Iso-Surface Volume Rendering

Speed and Accuracy for Medical Applications

Marco Bosma

Bosma, Marco Klaas
Iso-Surface Volume Rendering: Speed and Accuracy
for Medical Applications
Proefschrift Universiteit Twente, Enschede. -Met index, lit. opg.
-Met samenvatting in het Nederlands

ISBN 90-365-1397-5

Eerste Uitgave 2000
Druk: PrintPartners Ipskamp, Enschede

ISO-SURFACE VOLUME RENDERING: SPEED AND ACCURACY FOR MEDICAL APPLICATIONS

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. F.A. van Vught,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 20 oktober 2000 te 16.45 uur.

door

Marco Klaas Bosma

geboren op 28 oktober 1970

te Losser

Dit proefschrift is goedgekeurd door:

Prof. Dr.-Ing. O.E. Herrmann (promotor) en
Ir. J. Smit (assistent-promotor)

SUMMARY

This thesis describes the research on the accuracy and speed of different methods for the visualization of three-dimensional (3D) sets of (measured) data. In medical environments, these 3D datasets are generated by for instance CT and MRI scanners. The medical application makes special demands on the visualization methods.

Medical application of 3D visualization methods asks for high accuracy. For practical application, the speed of the visualization should also be high, preferably real-time (25 images per second or higher). Because most 3D visualization methods are very computational intensive, real-time high quality visualization is not (yet) possible in software. As a consequence, many different visualization methods have been developed that either sacrifice the speed for quality or sacrifice the quality to obtain high speed.

To meet both the demand for high speed as well as the demand for high quality, a visualization method has been developed that reduces the number of computations drastically without affecting the image quality. This visualization method, called Iso-Surface Volume Rendering, is able to find and visualize surfaces in a three-dimensional dataset with high speed (approximately 5 images per second).

To be able to compare the accuracy of different visualization methods, the signal theoretic background of the visualization methods and the data acquisition process for CT and MRI scanners is investigated. Using this information, the accuracy of the approximation of the location of a measured step edge is investigated. This accuracy only depends on the parameters used for acquisition, and is hence independent of the visualization method used. It is possible to approximate the location of the step edge using a surface at which the measured data has constant

value. This value is called the iso-value and the corresponding surface is called the iso-surface.

Next, the accuracy at which the different visualization methods are able to determine and visualize the location (and shape) of this iso-surface is investigated. Some methods are not able to visualize a surface, while the accuracy of other methods is limited because the data values are only determined at some fixed locations. The accuracy of these methods can be improved by decreasing the distance between these points. This will however also increase the calculation time dramatically. The Iso-Surface Volume Rendering method uses iteration to find the location of the iso-surface with a very high accuracy, without significantly affecting the calculation time.

Finally, a description of the different optimizations used to increase the speed and some examples of application of the Iso-Surface Volume Rendering on medical data are given.

SAMENVATTING

In dit proefschrift wordt het onderzoek beschreven naar de nauwkeurigheid en snelheid van verschillende methodes voor de visualisatie van drie-dimensionale (3D) sets van (meet-) waarden. In de medische wereld worden deze 3D datasets gegenereerd door bijvoorbeeld CT en MRI scanners. De medische toepassing stelt bijzondere eisen aan de visualisatie methodes.

Voor de toepassing van 3D visualisatie technieken in de medische wereld is een hoge nauwkeurigheid vereist. Voor de praktische toepasbaarheid moet de snelheid van deze visualisatie hoog zijn, liefst real-time (25 beelden per seconde of meer). De meeste 3D visualisatie methodes zijn echter zeer rekenintensief, waardoor real-time visualisatie met hoge kwaliteit in software (nog) niet mogelijk is. Daarom zijn veel verschillende visualisatie methodes ontwikkeld die of een goede kwaliteit hebben met een lage snelheid, of een hoge snelheid met een lage kwaliteit.

