTeX. For development and testing MATLAB and Intel RealSense SDK were used.

**School of Information and Communication Technology**
Fredrik Bajers Vej 7
DK-9220 Aalborg Ø

**AALBORG UNIVERSITY**
STUDENT REPORT

**Title:**
Volume Visualization of medical scans in virtual reality

**Theme:**
Master Thesis

**Project Period:**
1st February 2018 - 29th May 2018

**Project Group:**
Mixed gr. 181040

**Participants:**
Mads Bang Hoffensetz
Christian Nygaard Daugbjerg

**Supervisor:**
Martin Kraus

**Page Numbers:** 42

**Appendices:** 4 Pages

**Date of Completion:**
May 29th, 2018

**Abstract:**

Medical scanners, such as CT- and MR- scanners, produce a stack of 2D scans, each displaying a slice through the patient. Doctors inspect these scans separately or side by side when diagnosing the patient or planning the surgical procedure. We propose a system that combines the stack of 2D medical scans and uses volume visualization to render a 3D reconstruction of the scanned area of the patient in virtual reality.
Alongside the volume visualization, we propose several tools that the doctors can use to explore the data. First is a menu from which the user can select preset transfer functions. Transfer functions are used to map the volume data to alpha and color values. Second is a clipping plane that assigns different transfer functions to the part of the volume above the plane, e.g., rendering skin below and the skeleton above, or rendering everything above fully transparent to give a clear view of the slice. The user can freely position and orientate the clipping plane
An evaluation with medical personnel showed that the participants were able to diagnose a patient purely based on the proposed system.

# Contents

# Preface

This report covers the master's thesis project made by Mads Bang Hoffensetz in the master's programme Medialogy and Christian Nygaard Daugbjerg in the master's programme Vision, Graphics and, Interactive Systems at Aalborg University, in the spring semester 2018. The project has been carried out in collaboration with Aalborg University Hospital and Aalborg University

We would like to give our thanks to our collaborators at Aalborg University Hospital, Jane Petersson, and Johan Poulsen, for a constructive collaboration and aid in evaluating the project.

Aalborg University, 29th May 2018

Mads Bang Hoffensetz
mhoffe13@student.aau.dk

Christian Nygaard Daugbjerg
cdaugb13@student.aau.dk

# Project summary (in Danish)

Den langsigtede motivationen bag dette rapport er at udvikle et system der kan kombinere medicinske røntgen scanninger, som CT- og MR-scanninger, med *da Vinci* robotic surgery system for minimally invasive surgery. *da Vinci* robotic surgery system bruges til robot assisterede operation, hvor kirurgen sidder ved en terminal og styrer robotarme med operations instrumenter. Patient ligger på en båre med robotarmen stukket gennem huden, så kirurgen kan bruge dem til at operere med. Ud over instrumenterne bliver der også placeret en robotarm med et stereoskopisk kamera i patienten, så kirurgen kan se patienten indefra med stereo syn under operationen. Det projekt der var foreslået i samarbejde med Aalborg universitets hospital var at bruge volume visualization til at lave en 3D rekonstruktion af patienten ud fra røntgen scanninger og kombinere det live kamerafeed af patienten med 3D rekonstruktionen. Den løsning skulle give kirurgerne mere information under operationen og praktisk talk give dem mulighed for at "se" igennem fedt, væv og organer. Fokusset for dette projekt er at udvikle volume visualization delen af systemet, og derfor ikke at bygge et interface mellem *da Vinci* robotic surgery system og vores system. Som delmål fokuserer dette projekt på at udvikle et system der virtuelt kan rekonstruere det scannede område af en patient ud fra røntgen scanninger og give brugen værktøjer til at udforske rekonstruktionen. Systemet skulle kunne give hospitalspersonalet samme information til atr diagnoserne en patient og planlægge en operation. Røgten skanninger, som CT- og MR-scanninger, producerer en stak 2D billeder, som hver viser et snit gennem patienten. Lægerne kigger på scanningerne side om side for at sætte patientens diagnose. Vores system skal kunne give samme information til lægerne, som 2D scanningerne gør, men i en samlet visualisering. Vores system er udviklet til at blive vist i et virtual reality miljø, for at emulere det stetoskopiske syn system som *da Vinci* robotic surgery transmitterer.

Som nævnt tidligere brugte vi teknikken volume visualization til at vise en 3D rekonstruktion af en patient. Som navnet antyder bruges teknikken til at visualisere 3D data, et volumen, som i vores tilfælde kom fra en stak 2D skanninger. Kort opsummeret fungerer teknikken sådan at man for hver pixel følger den pixel's viewing ray gennem volumenet og samler data volumen data punkter undervejs. Værdierne bliver kortlagt til en farve og alpha værdi, med en såkaldt transfer funktion, og samlet til den endelige farve som er outputtet for den pixel.

Som værktøjer til at udforske volumen visualiseringen designede vi to værktøjer. Det første er en række forudindstillede transfer functions, som brugen frit kunne skifte imellem. De giver brugeren mulighed for se forskellige aspekter a volumenet, f.eks. skelettet eller organer. Det andet værktøj er en klippe-plan, som brugeren kan flytte på. Klippe-planet klipper den del af volumen som er over planet, så brugen kan komme til at se et bestemt snit gennem patienten.

Systemet blev evalueret af ekspert brugere, dvs. læger og andre hospitals medarbejdere. Overordnet gav de en meget positiv respons på systemet og det var muligt for dem at diagnoserne patienten uden problemer. Nogle havde nogle problemer med at interagere med systemet og skulle bruge ekstra instruktioner fra os før de forstod det. De påpegede også nogle elementer der mangler for at systemet skulle kunne bruge i praksis, som f.eks. at kunne lave præcise længde målinger.

Opsummeret er konklusionen på projektet at systemet forsyner lægerne med den information de skal bruge for at kunne diagnoserne patienten, men mangler nogle forbedringer før den skulle kunne bruges i praksis. Derudover ville det nok tage en tilvænningsperiode at få indført systemet workflow.

# 1 | Introduction

When setting a medical diagnosis and planning a surgery, the doctors examine X-ray scans. The medical scans can, for example, show the location of a tumor and help identify the best approach to remove it. The motivation for this project is to make the first step towards incorporating medical scans with the *da Vinci* robotic surgery system for minimally invasive surgery.

During robot-assisted surgery, the surgeons view a real-time video feed from a camera placed inside the patient. Incorporating medical scans of the patient with the live camera feed would essentially provide the ability to "see through" the patient which in turn could help the surgeons to find the best point of entry for the surgical operations and avoid vital organs, blood vessels, and tissue.

With the rendering technique volume visualization, it is possible to create virtual 3D reconstructions of the scanned area of the patient from medical scans, such as CT- and MR-scans. The end goal of this project is to superimpose the reconstruction over camera feed from the surgical system. For this project, it is out of scope to both develop a robust volume visualization system and implement it with the *da Vinci* robotic surgery system. The focus of this project is to develop a volume visualization system based on actual medical scans, which accommodates the requirements of the doctors at Aalborg University Hospital to diagnose patients and plan surgeries.

The robotic surgery system uses a stereoscopic camera, providing the surgeons with depth when viewing it through the stereo display. To make our system resemble the robotic surgery system the volume visualizations is developed for virtual reality environments.

# 2 | Problem Description

In collaboration with Aalborg University Hospital, a project was proposed involving the use of the rendering technique volume visualization and virtual reality to visualize medical scans in three dimensions. Medical scanners produce a stack of 2D images, each containing a slice through the patient. Doctors use the scans to diagnose patients and plan surgeries. The proposed project was to combine the stack of 2D scans into a 3D volume and virtually reconstruct the scanned area of the patient with volume visualization. A 3D reconstruction of the patient would improve the surgery planning process, as the doctor would have all information in one object instead of in a viewing a collection of 2D image side by side. The 3D visualization would also improve upon the 2D slices by contributing with spatial context to the area of interest and by providing a look of the area of interest from an arbitrary angle.

The focus of this project is to propose a system that constructs and visualizes a 3D volume from a stack of 2D medical scans. The visualization should work in a VR environment, and the user should be able to manipulate it in real time, which means performance is an essential factor as well as visual quality. The proposed solution should ideally interface directly with other medical equipment. i.e., it should be able to process the files provided the medical scanners.

The project also focuses on proposing an intuitive interface for the doctors and surgeons to manipulate the appearance of the volume visualization and access desired information.

From the described problem statement we arrive at the following problem statement.

*How can volume visualization be used to create a 3D reconstruction of a patient from a set 2D medical scans, and provide the doctors with information needed to diagnose patients and plan a surgery?*

## 2.1 Medical scans

For developing and testing our system, Aalborg University Hospital provided us with two sets of CT-scans. However, to avoid showing personally identifiable information in this report, any figures related to medical scans is based on the CT Lymph Nodes dataset from the cancer imaging archive, which are freely available to use for commercial, scientific and educational purposes (Roth et al., 2014).

One scan of a patient consists of a collection of 2D images which each contain a slice through your body. *Figure: 2.1* shows a few examples of these slices.

**(a)** **(b)** **(c)**

**Figure 2.1:** Examples of slices from CT-scans

The slice has a grey scale value for each pixel. A black pixel means there is no information, i.e., air. Very bright pixels usually represents bone or calcium.

The pixel width and height of the provided CT scans were all 512 pixels. The depth, i.e., number of slices, varied, as the slice count and the thickness of the slices can be adjusted on the scanner. Thinner slices provide more accurate information but introduce more noise. The two sets of scans supplied to us consisted of 392 and 372 number of slices, with 0.625 mm and 0.6 mm slice thickness respectively.

# 3 | Related works

In the 80s volume rendering started emerging, first in the papers by Levoy (1988) and Drebin et al. (1988). Previous techniques for visualization of volumes involved approximating surfaces with geometric primitives (Šrámek, 2006). These surface rendering techniques require an unambiguous segmentation of the volume data to build the surface models. Depending on the data, a segmentation might be a difficult task, e.g., for thin structures or edges with low contrast. Another downside to surface rendering is the potential loss of data, as any non-surface voxels will not be rendered (Kaufman, 1999). Volume rendering techniques do not employ intermediate geometrical primitives to render the volume but try to capture the entire volume in a single 2D image, to avoid the potential loss of information. This also makes the necessity of segmentation obsolete, and fuzzy spectral classification replaces it, i.e., mapping the density values of the volume to optical parameters *volume rendering integral* with so-called *transfer functions*. The *volume rendering integral* and *transfer functions* are further described in *Section 5.1* and *Section 5.2.3*. The use of transfer functions opens up for a broad range of possible visualizations. The two most prominent volume rendering areas are Direct surface rendering/isosurfacing and direct volume rendering which are discussed below.

## 3.1 Direct surface rendering / isosurfacing

Volume rendering techniques in the category of direct surface rendering achieve visualizations resembling the earlier surface rendering techniques, with the difference being that no explicit surface model is built from geometric primitives. Instead, the transfer function is set up to emphasize the desired isosurfaces.

Parker et al. (2005) proposed a method to extract multiple isosurfaces from medical CT scans to visualize skin and bone. As the isosurface of the skin layer occludes the bone isosurface, they implemented a version of the algorithm that renders the outer layer semi-transparent to visualize multiple layers of isosurfaces in the same image. Hadwiger et al. (2005) proposed an adaptive sampling algorithm, to accurately find the intersection between the viewing ray and the isosurface.