Om aan beide eisen tegemoet te komen is een visualisatie methode ontwikkeld die het rekenwerk drastisch reduceert zonder de kwaliteit aan te tasten. Deze visualisatie methode, genaamd Iso-Surface Volume Rendering, is in staat om met vrij hoge snelheid (plm. 5 beelden per seconde) oppervlakken in een drie-dimensionale set van meetwaarden te berekenen en visualiseren.

Om de nauwkeurigheid van verschillende visualisatie methodes met elkaar te kunnen vergelijken, wordt eerst de signaal-theoretische achtergrond van de visualisatie methodes en de data acquisitie door middel van CT en MRI scanners bekeken. Met behulp van deze informatie wordt gekeken hoe nauwkeurig de locatie van een gemeten stapovergang kan worden benaderd. Deze nauwkeurigheid is alleen afhankelijk van de instelling van de scanner en is dus onafhankelijk van de gebruikte visualisatie methode. Het blijkt vrij goed mogelijk om de locatie van een 3D

stapovergang te benaderen met behulp van een oppervlak waarop de meetwaarde constant is. Deze meetwaarde wordt de iso-waarde genoemd en het bijbehorende oppervlak het iso-oppervlak.

Vervolgens wordt gekeken hoe nauwkeurig de verschillende visualisatie methodes de locatie (en vorm) van dit iso-oppervlak kunnen bepalen en visualiseren. Sommige methodes zijn niet in staat om een oppervlak te visualiseren. Andere methodes zijn beperkt in de nauwkeurigheid doordat de meetwaardes slechts op enkele vaste locaties worden bepaald. De nauwkeurigheid van deze methodes kan worden verhoogd door de afstand tussen deze punten kleiner te maken, maar dit resulteert tevens in een sterke toename van de rekentijd. De ontwikkelde Iso-Surface Volume Rendering methode maakt gebruikt van iteratie om de locatie van het oppervlak met vrij te kiezen nauwkeurigheid te bepalen zonder dat daarbij de rekentijd sterk wordt beinvloed.

Tot slot wordt een beschrijving gegeven van de verschillende optimalisaties die gebruikt worden om de snelheid te verhogen en wordt een aantal voorbeelden gegeven van toepassingen van de Iso-Surface Volume Rendering methode op medische data.

ACKNOWLEDGEMENTS

The research described in this thesis would not have been possible without the support of many people. I am very grateful to all the persons who have contributed directly or indirectly to my work and to this thesis.

First of all I would like to thank Professor Otto Herrmann for being my promotor. I would like to thank Jaap Smit for being my assistant promotor and for being my supervisor over the last four years. We have had many long discussions, that have lead to many new ideas.

I would also like to thank the other members of my Ph.D. commission: Prof. dr. ir. F.J.A.M. van Houten, Dr. ir. F.A. Gerritsen, Prof. dr. ir. P.P.L. Regtien and Prof. dr. G. Zimmermann.

Furthermore, I would like to thank all colleagues of the laboratory Signals & Systems - Network Theory (former laboratory for Network Theory and VLSI Design) for their help during my research. Especially I would like to thank my former room mate Frits de Bruijn for the many interesting discussions both on my research as well as on many other topics.

My research was sponsored by the EVM Advanced Development group of Philips Medical Systems Nederland BV in Best, The Netherlands. I would like to thank all members of this group for their support on this research. I especially would like to thank Steven Lobregt, who was the supervisor of this project, for his help during the research and for his comments on my thesis.

Furthermore, I would like to thank some people working at other departments of Philips Medical Systems: Jacques den Boer and Miha Fuderer for their information about MRI scanners and help with measurements on an MRI scanner, Thijs Adriaansz and Jan

Timmer for their information about CT scanners and help with measurements on a CT scanner and Jeroen Terwisscha van Scheltinga for giving some useful comments on my thesis.