Parker presents how additional shading, like shadows and specular reflection, enhance visual cues of the visualization. Hadwiger et al. (2005) performs high-quality shading of the isosurface by first reconstructing the gradient over the volume, which is used as implicit normals of the surface and for computations of the curvature. Blinn-Phong shading and tone shading is computed from the surface normal vectors. The curvature information makes a variety of non-photorealistic effects possible like silhouette outlining.

## 3.2 Direct volume rendering

Direct volume rendering techniques aim at displaying the entire dataset by accumulating contributions of volume data along a viewing ray to a pixel in the output image. The mapping from the transfer function defines the contribution of each data point.

Stegmaier et al. (2005) proposed a simple ray casting algorithm for direct volume rendering. The algorithm involves tracing an independent ray out for every pixel, sampling and accumulating volume data at a finite number of positions along the ray to produce the color and opacity of the pixel. The algorithm is implemented with a single shader pass, which is possible due to the looping and branching support of DirectX Pixel Shader 3.0. For previous shader models, the algorithm had to rely on multiple rendering passes and additional pixels tests, e.g., early depth test, to simulate loops with dynamic iterations.

## 3.3 Optimization

Various techniques have been proposed to improve the efficiency of the ray casting and sampling. Hadwiger et al. (2005) and Kruger and Westermann (2003) use empty space skipping methods to avoid unimportant data sampling. Both perform empty space skipping by dividing the volume into a number of blocks and storing the minimum and maximum data values in each block. The minimum and maximum information are used to determine if the block contains any useful data, or the entire block can be skipped.

Additionally, Kruger proposed an early ray termination method, to avoid traversing the data further, when any following samples will not contribute to the final result. The technique is to compare the so-far accumulated opacity of a ray to a threshed for each iteration of the sampling loop. When the accumulated opacity exceeds the threshold, no further values are sampled along the viewing ray.

LaMar et al. (1999) extended direct volume rendering with multi-resolution techniques. The multi-resolution is achieved by constructing a hierarchy with different resolutions of the volume. Based on the user's field of view and distance to the volume it is determined which quality level of the resolution hierarchy to use for each voxel of the volume.

## 3.4 Data exploration

In statistics, data exploration is a step in the data analysis process where the analysts get an overview of the data. This concept can be adopted in volume rendering as users typically find themselves in a similar situation; with a lot of data in which they need to find specific information. Multiple publications propose tools to aid the user in exploring the volume data, which this section discusses.

### 3.4.1 Volume Clipping

Clipping is a straightforward technique for data exploration that involves cutting parts of the volume, so they do not appear in the visualization. The method can be used to get a better look at occluded regions of the volume or removing unnecessary data.

An elementary form of clipping is to use clipping planes aligned with the faces of the volume, as used by Rößler et al. (2006) and Van Gelder and Kim (1996). This technique allows for visualization of axis-aligned cuts. Kniss et al. (2001) proposed a more advanced version of the axis-aligned cutting planes that instead use a single cutting plane with an arbitrary position and orientation. By giving the user control over the clipping planes position and orientation, it provides the options to view any cut of the volume. In addition to clipping the volume, Kniss et al. (2001) also maps the slice through the volume to the clipping plane which can be blended and visualized independently of the volume itself.

Clipping can be performed with more complex structures than planes as well. Stegmaier et al. (2005) proposed a method which uses a binary volume for clipping. By multiplying the binary volume with the data volume, each voxel of the data volume is either clipped or kept. Similarly, Rößler et al. (2006) proposed a system that used a human brain atlas to display specific areas of the brain.

Both Stegmaier et al. (2005) and Rößler et al. (2006) concludes that clipping techniques has a severe impact on the performance. Rößler et al. (2006) used a volume of size $256^3$ and a viewport of size 510 x 820 for testing. An average frame rate of 27.20 fps was achieved without clipping, and 11.31 fps was achieved with clipping. The paper did not specify if clipping planes or a brain atlas was used as the clipping technique. Stegmaier et al. (2005) tested multiple volume sizes, the experiment most resembling that of Rößler et al. (2006) was with a volume of size $256^2$ x 110. With a viewport with a $512^2$ resolution they achieved an average framerate of 10.8 fps, and with a $1024^2$ resolution 3.2 fps was achieved.

### 3.4.2 Segment view

If the volume visualization includes opaque isosurfaces, it might be necessary to implement data exploration tools to view areas of the volume that are occluded by opaque isosurfaces. Both Bruckner and Groller (2006) and Viola et al. (2006) propose solutions to this problem. Bruckner and Groller (2006) introduces a technique that creates exploded views of the volume data. First, the volume has to be segmented, e.g., into skin and bone. Each segment is further divided into a number of sub-segments which will separate in the exploded view. The user chooses a segment to view, and the system transitions into an exploded view by moving any occluding segments to the sides. The tool can also be used to view the other sides of segments e.g., the back side of the skin when it is offset from the skeleton. Viola et al. (2006) proposes a similar method, that also requires segmentation of the volume and the user selects which segment to view. To view a segment of the volume any occluding voxels are rendered transparent or semi-transparent. The method also estimates the most optimal viewpoint for view the selected segment.

Bruckner and Groller (2006) and Viola et al. (2006) both conclude that their systems has the potential to be useful data exploration tool for volume visualization, but the performances of their current implementations are too low for use in a real context.

The VR app *Medical Holodeck* (www.medicalholodeck.com) uses multiple clipping planes as well as interactive user-driven adjustments of transfer functions. The user can use a virtual UI attached to their HTC Vive or Oculus Rift controllers to select an intensity range within which to map the transfer function. All intensities outside this range are mapped to full transparency. Medical Holodeck boasts a lot of features and allows a great deal of customization. The app supports collaboration such that multiple people can explore the same model in an on-line session, using voice chat and annotation tools within the app. This, according to the developers, makes it ideal for educational purposes or even preoperative planning.

### 3.4.3 Visual enhancement

Visual enhancement of volume data involves highlighting, magnifying or enhancing important information of the volume to aid the user in understanding the data. Ebert and Rheingans (2000) and Taerum et al. (2006) proposes solutions in this area. Ebert and Rheingans (2000) experimented with several non-photorealistic rendering techniques to enhance features and, improve depth and orientation cues. Feature enhancement involves the enhancement of boundaries and silhouettes. Depth and orientation cues involve tone shading and intensity based on depth. The purpose of the solution proposed byTaerum et al. (2006) is to give close-up views of small areas of the volume, by having the user select the area of interest in a low-resolution volume rendering and displaying that area in larger scale and with a higher resolution next to the volume. This technique allows the user to investigate an area of interest of the volume, while still having the spatial context of the entire volume.

Both Ebert and Rheingans (2000) and Taerum et al. (2006) conclude that their visual enhancement techniques provide the user with information that can improve their understanding of the volume.

# 4 | Requirements Specification

This section specifies the list of technical requirements, which this project aims to fulfill. The requirements are set up to satisfy the problem described in *Chapter 2* and are based on previous work in the field presented in *Chapter 3*.

1. The system must be able to combine a stack of medical scans of a patient into a 3D volume data structure.
2. The system must be able to visualize the 3D volume so that it provides the user with at least the same information 2D medicals scans does.
3. The volume visualization must work in a VR environment.
4. The volume visualization must be rendered at real time.

# 5 | Theory

## 5.1 Optical Model for Direct Volume Rendering

Direct volume rendering can be derived from optical models of light interaction with a volume. In this work, we consider a volume to consist of scalar values arranged in a 3D grid. For a physical representation of a 3D volume, the data values can be considered as particle densities of a "participating medium", which both absorbs light passing through it and emits light. The absorption and emission are the optical properties of the medium. The light that leaves the volume and reaches the eye, along a single ray, can be computed by the volume rendering integral, presented in *Equation: (5.1)* (Max, 1995) (Hadwiger et al., 2006).

$$I(D) = I_0 e^{-\int_{t_0}^{D} \tau(t')\mathrm{d}t'} + \int_{t_0}^{D} g(t) e^{-\int_{t}^{D} \tau(t')\mathrm{d}t'} \mathrm{d}t \tag{5.1}$$

The volume rendering integral incorporates both the absorption and the emission of the medium. $t_0$ and $D$ respectively represent the start and end points of a light ray that ends at the eye and starts at the opposite side of the volume. The background light that enters the volume at $t = t_0$ is represented by $I_0$ and the light that exits the volume and reaches the eye is described by $I(D)$. The absorption of the volume between points $t$ and $t_0$ is represented by the term

$$T(t) = e^{-\int_{t_0}^{t} \tau(t')\mathrm{d}t'} \tag{5.2}$$

Where $\tau(t)$ is the extinction coefficient at $t$, which defines the rate at which light is occluded. Opacity is defined as

$$\alpha = 1 - T(t). \tag{5.3}$$

The emission of the volume between points $t_0$ and $t$ is represented by the term

$$I(t) = \int_{t_0}^{t} g(t)\mathrm{d}t \tag{5.4}$$

where g(t) is the source term which describes the light emission at that point in the volume. Usually, the source term is a mapping from a scalar density to a color with a transfer function. To calculate the integral numerical integration is required. The Riemann sum is the most straightforward solution for this purpose (Max, 1995). The volume rendering integral can be approximated over $n$ ray segments of equal length. The length of the ray segments is defined as

$$\Delta x = (D - t_0)/n \tag{5.5}$$

and the Riemann sum yields

$$I(D) \approx I_0 \prod_{i=1}^{n} t_i + \sum_{i=1}^{n} g_i \prod_{j=i+1}^{n} t_j \tag{5.6}$$

where

$$t_i = e^{-\tau(i\Delta x + t_0)\Delta x} \tag{5.7}$$

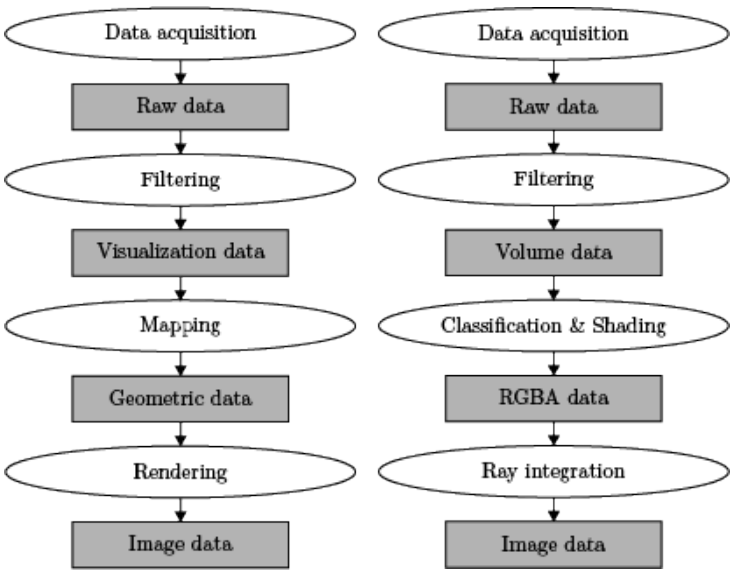
defines the transparency of the $i$th segment and

$$g_i = g(i\Delta x + t_0)\Delta x \tag{5.8}$$

defines the source term of the $i$th segment. $t_0$ here is the back most position on the ray rather than a transparency value.