And last but not least, I would like to thank my parents, family and friends for their support during the last years.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

In this first chapter, an overview of the problem area will be given. Furthermore, this chapter will give an overview of the outline of this thesis.

1.1 Problem area

The projection of three dimensional data into two dimensional images requires a huge amount of computations. A lot of research has been done in the last years with the goal to obtain interactive (multiple images per second) to real-time (25-30 images per second) rendering speed. One approach to obtain a high rendering speed is by implementing the algorithms in hardware. The main disadvantage of this approach is the required time to develop this hardware and the impossibility to change the algorithm once the hardware is built. A lot of research has therefore been done on optimization of visualization algorithms in software. This has increased the rendering speed drastically because of both the improved algorithms as well as the increased speed of general purpose processors.

To achieve low computation times, many visualization algorithms sacrifice the image quality for rendering speed. In most algorithms, the volume data is re-sampled. As the computation time in these algorithms depends heavily on the chosen re-sampling frequency, the re-sampling frequency is often chosen as low as possible. This will however significantly affect the perceived image quality by introducing re-sampling artefacts.

When three-dimensional visualization algorithms are used for medical applications, these rendering artefacts are intolerable. Although interactive to real-time speed is also important for a visualization algorithm to be useful in medical applications, image quality should not be sacrificed to obtain this goal.

To be able to find a visualization algorithm that is able to achieve the highest possible image quality, some knowledge about the signal-theoretic background of the medical acquisition device as well as the digital image processing steps in visualization algorithms is necessary.

1.2 Outline of this thesis

First, in the following chapter, some aspects of the signal theoretical background that is used to investigate data acquisition and volume rendering will be discussed. In this chapter, the theory of ideal reconstruction of sampled data will be discussed as well as

more practical reconstruction methods. Furthermore, an overview of different methods for calculation of gradient fields in a discrete three-dimensional data field and an overview of methods to find an edge in a sampled data field will be given.

In Chapter 3, an overview of some frequently used volume visualization methods (especially for medical applications) will be given. Chapter 4, will give some more detail on one of these visualization methods called volume rendering. In this chapter, an overview of the basic volume rendering algorithms as well as some improved implementations will be given.

Chapter 5 will give a definition of an iso-surface in a discrete three-dimensional data field. In this chapter, also a new volume rendering algorithm called iso-surface volume rendering is described. This algorithm aims to visualize the iso-surfaces described.

As mentioned earlier, a thorough understanding of the signal theoretic aspects of data acquisition is necessary. Chapter 6 will describe these aspects of the two medical acquisition devices that are most often used as a source for volume rendering applications.

In Chapter 7, the accuracy of two steps in the signal processing chain is investigated. First, the accuracy of estimating a step edge location after filtering and sampling will be investigated. Hereafter, a spatial error bound for the visualization of iso-surfaces in three-dimensional data will be introduced. The volume rendering algorithms described before will be evaluated using this spatial error bound.

Chapter 8 contains some details about an efficient implementation of the iso-surface volume rendering algorithm. This chapter will describe some methods to improve the speed of this algorithm with minimal loss of image quality.

In Chapter 9, some examples of applications of the iso-surface volume rendering to real medical (patient) data will be given.

Chapter 10 will present some come conclusions and topics for future research.

CHAPTER

2

SIGNAL THEORETIC BACKGROUND

To be able to investigate the accuracy of different visualization techniques, it is important to have a good insight in the basic principles of the different signal processing steps used. This chapter will describe the signal theoretic background of the main signal processing steps that are used in the whole signal processing chain, from data acquisition to visualization.

2.1 Ideal reconstruction

The ideal reconstruction of sampled 1D data assumes that the original input signal is band-limited with some cut-off frequency f_c. The frequency at which the signal should be subsequently sampled should be at least twice as high as the highest possible signal frequency f, to allow an ideal reconstruction of the input data. This criterion will be referenced as the Nyquist frequency domain criterion [OPP75]. This criterion can be extended to the reconstruction of 3D (medical) data as well.