## 5.2 Visualization and volume rendering pipeline

For the design of visualization algorithms and applications, the visualization pipeline is a valuable tool. Many variations on the pipeline exist, for this project we refer to the pipeline from Kraus (2003), presented in *Figure: 5.1a*. The rectangular nodes represent intermediate data structures.




**(a)** Visualization **(b)** Volume rendering

**Figure 5.1:** Pipelines.

The pipeline describes the flow from acquiring raw data to visualizing it on the screen. The visualization pipeline in *Figure: 5.1a* includes four stages: data acquisition, filtering, mapping, and rendering.

*Data acquisition* comprises all actions required to acquire and import raw data into the pipeline. The *filtering* stage reduces and performs pre-processing on data, as the raw data often is too detailed or only a selection of the data is needed. The following step is *mapping*, which transforms the filtered data into geometric primitives and the final stage is *rendering* the primitives to the screen.

Volume rendering does not necessarily fit into the visualization pipeline. Especially the mapping and rendering stages does not fit, as volume rendering often does not involve generating primitives, instead the image is composited of a number of samples from the volume. Instead a modified version of the visualization pipeline named the volume rendering pipeline exits. Just like the visualization pipeline, many variations of the volume rendering pipeline exits. Again we refer to the pipeline by Kraus (2003), which is presented in *Figure: 5.1b*. This variation is aimed at direct volume rendering. The four stages are described in the following subsections.

### 5.2.1 Data acquisition

The purpose of the *data acquisition* stage has not changed from the original visualization pipeline, only the type of data might differ.

For medical applications, data from CT and MR scans are often used. The scanners produce a stack of 2D scans stored in the DICOM file format. For description of the DICOM file format, see *Section 5.4*.

### 5.2.2 Filtering

Filtering options for volumetric data can include downsampling and geometric clipping of the data, as raw volume data might take up a lot of memory or be more detailed than necessary. Different methods of clipping are described in *Section 3.4.1*.

### 5.2.3 Classification and shading

*Classification and shading* is the process of mapping the data set from scalar values to optical properties of the volume integral (*Equation: (5.1)*), i.e., the extinction coefficient ($\tau(t)$) and the source term ($g(t)$). The classification typically assigns the discretized properties color ($C$) and opacity ($\alpha$), which are combined into the RGBA values. Transfer functions represent the mapping from volume data to optical properties. The application of transfer functions can be performed before or after the scalar values are sampled during ray integration. Applying the transfer function before ray integration means that every data point of the volume is converted to RGBA values before ray integration and during ray integration, these values are sampled directly and can be composited without conversion. This approach is called pre-classification. The other option is to perform ray integration on volume data and apply the transfer functions to each sampled data value before compositing. This method is called post-classification. Pre-classification requires more memory, but fewer computations during ray integration compared to post-integration.

### 5.2.4 Ray integration

To generate image data the volume rendering integral is employed to compute the color and alpha value for each pixel.

The color and alpha values for a pixel are found by computing the volume rendering integral discretized by the Riemann sum (*Equation: (5.6)*) along a pixels viewing ray. The Riemann sum (*Equation: (5.6)*) can be computed iteratively by compositing. The most common compositing schemes are front-to-back and back-to-front compositing (Weiskopf, 2006).

As indicated by the name, front-to-back compositing steps through the volume from the front side to the back side. The front of the volume is defined where the viewing ray enters the volume, and the back is where the viewing ray exits the volume. The compositing equations for front-to-back are

$$C_{dst} = \alpha_{dst}C_{dst} + (1 - \alpha_{dst})C_{src}, \tag{5.9}$$

$$\alpha_{dst} = \alpha_{dst} + (1 - \alpha_{dst})\alpha_{src}. \tag{5.10}$$

C and $\alpha$ represents the color and opacity respectively. The subscript *src* denotes the data from the current position of the ray value and *dst* denotes the accumulated color and opacity.

The compositing equation for back-to-front is

$$C_{dst} = \alpha_{src}C_{src} + (1 - \alpha_{src})C_{dst}. \tag{5.11}$$

The opacity of the destination color does not need to be accumulated, as only the source alpha is used in the compositing.

The color and alpha values for each pixel makes up the image data, which is displayed on the screen or passed to a frame buffer and thereby concluding the volume rendering pipeline.

To be able to reconstruct a signal, one must sample at least the Nyquist frequency of that signal. Kraus (2003) states that in our case this means that the samples along a ray should match or exceed the number of voxels along it. If a transfer function is used, which is not the identity function, one must further increase the sampling frequency, making it the product of the Nyquist frequencies of the signal and the transfer function (Kraus, 2003).

## 5.3 Optimization

As described in *Section 3.3*, various schemes have been proposed for optimizing the process of volume rendering. This section describes the optimizations implemented in this project.

### 5.3.1 Empty space skipping

As the name suggests empty-space skipping is the process of skipping parts of the data, that is going to be rendered transparent anyway. Volume data often contains areas with unimportant information, e.g., in medical scans the air around the patient is not relevant to visualize. The theory presented in this section is based on Hadwiger et al. (2005).

In the base volume rendering algorithm, rays are traced through the volume and data points are sampled along the rays. The volume is usually stored in a cube structure, and the start positions of the rays are at the edge of the cube. The specific goal of empty-space skipping is to alter the ray start positions to be near to first voxel containing essential information instead of the cube edge.

Empty-space skipping can roughly be divided into two steps: Identifying the empty spaces and utilizing the information the volume visualization algorithm.

Identifying the empty spaces is done by subdividing the volume into small blocks, e.g., with a size of $4^3$ or $8^3$ voxels per block. For each block that minimum and maximum intensity within the block is stored in an additional structure. The blocks do not change the volume, but the minimum and maximum values can be used to determine if a block contains essential information or an empty space.

To obtain ray start positions from the min/max blocks the front faces of the bounding geometry of the non-empty space blocks are rendered in the correct positions relative to the volume. The geometry is rendered to an offscreen frame buffer that stores its depth values. The depth values define where the rays should start. The same is done with the back faces to obtain the end positions for the ray tracing.

Before initiating the ray tracing process, the volume visualization algorithm looks up in the texture containing the stored depth values to set the start and end positions for the ray tracing.

### 5.3.2 Early ray termination

Early ray termination is a straightforward method for optimizing volume visualization. During the ray integration step, the color and opacity are iteratively accumulated. Early ray termination is merely a check during each iteration if the accumulated opacity is above a certain threshold close to one. The technique is useful because when the accumulated opacity reaches or gets close to one any subsequently sampled values will have very little to no effect on the final color.

### 5.3.3 Pre-integrated classification

Visualizing CT or MRI scans in form of volumetric scalar fields is often more useful when combined with specialized transfer functions, highlighting certain aspects of the model. These transfer functions are often not linear and contribute to the combined Nyquist frequency of the system. Depending of the visualization criteria, transfer functions might be segmented into multiple colors as well as having one or more peaks and otherwise high frequency changes. In GPU based direct volume implementations, transfer functions are usually encoded into 1D textures, effectively making them piecewise linear functions because of linear interpolation in the texture sampler. If the sampling frequency does not exceed the volume Nyquist frequency multiplied by the transfer function Nyquist frequency, visible aliasing artifacts will occur. Pre-integrated classification aims to solve this, providing better visual results at a lower sampling frequency (Kraus, 2003).

The reason we would normally need to factor in the nyquist frequency of the transfer function is that the scalar field might change more than one unit across one ray segment. The also discretely sampled transfer function texture might contain information between these two values, which is lost, unless we integrate over the range. Integrating over ranges in the transfer function in addition to evaluating the volume rendering integral is a lot of work. Pre-integrated classification solves this by storing the pre-calculated integration of the transfer function for all possible input value ranges. If e.g. a given ray segment goes through a steep gradient inside the volume, i.e. starts at a low value and ends at a relatively large value or vice versa, these two values are used to look up the result of the integration of the transfer function across the ray segment, stored in a look-up table. This look-up table is two-dimensional and stores the mapping of every combination of start and end value.

## 5.4 DICOM

Digital Imaging and Communications in Medicine (DICOM) is the international standard for storing and transmitting medical image data. It defines a file format for medical images that can be exchanged with the data and quality necessary for medical use. DICOM was designed with the purpose of having one format that works across all medical equipment and workstations from different manufacturers. The primary use of the format is in areas like radiology, cardiology imaging, and radiotherapy.

The DICOM files contain additional data alongside the images. The data can, for example, be patient ID to make sure the scans are never separated from the patient Mildenberger et al. (2002).

# 6 | System design

This chapter presents the design of system developed in this project. The design choices aim to fulfill the requirements presented in *Chapter 4*.

## 6.1 Hardware

Ferrand et al. (2016) mention how tracking of the viewer's position enables motion parallax, which is a powerful depth cue. The correlation between movements of the user and the virtual vantage point creates a strong sense of immersion as well as an intuitive way to explore a visualization. Systems like the Vive include tracking of the HMD, providing the benefits mentioned above. Headsets like The HTC Vive and Oculus Rift promise this immersion at a much lower price than some alternative setups like CAVE (Ferrand et al., 2016). For developing our visualization prototype, we use a powerful computer with a NVIDIA GeForce GTX 1080 graphics card situated in a dedicated university VR lab and a regular HTC Vive HMD. The setup we use for user testing is a powerful laptop with a NVIDIA GeForce GTX 1060 graphics card and the HP windows mixed reality headset, as it is better suited for transportation.

| | HTC Vive | HP Windows Mixed Reality Headset |
|---|---|---|
| Resolution | 2160 x 1200 | 2880 x 1440 |
| Display | OLED | LED |
| Refresh rate | 90Hz | Up to 90Hz |
| Field of view | 110° | Up to 105° |

**Table 6.1:** HMD specifications

Both the HTC Vive and the HP mixed reality systems come with two motion controllers. The controllers are tracked like the HMD, which means the user's hand movements can be used within the VR environment. The controllers for the two headsets have very similar button layouts, which makes it simpler to develop software that works with both systems.

## 6.2 Software

The Unity3D game engine is easy to use and allows for fast prototyping. A great advantage of using a popular game engine is the fact that it interfaces well with HMDs and other possible display systems (Ferrand et al., 2016). Unity is used by millions of users worldwide and is continually improved. Apart from games, Unity has been used to create several "serious" apps (Ferrand et al., 2016). We develop our visualization prototype in Unity 2017.3.

We use the SteamVR plugin for Unity from Valve Corporation. It provides a set of easy-to-use components for VR prototyping and development. Especially useful to us is the interaction system included in the plugin. Apart from the prebuilt player object, which represents the user inside the virtual reality world, the plugin also comes with components that make it easy to set up a scene in which the user can pick up objects with the controllers and move them around. Interacting with various user interfaces is also made more accessible, giving us the freedom to experiment with interaction methods, e.g., modifying transfer functions while using the app.

## 6.3 Virtual environment setup

To fulfill the satisfy the requirements in *Chapter 4* the system needs to visualize a 3D reconstruction from medical scans in a VR environment and provide the user with tools to manipulate to explore the visualization.

As the purpose of our system is to be used in a professional setting, we chose to present the visualization in an otherwise empty VR environment with no distractions. The VR scene consists of floor platform and a dark background. When launching the system, the user starts in the middle of the VR scene, and the visualization is placed in front of him. Figure *Figure: 6.1* presents a screenshot from the VR scene.

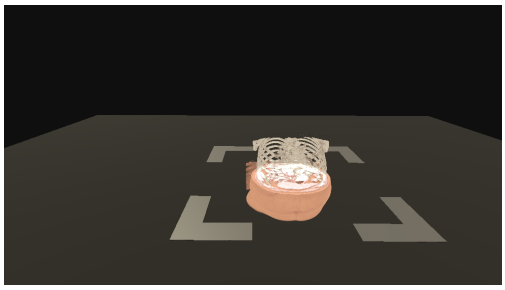


**Figure 6.1:** VR scene

## 6.4 Data exploration

As presented in *Chapter 3*, there are many ways to visualize a volume, depending on what transfer function. An option is to use isosurfacing (*Section 3.1*) to render specific isosurfaces, but as medical scans contain a lot of valuable information, it might not be possible to show all data with one isosurface. Another option is to use direct volume rendering (*Section 3.2*), which shows all volume data at the same time, the downside of this method is that the result often produces fuzzy edges which might make the visualization too unclear to be used in a medical purpose Kraus (2003). As the medical data is both very detailed and the doctors needs to be able to inspect the data with high precision, we chose to provide the additional tools to explore the data. The following sections present the tools.

### 6.4.1 Clipping plane

Looking at the data exploration techniques from previous work, presented in *Section 3.4*, the method we found best suited was the six degrees of freedom clipping plane proposed by Kniss et al. (2001). The strong argument for using a clipping plane is that it allows the user to view a 2D slice of the volume which is very similar to looking a the 2D medical scans that the doctors currently use, as stated in *Chapter 2*. Using a clipping plane with six degrees of freedom, as opposed to axis aligned planes, improves upon the 2D medical scans by allowing the user to view a slice through the patient with an arbitrary position and orientation.

The clipping plane technique proposed by Kniss et al. (2001) is made for a desktop and is controlled by a computer mouse. The user clicks and drags position and orientation handles to manipulate the plane. As mentioned in section Hardware we use six DoF controllers for this project. With the controller manipulation of the clipping plane can be made very easy by mapping the position and orientation of a controller to the clipping plane.

To allow the user to both position and orient the clipping plane, but also lock it in place without having to hold the controller in the same place while inspecting we implemented a grab mechanic. The grab mechanic known from many VR applications, as it closely resembles picking up an object in the real world. The mechanic involves moving the controller to a virtual object and pressing and holding a button to make the

object follow the relative movements of the controller. With this mechanic, the user can place the clipping plane in the desired position and keep its position after letting go.

Using the clipping plane divides the volume into three sections: bellow the clipping plane, above the clipping plane and the slice itself. Three independent sections open the possibility to apply different transfer functions to each and thereby create a more significant visual separation between each section or have each section convey different information. One example is to visualize the skin below the clipping plane, bones above the plane and separate the slice from the rest of the volume by mapping it to a different color scheme.

### 6.4.2 Transfer function presets

As mentioned at the beginning of *Section 6.4*, the design of the transfer function can take many forms, and each form visualizes different information, i.e., specific isosurfaces or a semi-transparent view of the volume. In our experience, it is not possible to define one transfer function that both cover all information of the volume and provide that information in high detail.

A solution to this problem would be to give the user the possibility to control the transfer function inside the application. A big downside of a transfer editing tool is that it would require complicated controls. Currently, we can define the transfer functions with gradient editor tool, see *Figure: 6.2*, in which the gradient is computed from a set of nodes. Nodes for color and alpha values are separated. Creating a similar tool for a VR environment would most likely require using the motion controllers as laser pointers for creating, editing and moving the notes. This solution provides the user with full control over the transfer function but would be time-consuming to use, and it is difficult achieving high precision with the the laser pointing technique. The second argument against a transfer function editing tool is that defining a valuable transfer function is not easy, e.g., for visualizing a specific isosurface the alpha values needs to make only the particular value range opaque. Thirdly the data values are very similar between scans, i.e., bones always lie within a specific intensity range, so redefining new transfer functions for each patient would be redundant.

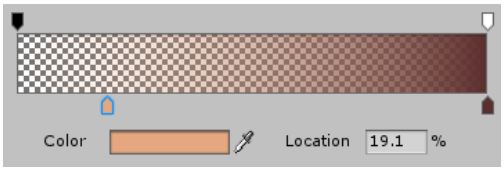


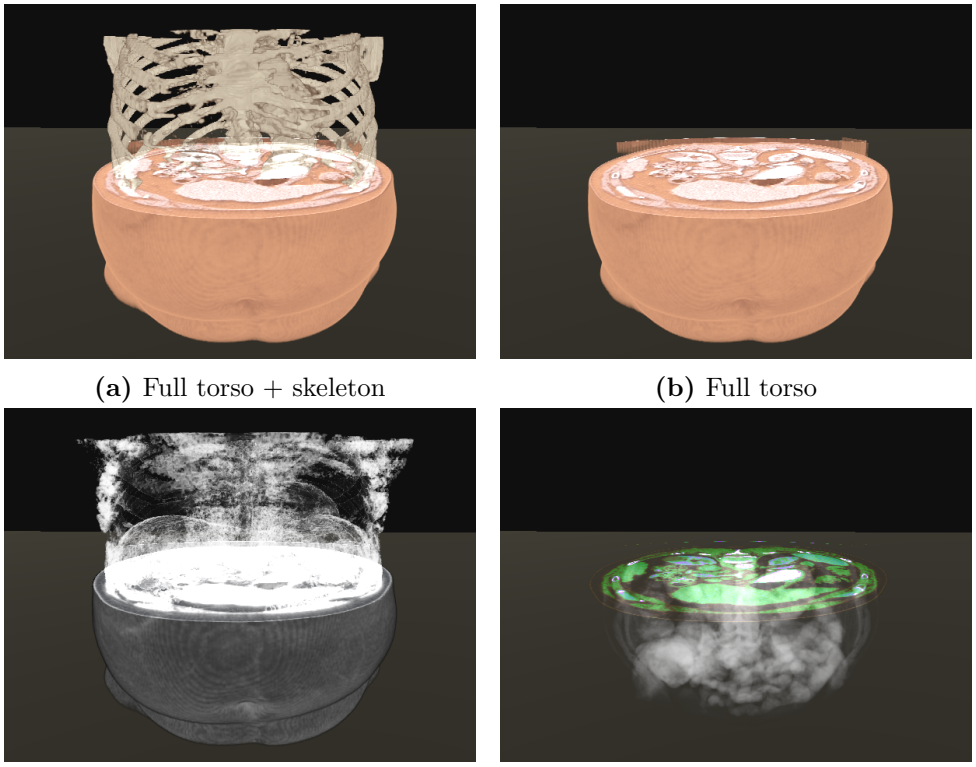

**Figure 6.2:** Unity3D Gradient editor tool

The discussed downsides of giving the user full control of the transfer function point towards another solution: providing the user with a set of preset transfer functions to chose from. Each preset can be specialized for a specific purpose, e.g., investigating the skeleton or a particular organ, and the user can quickly swap between them.s.

To show the possibilities of what kinds of information our system can visualize, we define four different transfer function presets. Each preset consists of three transfer functions as described at the end of *Section 6.4.1*. *Figure: 6.3* shows the four visualizations and the list below gives a description of each.

1. Visualizes a colored, opaque view of the torso below the clipping plane, the skeleton above the clipping plane and the slice with a slightly brighter color than the torso.
2. Similar to 1., but everything above the clipping plane is transparent, to give clear view of the slice.
3. Visualizes a greyscale, opaque view of the torso below the clipping plane, a semi transparent view above the clipping plane and slice with bright greyscale colors.

4. Visualizes a semi-transparent view of the volume below the clipping plane and the slice with a green color range, to clearly separate it from the volume. Everything above the clipping is fully transparent.



**(a)** Full torso + skeleton **(b)** Full torso

**(c)** Full torso + semi transparent view **(d)** Semi transparent view

**Figure 6.3:** The four transfer function presets designed for our system

To change between the presets, we chose to use touchpad that can be found on both the HTC Vive controllers and the Windows mixed reality motion controllers. While a finger is placed on the touchpad, the controller tracks the touch position. Additionally, the touchpad registers presses. We used the touchpad to create a radial menu, where each slice corresponds to a transfer function preset. The radial menu system gives the user the ability to chose a transfer function preset by placing his finger on the slice of the desired preset and pressing the touchpad. *Figure: 6.4* shows a screenshot of the radial menu.

### 6.4.3 Movement

An essential aspect of data exploration of a 3D object is the ability to view it from any direction. As described in Hardware, the HMD's used for this project tracks the user's head movement, both orientation, and position, which allows the user can walk around the visualization. Physical movement can be a good solution in cases where the user only has to move a little to change the viewing angle slightly. In cases where the user wants to see the 3D object from the opposite side, it can be a less attractive option, as it requires enough free physical space and is time-consuming.

To provide the user the option to view the volume from a very different angle, without having to move out of place, we implemented the same grab mechanic described for clipping plane for volume itself (*Section 6.4.1*). With the ability the grab the volume the user can position and orient it to get the desired viewing angle. The user also has to possibility to grab the volume with one hand and the clipping plane with the other to manipulate both simultaneously.

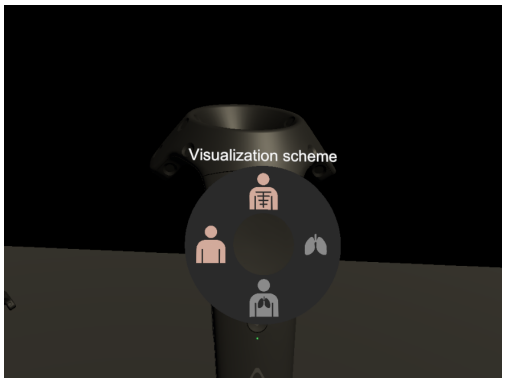



**Figure 6.4:** View of the radial transfer function preset menu

In situations where the user for some reason gets far away from the volume or clipping plane, we chose to implement the ability to teleport within the VR environment. The teleport mechanic is often used in VR applications that require the user to move around, as it allows the user to move over distances in the virtual scene without having to physically walk the same distance. To teleport the user presses and holds a button on a controller which displays an arc from that controller unto the floor of the VR scene. The intersection between the arc and the floor is the position the user will the teleported to when letting go of the controller. The user can move the controller to manipulate the arc before teleporting. For teleporting, we chose the touchpad on the controller that is not used for transfer function presets *Section 6.4.2*.

# 7 | Implementation

This chapter presents how the system design described in *Chapter 6* was implemented. *Figure: 7.1* provides a brief overview of the system, each step is explained throughout the chapter.

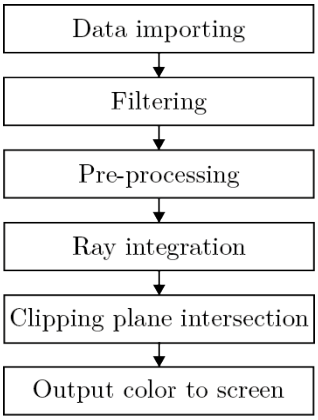



**Figure 7.1:** Pipeline of system

## 7.1 Importing data

The first step of the system is to load the scans from medical devices (CT- and MR-scanners) into a 3D Render Texture in Unity3D (Technologies, 2018a). We replaced the Data acquisition step of the volume rendering pipeline in *Figure: 5.1* with a Data importing step, as acquiring the raw data is done with medical scanners at the hospitals, and therefore the process is out of our hands. Instead, our task is to import the medical scans into our system.