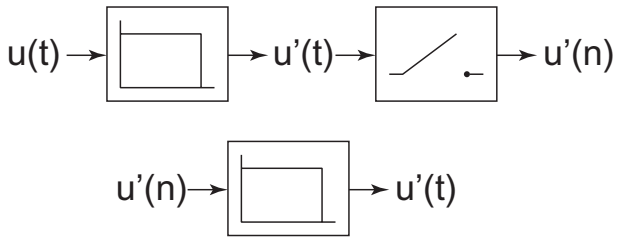


Figure 2.1 Sampling and reconstruction steps in a typical one-dimensional digital signal processing chain

In Figure 2.1, the sampling and reconstruction steps in a typical digital signal processing chain are shown. Characteristic for systems for ideal reconstruction of one dimensional data is the presence of an ideal analog low-pass filter which cuts off all frequency components which, when this analog filter would be omitted, would fold back during sampling. In this way, a non band-limited signal u(t) is filtered to obtain a band-limited signal u'(t) with cut-off frequency f_c. According to the Nyquist criterion, this band-limited signal can now be sampled without folding when the sample frequency is at least $2 \cdot f_c$. By using an ideal low-pass filter with the same cut-off frequency as the pre-filter, the band-limited signal u'(t) can be reconstructed from this sampled data without error. The original non band-limited signal u(t) can however not be reconstructed.

The reconstruction of scanned objects from scanner data using volume rendering techniques differs in many ways from this traditional one dimensional signal processing problem. The latter technique mainly uses linear, time invariant filters, whereas the processing step in volume rendering is essentially place variant

and non-linear as far as the calculation of the opacity and the shading of the surface concerns.

2.2 Data acquisition

In 3D scanners, like for instance the CT and MR scanners, there is no explicit pre-filter. Instead, the acquisition process in the apparatus measures a physical quantity in a region of interest. The intrinsic properties of these scanners can be characterized by a continuous point-spread function (PSF), which is more or less independent of the position of actual measurement.

While this point-spread function is continuous, the data is only acquired at discrete locations. Due to this sampling mechanism, a 3D array of numbers representing measured physical quantities on a 3D grid is obtained.

The data acquisition process can also be described by calculating the convolution of the continuous point-spread function with the input data, after which the data at discrete locations can be calculated by sampling the resulting continuous function. Using this approach, the data acquisition can be described in the frequency domain by multiplication of the Fourier transform of the point-spread function, also called the modulation transfer function (MTF), with the Fourier transform of the data, after which the resulting signal is sampled. The Nyquist criterion is hence met when this frequency domain product is negligible at frequencies higher than half the sampling frequency.

Figure 2.2 shows a 2D equivalent example of this data acquisition process. The letter E has been acquired with a CCD camera. The left image shows the original object, while the right image shows the acquired image. Each sample (square) in this image has a grey-value that corresponds to the amount of the original object that is present in the surrounding of the sample location. Inside the object, the samples are black while outside the samples are white. At the edge of the object, the acquired value has a grey-value that depends on the distance to the border of the object. In 2D and 3D imaging, the acquired samples are often depicted as grey pixels. For this reason, the sample values are often referred to as grey-values.

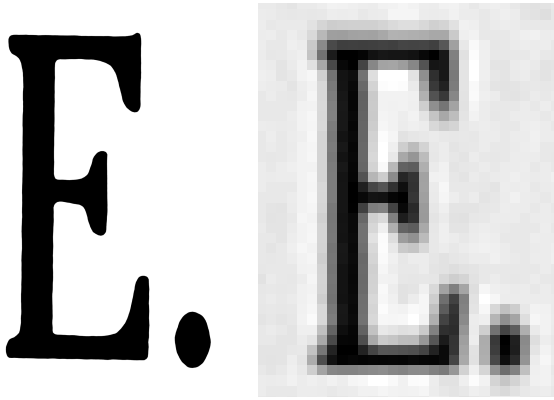


Figure 2.2 A two-dimensional equivalent example of the data acquisition process showing the original (left) and acquired object

It can also be seen in the right image of Figure 2.2 that the acquired grey-value of the background is not pure white. This is caused by noise in the acquisition device. Both the point-spread function as well as the noise properties of two medical acquisition devices will be described in a chapter about the signal theoretic aspects of data acquisition in these medical scanners.

2.3 Reconstruction and interpolation

Sharp low-pass filters have in general a rather long impulse response. The ideal low-pass filter has an impulse response which has a sin(x) / x shape, as shown in Figure 2.3.

As described in section 2.1, an ideal low-pass filter can be used to reconstruct a properly sampled band-limited signal without error. This filter is however unusable in practical applications. Because the filter has an infinitely long impulse response it can only be applied to infinitely long input signals. In most 2D and 3D applications however, the input signal is very short. In 1D applications, the input signal may be much longer. This is especially true for audio signals. The non causality of this filter however prohibits real-time processing because also signals in the future are necessary. Using a non causal filter for real-time processing is only possible by introducing a delay as long as half the width of the filter and hence with filters with a finite length.

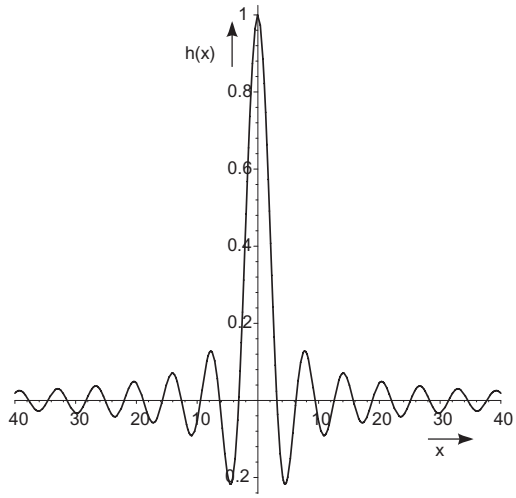


Figure 2.3 Impulse response of an ideal low-pass filter

Although it is possible to use a reconstruction filter with a long impulse response, this will generally not be desirable due to the high computational cost. When for instance a reconstruction filter with a width of sixteen samples is used, this would require 4096 samples in a 3D reconstruction filter.

As the point-spread function of the acquisition device is generally not an ideal low-pass filter and the Nyquist criterion is often not met, it is also not useful to use an ideal low-pass filter for the reconstruction of the data. To reduce the complexity, especially in 3D applications, a reconstruction filter with a very short impulse response is desirable.

The reconstruction filter with the shortest possible impulse response is the nearest neighbor interpolation function. This function assigns the value of the nearest sample to a non sample location. This requires only one input sample and no computations in 1D, 2D as well as 3D applications. The discontinuity of this reconstruction filter will however result in clearly visible artefacts when used in 2D or 3D visualization algorithms.

The simplest and smallest continuous reconstruction filter is the linear interpolation function. In one dimension, this function uses two samples and approximates the intermediate values by a straight line.

$$L(a, b, x) = a(1-x) + bx = a + (b-a)x \tag{2.1}$$

Where a is the value of the first sample, b the value of the second sample and x the location between the two samples ($x \in [0, 1]$).

The linear interpolation function is due to the low cost and the continuity an often used reconstruction filter in one, two and especially three dimensional applications.