As described in *Section 5.2.1*, the medical scans come in the file format DICOM. The directory structure used by the institution, for which this implementation is specifically designed, has CT-scan slices placed as individual DICOM files inside a given folder. The files are numbered in their filename, but the number does not correspond the actual slice position, so file name could not be used to determine the order in which images should be loaded into a 3D volume. The DICOM format defines tags that are included in the metadata header for each file. We use this metadata to construct the 3D volume correctly. We use the open source C# library Fellow Oak DICOM (often written as fo-dicom). This library contains functionality to read and parse DICOM metadata, load and render image data. It has an implementation specifically tailored for Unity, the game engine we are using, however, the specific part of the library which converts DICOM image data to Unity3D's texture objects is not part of the open source project has to be purchased. Luckily the scans provided by the institution uses a very simple codec. They use the DICOM transfer syntax with the UID 1.2.840.10008.1.2.1, i.e. explicit value representation little endian. This is probably the simplest format to manually parse. The voxels within one of these image files are laid out row-by-row as 16-bit individual voxel intensity values. No compression is done (DICOM Library, 2018) (National Electrical Manufacturers Association, 2018). Our implementation uses the fo-dicom library to get a reference to the array of voxels and create a texture object from it. After all images are loaded, they are sorted according to their slice location obtained from the metadata before getting passed on to the next step in the pipeline, the creation of a 3D texture.

Combining the images into a 3D render texture is performed on the GPU with a compute shader. On the CPU all images are stored in the Texture2DArray class (Technologies, 2018b), which is passed to a compute shader alongside an empty 3D Render Texture. By default, a render texture in Unity3D is 2D but setting

the *volumeDepth* attribute to the desired depth changes the render texture from 2D to 3D.

The 3D render texture is stored in the format *R8*, which is a 8-bit single channel format. One channel is sufficient as the medical scans are grey scale.

As described in *Section 2.1*, the medical scans used for testing was equal in width and height, but the depth differed. The non-cubic 3D texture results in the texture being stretched when visualized within a 1x1x1 cube. A simple solution is to scale cube, which the volume is visualized within, to fit the dimensions of the 3D texture. Scaling the cube does, however, create some complications with the computations for empty space skipping; this is further described in *Section 7.3.1*.

To produce a cubic 3D texture the texture is padded with transparent black 2D textures along the depth (z-axis). The 3D texture is padded equally on both ends to keep the volume data in the center.

## 7.2 Filtering

In the filtering step, the volume is downsampled to both reduce texture size and the required sampling frequency. *Section 5.2.4* describes how the sampling frequency for the volume visualization algorithm depends on the Nyquist frequency of the transfer function and the volume texture. Because we implement pre-integrated classification, we avoid the dependency of the transfer function, and only the texture's Nyquist frequency dictates the sampling frequency. *Section 7.3.2* describes implementation of pre-integrated classification.

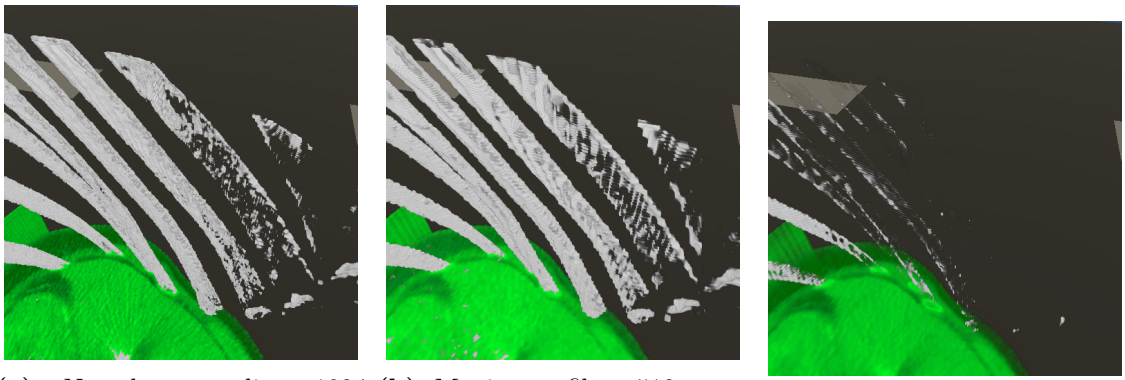
The sampling frequency has a significant impact on the performance, as it directly translates to the number of texture lookups. A straightforward approach to reducing the required sampling frequency is to downsample the volume texture as it lowers the texture's Nyquist frequency.

*Section 7.5* describes how the volume and the slice defined by the clipping plane is sampled independently. This means the data does can come from two different volumes. We took advantage of this by using a downsampled volume texture to visualize the volume itself, while using a full resolution volume texture only for the slice. This approach was chosen, as we hypothesized that the volume would mostly be used for spatial context and the slice would be used for detailed information. Going from this hypothesis, it was decided to only focus on simple downsampling schemes as the visual quality of the volume was less critical. As the downsampling method, we chose to use a maximum filter with 2x2x2 box kernel. The obvious choice for a simple downsampling scheme might be a mean filter. However, Kraus and Burger (2008) explains that the opacity accumulated when traversing through the finer voxels on the non-downsampled volume should equal the opacity of the single coarser voxel. Using a maximum filter for downsampling come closer to acheive that goal.

*Figure: 7.2* shows the result of downsampling with a maximum filter and mean filter compared to the high resolution volume texture. From the figure it is clear that the maximum filter retains the overall opacity of the volume better than the mean filter.

## 7.3 Preprocessing

Two data structures are computed during the pre-processing step. The data structures are required for the optimizations schemes: Empty space skipping and Pre-integrated classification.



**(a)** No downsampling 1024 steps per ray **(b)** Maximum filter 512 steps per ray **(c)** Mean filter 512 steps per ray

**Figure 7.2:** Downsample results

### 7.3.1 Empty space skipping

As described in *Section 5.3.1*, performing empty space skipping requires dividing the volume into blocks and storing the minimum and maximum intensity for each block. Subsequently, the stored values determine if a block should be skipped or not, and the 3D grid of non-empty blocks is used to generate a mesh. At runtime, the volume visualization algorithm can get the start and end position for ray tracing from the generated mesh. The preprocessing part of empty space skipping is computing the min/max blocks and generating a mesh from the blocks.

The min/max blocks are computed with a compute shader. Compute Shaders can efficiently run parallel processes, which significantly speeds up the process of computing the min/max blocks, as several blocks can be computed simultaneously. Each block is computed by looping over every value in the block and storing the maximum and minimum in an array. Once the values for each block is found, the full array is passed back to the CPU.

The final part of the empty space skipping preprocessing is rendering the front faces of the block grid. Quads are constructed at the faces of the blocks to render the grid. The maximum and minimum values determine if a block contains useful information, which in turn determines if quads are constructed at the faces of the block. Only faces at the perimeter of the block grid are rendered, to reduce vertex count.

The constructed mesh is applied to the volume object to make the vertex positions directly available to the shader. The vertex positions are used to compute the start positions of the rays in the ray integration stage.

The method described above of applying the block mesh directly to the volume object differs from the approach presented in *Section 5.3.1*. The approach used in other publications is to render the block mesh to an off-screen frame buffer, storing the fragment position for each fragment, and passing this information to the shader. The reason for not following this approach is that rendering to an off-screen buffer requires an additional virtual camera or at least a camera position to know from where to render the mesh. Our system should work in a VR environment, in which SteamVR (*Section 6.2*) internally uses two virtual cameras, one for each eye. This makes it not as trivial to obtain the correct camera transformation matrices for rendering the mesh.

Passing the entire mesh to the rendering pipeline and using front faces as ray starting positions does, however, come with some adverse side effects. The most prominent is overdrawing, which both negatively impacts performance and causes visual errors. The problem is that all front faces will initiate ray sampling, starting at the fragment position on the front face, when overdraw occurs, several sampling rays will be

initiated along the same viewing ray, which results in wasted computations. When the volume is rendered semi-transparent, the overdraw is visible, see *Figure: 7.3*.

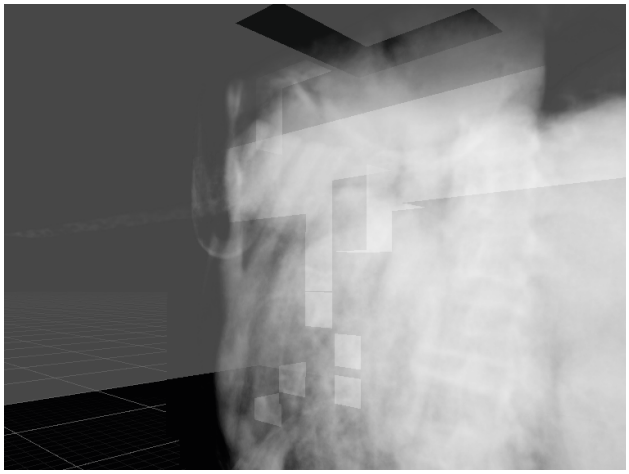


**Figure 7.3:** Empty space skipping visual errors

Another downside is that the back faces of the mesh are not utilized, which means there is no way of knowing when the rays exit the grid mesh. The rays will not be terminated before reaching they exit the boundaries of the volume. This as well causes wasted computations, as it is already known that there is only empty space outside of the grid mesh.

The process of using the information provided by the mesh is explained in *Section 7.4*.

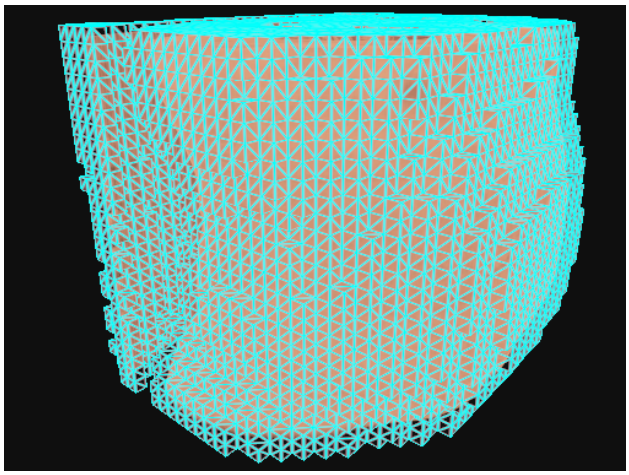


**Figure 7.4:** Empty space skipping block mesh

### 7.3.2 Pre-integrated classification

As described in *Section 5.3.3*, pre-integrated classification requires storing the pre-calculated integration of the transfer function for all possible input values in a 2D lookup table.

The 2D lookup table is a texture which is regenerated every time the transfer function is changed. The implementation should optimally allow for users to change the transfer function and see changes in perceived real time. It should optimally at least be so that it does not take a long time to calculate this lookup table. A

few optimizations are developed by Kraus (2003). One of them utilizes the fact that extinction approaches zero as ray step length decreases. This optimization is called accelerated approximative pre-integration (Kraus, 2003).