The definition of the nearest neighbor interpolation function can also be applied to higher dimensional fields. In 2D imaging applications for instance, the reconstructed value is equal to the value of the nearest pixel. The linear interpolation however is not linear in all directions in 2D and 3D. This is caused by the repeated application of the linear interpolation in each dimension. In 2D, two interpolations are used to interpolate in the x-direction, while a third interpolation is used to interpolate these two interpolated values in the y-direction.

$$\begin{aligned} L2(a, b, c, d, x, y) &= L(L(a, b, x), L(c, d, x), y) \\ &= (a(1-x) + bx)(1-y) + (c(1-x) + dx)y \end{aligned} \tag{2.2}$$

While this function is linear in the x- and y-directions, on the diagonal where $y=x$ this yields:

$$\begin{aligned} &L2(a, b, c, d, x, x) \\ &= (a(1-x) + bx)(1-x) + (c(1-x) + dx)x \\ &= (a - b - c + d)x^2 + (-2a + b + c)x + a \end{aligned} \tag{2.3}$$

The two dimensional linear interpolation function (also known as the bi-linear interpolation function) is hence in general a quadratic function. In a similar way, it can be derived that the three dimensional linear interpolation function (tri-linear interpolation) is in general a third order polynomial.

Although the linear interpolation function is a continuos function, its derivative is a discontinuous function. This discontinuity may lead to visible artefacts when the derivative is used in a visualization algorithm. Hence, these applications will in general ask for a reconstruction filter with a continuous derivative.

Figure 2.4 gives an overview of a family of filters that can be constructed by repeated convolution of a square function. The

convolution of two square functions results in a triangular function. Convolving the square function three times results in a quadratic function that will be explained more thoroughly later in this section. Repeating this convolution will in the end lead to a Gaussian function.

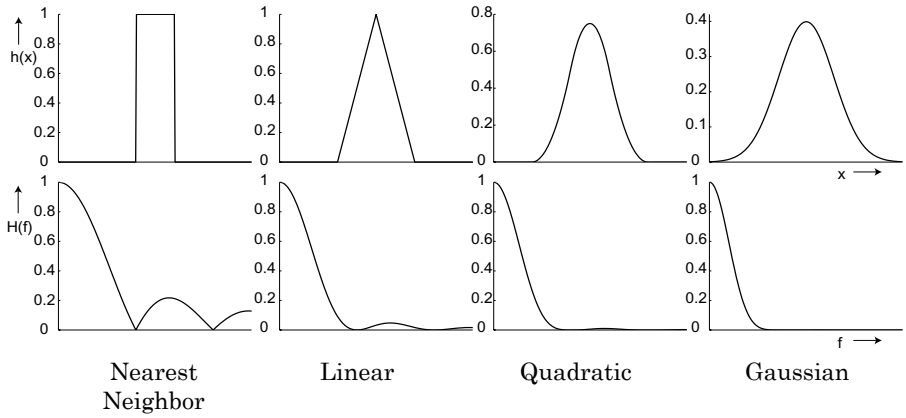


Figure 2.4 A family of reconstruction filters

Because convolution in the place domain corresponds to a multiplication in the frequency domain, convolving the square function n times will result in a frequency response that is the n-th power of the frequency response of the square function, as shown in the bottom row of Figure 2.4. Because the frequency response of the square function has the form $\sin(f)/f$, the n-th convolution will have the form $(\sin(f)/f)^n$. When n approaches infinity, this will result in a function with a Gaussian shape, which could be expected because the Fourier transform of a Gaussian function is also Gaussian.

This family of filters is known as the B-spline polynomials [SCH46]. In computer graphics, these polynomials are often used for tracking a path or surface through sample locations. They can however also be used as reconstruction filters.

The first two filters have already been described as the nearest neighbor and the linear interpolation functions. Hence, these filters will be called interpolating reconstruction filters.