Normally we would need to integrate over the all values in the transfer function between ray step start and end positions. The resulting lookup table would contain these pre-calculated integrations, mapping from start and end value, as x and y texture coordinates, to accumulated RGBA values. Iterating over parts of, or the whole transfer function for each pixel in the lookup table can be optimized away using accelerated approximative pre-integration.

In practice constructing the pre-integration lookup table takes two steps. Firstly the cumulative sum of the transfer function is computed an stored in a 1D texture. *Equation: (7.1)* presents cumulative sum computation.

$$I(j) = \sum_{k=0}^{j} T(k) \tag{7.1}$$

where $I$ denotes the cumulative sum texture, and T denotes the transfer function, which is stored as a 1D texture as well.

*Equation: (7.2)* and *Equation: (7.3)* computes the pre-integration lookup table from the cumulative sum texture. Equation 3 is used in the situations where $x$ and $y$ are of equal value.

$$P(x, y) = (I(y) - I(x))/(y - x) \tag{7.2}$$

$$P(x, y) = I(x) - I(x - 1) \tag{7.3}$$

where $P$ denotes the pre-integration lookup table. This table can now be used in the ray integration step to reduce samples drastically in areas with a steep gradient in the scalar field – retaining the visual quality.

Another optimization mentioned by Kraus (2003) is the use of the heuristic that users will often only change a part of the transfer function at one time. This means that we could potentially skip calculations in areas that are unchanged. Since we are calculating the lookup table in a compute shader running on a modern high-end GPU, the time to generate the lookup table is minuscule and performance gain would be negligible. The complexity of implementing this optimization outweighed its benefit, especially since we ended up providing a set of predetermined transfer functions.

Since the transfer functions were predetermined, the lookup table would not need to be calculated more than once, therefore one lookup table was stored per transfer function and calculated at startup.

## 7.4 Ray integration

As described in *Section 5.1*, the color of a pixel is computed by sampling and compositing colors along the viewing ray. An overview of the process is presented in *Figure: 7.5*, this process is applied to every fragment.

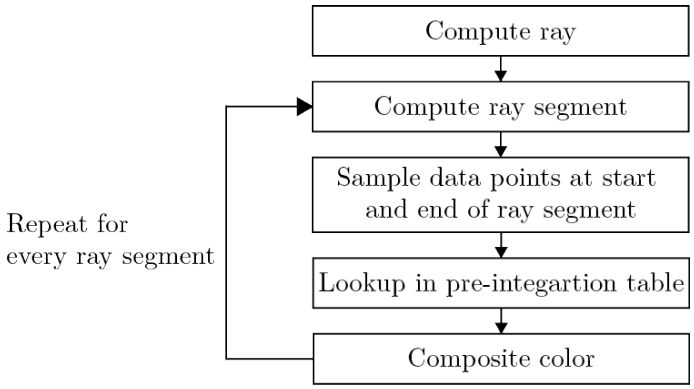



**Figure 7.5:** Ray integration procedure

Initialize a viewing ray requires direction and start position. These positions are encoded in the empty space skipping block mesh as vertex positions and interpolated by the graphics hardware before being passed to the fragment shader in charge of the ray casting. Ray directions are similarly computed per vertex, interpolated and passed to the fragment shader before being normalized. Both position and ray direction are defined with respect to the volume object, i.e. in object space. The local space of the object corresponds closely with the texture coordinate space of the volume and is therefore ideal for working in when stepping through the volume.

The number of iterations exceeds the loop unroll limit. The looping control structure (in this case a for-loop) is therefore preceded with the HLSL attribute [loop]. This makes the shader compiler create a dynamic loop instead of unrolling it into flat, consecutive instructions.

Each step along a ray is of a predetermined length. This step length is stored as a three dimensional vector pointing in the direction of the ray. This makes it so each iteration just increments the texture sampling coordinates using this vector. The length of the vector is set to $\sqrt{3}/n$ where $n$ is the sample count per ray. The reason for the $\sqrt{3}$ is that this is the maximum distance between any two vertices in a cubic mesh. Since the volume can be said to be contained within a 1x1x1 cube with respect to its texture coordinates, this step length ensures the whole of the volume, but not more, is always traced.

Each step along the ray defines a ray segment, spanning from the previous sample position to the current sample position. We denote the start and end of the ray segment as $r_{i-1}$ and $r_i$. The alpha and color contribution of the ray segment is the alpha and color value in the pre-integration lookup table at the coordinate $(s(r_{i-1}), s(r_i))$, where $s$ denotes the volume texture. This results in two texture lookups in the volume texture and one lookup in the pre-integration texture. As $r_{i-1}$ is always equal to $r_i$ of the previous loop iteration, $s(r_i)$ is stored for the following loop iteration to reduce the number of texture lookups. For the first sample position s($r_{i-1}$ is set to 0. Before performing the lookup, it is considered whether the sample position is above or below the clipping plane, to perform the lookup in the corresponding pre-integration texture.

The RGBA contribution is blended with the so-far accumulated RGBA contributions of the previous ray segments along the ray. The front-to-back compositing scheme (*Equation: (5.9)* and *Equation: (5.10)*) is used for blending the RGBA contributions. Before entering the loop, the accumulated RGBA contribution is initialized with zero in all color channels.

Two conditions can break the loop before reaching the maximum number of iterations. The first is when the ray step exceeds the boundaries of the volume. The second is the early ray termination described in *Section 3.3*. Early ray termination is executed by comparing the so-far accumulated alpha value to a value

close to one after the current ray segment contribution has been added. The loop ends when the alpha value exceeds the predefined maximum alpha value.

When any of the above conditions break the loop, the so-far accumulated RGBA value is the final color of the ray.

## 7.5 Clipping plane intersection

Traversing through the volume with a fixed step size combined with the clipping plane causes the complication that the sample points do not necessarily lie directly on the clipping plane, which results in the slice being inaccurate. Our solution to the issue is to sample the volume texture at the exact intersection between the viewing ray and the clipping plane. The clipping plane intersection is sampled independently from the ray integration loop, which means the data values can be mapped to alpha and color values with a different transfer function, as described in *Section 6.4.1*.

## 7.6 Output color to screen

The fragment output alpha and color contribution of the slice added to the ray color computed from ray integration. The contribution of the slice is zero in both alpha and all color channels if there is no intersection between the viewing ray and the clipping plane within the boundaries of the volume.

We do not have explicitly blend the colors of the background with the colors of the volume visualization, as it is handled by the graphics in the per-fragment operation blending. Blending is a fixed function stage of the pipeline, which we configure by specifying a blend equation. The blend equation specified for our shader is alpha blending.

# 8 | Evaluation

The evaluation of this project is divided into two parts: a technical evaluation that assesses the performance of the system and an evaluation by expert users.

## 8.1 Technical evaluation

As the implementation is a virtual reality application, maintaining a decent frame rate is important. We have done optimizations as described in *Section 7*. The effectiveness of these as well as the overall performance of the rendering system in different situations was measured in terms of average render time between frames and standard deviation of render time between frames.

To retain control of the experiment parameters, we created a specialized setup. The position and orientation of the volume in relation to the camera were strictly controlled and kept the same between tests. This way the amount of pixels to be filled by the graphics hardware was kept constant. We did, however, make the volume rotate. This was to emulate the fact that the volume would be viewed from different angles in a normal use case.

### 8.1.1 Results

A performance test was conducted for each of the four preset transfer functions, described in *Section 6.4.2*. Each performance test involves capturing the average render time per frame and standard deviation over 2000 frames with and without the implemented optimization schemes. The two optimization schemes evaluated in the performance test are downsampling of the volume and empty space skipping. From the two optimizations four combinations were tested:

- No optimizations scheme
- Only downsampling
- Only empty space skipping
- Both downsampling and empty space skipping

It is important to note that downsampling the volume allows for a lower sampling frequency. For the original size volume the maximum number of samples per ray is 1024, and for a downsampled volume the max number of samples per ray is 512. See *Section 7.2* for an elaboration on the topic.

The captured results for each transfer function preset is presented in *Figure: 8.1*. For a visual legend for the transfer functions see *Figure: 8.2*. The same numeric values of the results are presented in Appendix A. Finally, *Table: 8.1* shows the percentage increase in render time gained from the two optimizations schemes.

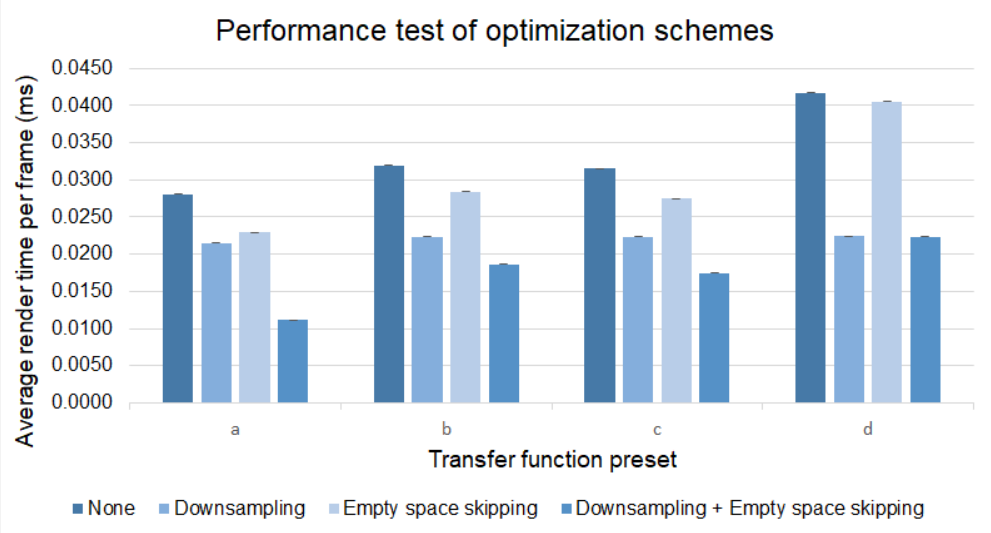
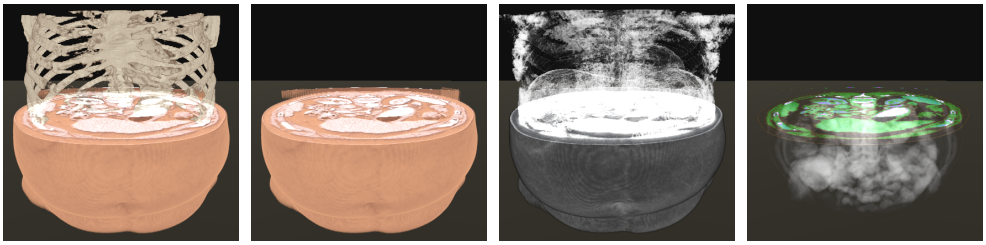



**Figure 8.1:** Performance of optimization schemes



**(a)** **(b)** **(c)** **(d)**

**Figure 8.2:** Transfer function preset legend

| **Transfer function** | a | b | c | d |
|---|---|---|---|---|
| **Downsampling** | 23.29% | 30.06% | 29.11% | 46.25% |
| **Empty space skipping** | 18.03% | 11.15% | 12.79% | 2.88% |

**Table 8.1:** Percentage of render time per frame gained from optimization schemes

### 8.1.2 Discussion

Looking at the individual performance of the four transfer function presets (*Figure: 8.1*), preset a performs with the lowest render time, corresponding to about 89.3 fps when both optimization schemes are employed. Presets b and c have similar performance, worse than preset a with fps averaging about 53.5 and 57.1 respectively. Preset d performs the worst, with an average of 44.6 fps. The result is the expected when considering the effect of early ray termination. The more opaque isosurfaces appear in the visualization, the more rays are terminated early. Because of this, preset a, that has the opaque torso below the clipping plane and the opaque bones above it, results in a lot of early terminated ray. Comparatively to preset d, that is rendered semi-transparent below the clipping plane and fully transparent above it, which results in

very few to no rays being terminated early. The second large factor is the fact that we do not have any logic that skips areas of the volume that are rendered transparent as a result of the clipping plane. This means that preset b and d that renders everything fully transparent above the clipping plane creates a worst case scenario as the sampling rays are not terminated before exiting the boundaries of the volume. Rendering the volume semi-transparent creates very similar scenario as very few, if any, rays are terminated early, which explains why presets b and c are very similar in performance.

Regarding the optimization schemes, the main observation of the results is that they both have a positive effect on the render time. Looking at the percentage increase in performance (*Table: 8.1*) downsampling contributes with largest a performance gain, between 23.3% to 46.3%, and empty space skipping adds between 2.9% to 18.0% performance.

Downsampling has the largest impact on preset d, and the least impact on preset a. With Empty space skipping it is the complete opposite. This can again be explained with early ray termination. As very few sampling rays is terminated when the volume is rendered semi- or fully transparent, each ray will reach a high number of samples, so reducing the sampling frequency is very impactful. When the volume is rendered with a lot of opaque isosurfaces, especially when the distance between the ray start position and the isosurfaces is short, only a few samples are taken along the ray before it is terminated. Therefore the sampling frequency is less impactful. Why empty space skipping is most impactful for opaque surfaces can be explained in a similar fashion. When empty space skipping is used, the distance between the ray start position and to its termination at an isosurface is short, if empty space skipping is not used the distance the ray travels before terminating is increased by a large factor. For semi-transparent visualization the viewing rays are rarely terminated before reaching the boundary of the volume structure, so the factor which by the ray travel distance is decreased when empty space skipping is used, is less than the and distance decrease for isosurface visualizations.

Looking at some of the previous work that employs similar optimization schemes, the performance of our system is significantly higher. Kruger and Westermann (2003) that implemented early ray termination and empty space skipping at best achieves a framerate of 23.4 fps, with a similarly sized volume and a smaller viewport. Stegmaier et al. (2005) and Rößler et al. (2006) that both implements volume clipping achieves framerates of 10.8 fps and 11.3fps respectively. A large reason for the big jump in performance from the previous work to our system is that the previous work we present in this project is $10 - 15$ years old, and the memory and computational power of GPUs have significantly increased within that timespan. What can be taken from this is that the computational power of GPUs has reached a level that makes real-time volume visualization a possibility.

## 8.2 Expert user evaluation

The primary purpose of our proposed system is to visualize data from medical scans, and for this reason, it was essential to get feedback from the actual professionals, i.e., doctor, surgeons, and radiologist. Only people with a medical background can take a critical look at the medical data and assess whether our visualization provides the information they need.

As mentioned in *Section 6.1*, hardware used for the user testing was a laptop with a Nvidia GTX 1060 graphics card and a Windows Mixed Reality headset. The equipment was set up in a meeting room at the hospital. Our collaborators at Aalborg university hospital tried our application, and several of their colleagues and students tested it as well.

The test participants were mostly surgeons. Most of the surgeons have experience with robot-assisted surgery, which involves six DoF controllers and stereoscopic vision. All of the participants had little to no previous experience with VR.

There was no specific procedure prepared for the expert evaluation, as the goal was to let them use it as they saw fit and get their opinion. Before the test participants tried the system, we gave them an introduction where one of us equipped the headset and presented all the functionality of the system, so the participants would know what they could do with the application and how to do it. Multiple participants would also be in the same room during the evaluation, meaning they would observe how others used it and saw how it looked both inside HMD and on a 2D screen.

### 8.2.1 Results

The results of the expert user tests were exclusively qualitative data from observations and feedback from the test participants. A summarization of the results is presented below.

Overall the reactions from the test participants were positive. The participants uttered expressions like "Amazing", "Useful" and "Quite intuitive" while trying the system.

One of the doctors described the systems as "another way to view CT scans", and added that newer generations probably would prefer to view scans in virtual reality or as 3D prints. 3D printing of medical scans is a new approach to view medical scans in 3D. The downsides of 3D printing are that they take a long time to print, and it is limited to one color mapping, whereas our system can produce volume visualization almost instantly and the transfer functions can be changed at runtime. The doctor also added that some might not find virtual reality intuitive and prefer viewing a 3D print for that reason.

One surgeon stated that our system would be useful for pre-operative planning as you adjust the scan to have the same orientation as the patient would have during the surgery. It was also stated that the volume visualization is better at locating some elements, like the a tumor and blood vessels around it, compared to the 2D scans, and that you could intuitively see what is in front and behind organs.

#### Data exploration

Regarding exploring the data, the test participants were quickly able to distinguish the different organs from each other. The experienced doctors even diagnosed the patient without being asked to do so. They identified a tumor in the left kidney and observed a previous colectomy.

Out of the four transfer functions we provided for the test, the ones with the full torso and bones, and the one with just the torso was preferred ones. They could see a purpose for all of them. To diagnose they mostly used the transfer function with a full torso and nothing above the slicing plane. The slicing plane was heavily used while diagnosing and exploring the volume. The semi-transparent transfer functions were said not to be equally as useful but could be helpful in getting an overview of organ boundaries.

#### Interaction

The majority of the test participants learned how to use the system very quickly. The system seemed exceptionally intuitive to the participants who have prior experience with the da Vinci robotic surgery system. Some participants had trouble understanding how to reach out and pick up the slicing plane and the volume and needed some guidance from us. Selecting transfer function was also troublesome for several participants, as they had to understand that the touchpad is not just one big button, but also registers the touch position.

The teleport ability was rarely used, and the participants who did use it took a little while to get used to it. One participant first thought he was repositioning the volume, and not himself.

As our volume visualization does produce some aliasing errors, we asked the participants if they noticed them. All answered that they did see some errors, but did not consider them a big problem as they were infrequent and not bound to specific points in the volume. If they consistently appeared in the same spots, it would probably have been a problem.

### Suggested improvements

During the test, a few suggestions for improvements came up. It was not possible for the users to move their head into the volume, as it would disappear. Several mentioned that it might be beneficial to be able to see the volume from inside out, as it would provide other perspectives.

It was also expressed that tools for making measurements would be necessary, as it is crucial to know the actual size of elements, and in the systems current form it is not possible to know the real size of the torso.

### Possibilities with the system

The test participants quickly saw potential in the visualizations and suggested several medical areas where the system could be useful. Afflictions like brain cancer, prostate cancer, and lung diseases were mentioned, and these would all be straightforward as they are investigated by taking CT or MR scans. A participant suggested the system could be used in combination with mammography, as mammography scans only produce a single 2D image and it is difficult to determine depth.

It was also suggested by several that several contrast phases of CT scans or multiparametric MR scans could be used with our system, which could provide even more information if the user could swap between the phases or they could somehow be combined.

A participant stated that the tool could be useful as a tool for teaching human anatomy.

We asked if the system could be useful for showing medical afflictions to patients, but they said that the system was good for doctors, as they could understand it, but it would probably require different visualizations that is more understandable to people with no medical education.

### 8.2.2 Discussion

Looking at the results of the expert user test, the reactions and responses to our system were positive. The big finding was the fact that the user was able to diagnose the patient and even recognize a previous operation performed on the patient, seemingly without much trouble. This proves that the system works, as it would not have much use if doctors could not use it to identify medical issues. This finding coupled with the positive first-hand impressions and the fact the user suggested different medical areas where the system could be used, means that the expert user not only found the system useful but also saw potential with it.

Some of the more experienced doctors talked about that it might be the newer generations that would take advantage of this kind of system, as they are used to inspecting 2D scans. In the same vein, it was said that some might prefer to 3D print an organ, as might find a physical object more intuitive than virtual reality interaction. These findings show that introducing a virtual system like ours into to the workflow at a hospital would come with a learning curve, as not everyone has experience with VR. Additionally, medical personnel might still prefer 2D scans as it is what they have been educated in using and have the most experience with. It is possible that the medical education would need to include our VR system before it would be a part of doctors workflow.

The system did contain a few technical issues, namely aliasing errors and the users not being able to move their heads inside the volume, without it disappearing. The issues did not hinder the users in diagnosing the patient, but regarding seeing the volume from the inside was mentioned as a nice feature to have. The fact that the users were still able to use the system without much trouble does, however, indicate that it is not pressing issues.

A missing element that is a 'need-to-have' of the system is a measurement tool, to be able to see real sizes of organs, as this is vital when planning an operation. A possible solution could be to calculate the distance between user-defined 3D points or place objects in the scene that the user knows the physical dimensions of.

Another possible improvement is regarding the teleport ability. One of the test participants who tried it, at first thought he was repositioning the volume instead of himself. Even though the participant learned to used the teleport quickly thereafter, it might be the better solution to replace the teleport ability with the ability to center the volume and clipping plane in front of the user with the press of a button. There is no other benefit of repositioning yourself in the VR scene other than getting close to the volume and clipping plane, so it might be easier and more intuitive to move everything to a good position in front of you, than moving around the scene to get the objects. An alternative could be to give the users a sort of extended arm ability, to grab objects at a distance with a laser pointer from the controller. Being able to pick up objects without directly colliding the virtual controller with the object, might also help the users who had a hard time getting used to that they needed to reach out to grab the volume and clipping plane. The extended arm could have a pull mechanism, to handle the situations where the user gets too distant from the objects.

A clear part of the future work for this project is developing the system to incorporate and utilize CT-scans with multiple phases and multi-parametric MR scans. This improvement were suggested by multiple test participants as a method to create more extensive visualizations. An approach would be to merge the scans of different phases or parameters and for example separate veins from organs with different transfer functions. There might, however, be some complications with the scans each phase/parameter not being completely aligened, if the patient moved between the scans. If it is not possible to merge the scans, another solution would be to let the user swap between the scans, similar to how they swap between transfer functions in the current system.

# 9 | Conclusion

An analysis of the proposed problem and similar systems laid the foundation for a list of requirements that our proposed solution should fulfill. From these requirements, a system was designed, implemented and evaluated. The system was designed to work in a virtual reality environment and contains a volume visualization of medical scan data and tools to explore the data. The system underwent a technical evaluation to analyze the performance of the system, and an expert user evaluation to investigate whether the system could provide the users with the necessary information.

The volume visualization was implemented with a fragment shader in Unity3D. The algorithm is a discretization of the volume rendering integral and employs several optimization schemes for enhancing performance and visuals. Transfer functions map the data values from the medical scans to colors and opacities. For this project four different transfer functions were defined, to show the capabilities of the system. The technical evaluation captured the rendering time per frame for each transfer function. The captured framerates were between 89.3 to 44.6 frames per second, depending on the transfer function. The more opaque surfaces the visualization contains, the better the performance.

The expert user evaluation of the system overall gave positive results, as the test participants were very impressed by the system and were able to set a correct diagnosis for the scanned patient. It varied how intuitive the participants found the system. Some understood how to use it right away, while others needed additional guidance from the test facilitators. The evaluation also unveiled features that the system lacked, such as being able to measure organs. Another finding is that implementing a system like ours into the workflow at the hospital would take time as medical personnel currently are used to inspecting 2D medical scans and therefore an adaption period would be required.

The problem statement, presented in *Chapter 2* was investigated throughout the project. A volume rendering algorithm was proposed to visualize 3D reconstructions from 2D medical scans, and virtual tools were proposed alongside to help users extract data from it. Expert users were able to use the system successfully and find the critical information. For users who did not find the system intuitive to use right away, better instructions or tools that fit their style of interaction better can be implemented.

The technical and expert user evaluations uncovered improvements that will benefit the system, and areas that would be interesting to investigate for future development. These are discussed in *Chapter 10*.

# 10 | Future work

Out system showed good potential to fulfill the problem described in *Chapter 2*, but there are several aspects that could be improved and areas the could be interesting to investigate.

## 10.1 Rendering improvements

An element of the rendering process that could be significantly improved is the empty space skipping optimization. As described in *Section 7.3.1*, the current solution of rasterizing the front faces of the grid mesh and using fragment positions as starting positions for sampling rays results in both performance and visual problems. The current implementation was chosen to circumvent working with separate off-screen buffers for each eye. However, storing the depth of empty space skipping mesh, both front and back faces, in off-screen buffers would solve the problems of overdraw, as any occluded front faces would not be considered as starting positions for the ray integration. The challenge is to first, from each eye position, render the depth of the mesh to two buffers, and secondly provide the correct buffer to each eye before rendering the volume. A solution might be to implement the volume rendering algorithm as an image effect, to make it possible to control when everything is rendered.

Another solution to improve the current empty space skipping implementation could be to perform sorting on the mesh, to ensure occluded front faces were not rendered to eliminate the overdraw.

An approach for improving performance would be to take transfer functions into account when computing the empty space skipping blocks. Currently, empty space is determined by a threshold to skip areas of the volume with no data, but depending on the transfer function parts of the volume might be rendered transparent, e.g., when only rendering the skeleton. If the empty space skipping grid only cover areas, that is not going to rendered transparent, even more, unnecessary texture lookups could be avoided.

To improve the visual quality of the rendered isosurfaces shading operations could be added to visualization. If the surfaces interact with the light, e.g., by casting shadows, the volume might look more realistic, and it possibly could improve depth cues.

## 10.2 User interaction

The evaluation uncovered several possible improvements and alternatives to the interactive tools the current system provides.

One of the more important features to develop is tools to make size measurements of the volume and get the real sizes of elements like organs and blood vessel and measure distances, e.g., from the skin an organ.

Another interaction that could be implemented differently is the actions of positioning yourself relative to the volume. In the current system, users can manipulate objects by picking them up, and they can teleport around in the scene to reposition themselves. A possibly more intuitive to the teleport ability could be the ability to reposition the volume and clipping plane to a position right in front of the user. Another alternative could be to extend the grabbing mechanic with a laser pointer from the controller to give the user the ability to grab objects at a distance.

It might also be necessary to implement options to move the clipping plane, other than picking it up and repositioning it. The evaluation showed that the doctors were able to diagnose the patient with the current solution, but in real sections, they might need to view an exact slice, e.g., for measurements. It might be

difficult to place the clipping plane precise enough due to hands shaking and small inconsistencies in the tracking of the motion controller. Solutions to this area could be to smooth the clipping plane movements or an option to move the plane in small increments.

## 10.3 Use in robotic surgery

As described in *Chapter 1*, the goal behind this project is to be the first step in implementing volume visualization of medical scans with *da Vinci* robotic surgery system for minimally invasive surgery, with the intention of providing the surgeons with the information from medical scans during operation. To fulfill this goal future work should focus on creating an interface between our current system and the robotic surgery system to be able to superimpose the volume rendering on top of the video feed from the surgery system. The future work in this area would also have to deal with the issue of aligning the volume data with the video data. When working with live surgery, the performance of the systems is also of very high importance. If there is a significant delay from the surgeons' input to he sees the tools move on his display, it will increase the difficulty of the surgery.

A part of the creating the interface would also be to give the user tools to manipulate the volume, e.g., positioning the volume, if it is not automated and changing transfer function.

## 10.4 More data

During the user evaluation it came up several times that our system could incorporate more advanced medical scans, i.e., CT-scans with multiple phases and multi-parametric MR-scans. These more advanced scans produce several stacks of scans of the same patient, each with a different purpose, such as highlighting veins or organs. One issue of using the scans is that they might not be aligned entirely if the patient moved between the scans. One solution to use multiple stacks of scans would be to align the scans in a preprocessing step to create a single volume containing all data, that would make possible to separate blood vessels from organs easily, e.g., by assigning them with different colors. A more straightforward solution could be to create separate volumes from the scans and give the user the ability to swap between them.

Several test users also suggested different medical areas in which our system could be useful, such as brain, lung or breast diseases. It would be interesting to expand the system to accept scans of other types than CT. It would likely also be necessary to adjust the transfer function to fit scans of other parts of the body.

# Bibliography

Bruckner, S. and Groller, M. E. (2006). Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1077–1084.

DICOM Library (2018). DICOM Library - About DICOM available transfer syntax.

Drebin, R. A., Carpenter, L., and Hanrahan, P. (1988). Volume Rendering. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88, pages 65–74, New York, NY, USA. ACM.

Ebert, D. and Rheingans, P. (2000). Volume illustration: non-photorealistic rendering of volume models. In *Proceedings Visualization 2000. VIS 2000 (Cat. No.00CH37145)*, pages 195–202.

Ferrand, G., English, J., and Irani, P. (2016). 3d visualization of astronomy data cubes using immersive displays. *arXiv preprint arXiv:1607.08874*.

Hadwiger, M., Kniss, J. M., Rezk-salama, C., Weiskopf, D., and Engel, K. (2006). *Real-time Volume Graphics*. A. K. Peters, Ltd., Natick, MA, USA.

Hadwiger, M., Sigg, C., Scharsach, H., Bühler, K., and Gross, M. (2005). Real-Time Ray-Casting and Advanced Shading of Discrete Isosurfaces. *Computer Graphics Forum*, 24(3):303–312.

Kaufman, A. E. (1999). Introduction to volume graphics. In *SIGGRAPH*, volume 99, pages 24–47.

Kniss, J., Kindlmann, G., and Hansen, C. (2001). Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. pages 255–562. IEEE.

Kraus, M. (2003). Direct volume visualization of geometrically unpleasant meshes.

Kraus, M. and Burger, K. (2008). Interpolating and Downsampling RGBA Volume Data. *13th International Fall Workshop Vision, Modeling, and Visualization 2008, VMV 2008*, pages 323–332.

Kruger, J. and Westermann, R. (2003). Acceleration techniques for GPU-based volume rendering. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 38. IEEE Computer Society.

LaMar, E., Hamann, B., and Joy, K. I. (1999). Multiresolution Techniques for Interactive Texture-based Volume Visualization. In *Proceedings of the Conference on Visualization '99: Celebrating Ten Years*, VIS '99, pages 355–361, Los Alamitos, CA, USA. IEEE Computer Society Press.

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

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

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

National Electrical Manufacturers Association (2018). DICOM Standard.

Parker, S., Parker, M., Livnat, Y., Sloan, P.-P., Hansen, C., and Shirley, P. (2005). Interactive Ray Tracing for Volume Visualization. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA. ACM.

Roth, H. R., Lu, L., Seff, A., Cherry, K. M., Hoffman, J., Wang, S., Liu, J., Turkbey, E., and Summers, R. M. (2014). A New 2.5d Representation for Lymph Node Detection using Random Sets of Deep Convolutional Neural Network Observations. *Medical image computing and computer-assisted intervention : MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention*, 17(0 1):520–527.

Rößler, F., Tejada, E., Fangmeier, T., Ertl, T., and Knauff, M. (2006). GPU-based multi-volume rendering for the visualization of functional brain images. In *SimVis*, volume 2006, pages 305–18.

Stegmaier, S., Strengert, M., Klein, T., and Ertl, T. (2005). A simple and flexible volume rendering framework for graphics-hardware-based raycasting. In *Fourth International Workshop on Volume Graphics, 2005.*, pages 187–241.

Taerum, T., Sousa, M. C., Samavati, F., Chan, S., and Mitchell, J. R. (2006). *Real-Time Super Resolution Contextual Close-up of Clinical Volumetric Data.* The Eurographics Association.

Technologies, U. (2018a). Unity - Scripting API: RenderTexture.

Technologies, U. (2018b). Unity - Scripting API: Texture2darray.

Van Gelder, A. and Kim, K. (1996). Direct volume rendering with shading via three-dimensional textures. In *Volume Visualization, 1996. Proceedings., 1996 Symposium on*, pages 23–30. IEEE.

Viola, I., Feixas, M., Sbert, M., and Groller, M. E. (2006). Importance-Driven Focus of Attention. *IEEE Transactions on Visualization and Computer Graphics*, 12(5).

Weiskopf, D. (2006). Visualization of 3d Scalar Fields. In *GPU-Based Interactive Visualization Techniques*, pages 11–79. Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-540-33263-3_2.

Šrámek, M. (2006). 20 Years of Volume Rendering. In *Proceedings of the 22Nd Spring Conference on Computer Graphics*, SCCG '06, pages 7–16, New York, NY, USA. ACM.

# Appendices

# A | Technical evaluation results

Listed below are the captured average render time per frame and the standard deviation for the optimization schemes downsampling and empty space skipping. Each table contains the results for one transfer function preset.

| **Optimization scheme** | **ms per frame** | **SD** |
|---|---|---|
| None | 0.0280 ms | 2.167E-4 ms |
| Downsampling + Empty space skipping | 0.0112 ms | 0.162E-4 ms |
| Downsampling | 0.0215 ms | 0.663E-4 ms |
| Empty space skipping | 0.0230 ms | 0.856E-4 ms |

**Table 10.1:** Transfer function 1: full torso and skeleton

| **Optimization scheme** | **ms per frame** | **SD** |
|---|---|---|
| None | 0.0320 ms | 2.653E-4 ms |
| Downsampling + Empty space skipping | 0.0187 ms | 1.180E-4 ms |
| Downsampling | 0.0223 ms | 0.101E-4 ms |
| Empty space skipping | 0.0284 ms | 2.248E-4 ms |

**Table 10.2:** Transfer function 2: full torso

| **Optimization scheme** | **ms per frame** | **SD** |
|---|---|---|
| None | 0.0315 ms | 2.590E-4 ms |
| Downsampling + Empty space skipping | 0.0175 ms | 1.247E-4 ms |
| Downsampling | 0.0223 ms | 0.100E-4 ms |
| Empty space skipping | 0.0275 ms | 2.108E-4 ms |

**Table 10.3:** Transfer function 3: Full torso and semi-transparent view

| **Optimization scheme** | **ms per frame** | **SD** |
|---|---|---|
| None | 0.0417 ms | 3.335E-4 ms |
| Downsampling + Empty space skipping | 0.0224 ms | 0.156E-4 ms |
| Downsampling | 0.0224 ms | 0.174E-4 ms |
| Empty space skipping | 0.0405 ms | 2.546E-4 ms |

**Table 10.4:** Transfer function 4: semi transparent view