The third filter in this family, the third order B-spline, is a piecewise quadratic reconstruction filter with a continuous derivative ([BOS95a],[DOD97]).

$$h(x) = \begin{cases} h(-x) & x < 0 \\ \frac{3}{4} - x^2 & 0 \le x < \frac{1}{2} \\ \frac{1}{2}x^2 - \frac{3}{2}x + \frac{9}{8} & \frac{1}{2} \le x < \frac{3}{2} \\ 0 & otherwise \end{cases} \tag{2.4}$$

Because the impulse response of this filter is not equal to one at $x=0$, the reconstructed value at a sample location is not equal to the sample value itself. Instead this value will be equal to 3/4-th of its own sample value plus 1/8-th of the value of its two neighbors (x=1 and x=-1). As this reconstruction filter is not a real interpolation function, this filter will be called an approximating reconstruction filter.

When applied in 3D, the reconstructed value at a sample location will be 27/64-th (42%) of the sample value, while 37/64-th (58%) of its value is contributed by the 26 surrounding samples. As a result, small details of about the size of a single sample will be filtered out. This may however also be an advantage, because uncorrelated noise will also be filtered out. Higher order filters in this family of filters will filter the sampled data even more severely. For instance, the fourth order B-spline, also known as the cubic B-spline, has a value of 2/3 at $x=0$, which means that only 30% of the sample value contributes to the reconstructed value at a sample location. This cubic B-spline function will be described later as a member of the cubic-spline functions. Higher order B-spline filters will in general not be used as 3D reconstruction filters because of the high attenuation of high frequency signals and the high computational cost.

The same family of filters will also play an important role in the signal theoretic analysis of the data acquisition process. In CT scanners, the point-spread function can be described by a similar convolution of square functions.

Besides this family of filters, other reconstruction filters can be constructed that can be approximating or interpolating. To be suitable for being used in visualization algorithms that are (partly) based on the derivative, these filters must satisfy a number of conditions:

1 the function should be continuous

2 the derivative of the function should be continuous

3 the function should be symmetrical

4 the value should be one at x=0 and zero at x=n_x, with n_x the locations of the other samples (only interpolating)

5 when the data is constant, the reconstructed value should be the same constant for all values of x

6 when the value of the samples is a linear function of the place, the reconstructed values should also be a linear function of the place

These conditions apply to one-dimensional reconstruction filters that are applied repeatedly in multi-dimensional reconstruction filters. The last condition is especially important when the reconstruction filter has to be used to extract or visualize a shape in two or more dimensions. This will be made clear later in section 5.1 about surfaces in a reconstructed volume.

Based on these conditions, it is possible to derive piece-wise polynomial (interpolating) reconstruction filters that satisfy these conditions. It is obvious that a reconstruction filter based on a single sample (such as nearest neighbor interpolation) cannot satisfy these conditions. A reconstruction filter based on two points that satisfies all conditions except the second is the linear interpolation function. This is also the only possible reconstruction filter based on two sample points that satisfies the sixth condition. The quadratic reconstruction filter described earlier satisfies all conditions except the fourth one. A real interpolating function based on three sample points that satisfies all the conditions is a piece-wise fourth order function:

$$h(x) = \begin{cases} h(-x) & x < 0 \\ 4x^4 - 3x^2 + 1 & 0 \le x < \frac{1}{2} \\ -2x^4 + 8x^3 - \frac{21}{2}x^2 + \frac{9}{2}x & \frac{1}{2} \le x < \frac{3}{2} \\ 0 & otherwise \end{cases} \tag{2.5}$$

A family of filters that is also commonly used in image processing is the set of cubic-spline functions [MIT88]. This family is a collection of third order polynomials:

$$h(x) = \begin{cases} h(-x) & x < 0 \\ \left(2 - \frac{3}{2}B - C\right)x^3 + (-3 + 2B + C)x^2 + 1 - \frac{1}{3}B & 0 \le x < 1 \\ \left(-\frac{B}{6} - C\right)x^3 + (B + 5C)x^2 + & \\ \qquad (-2B - 8C)x + \frac{4B}{3} + 4C & 1 \le x < 2 \\ 0 & otherwise \end{cases} \tag{2.6}$$

This piecewise polynomial function is continuous, symmetrical and has a continuous derivative for all values of B and C. Furthermore, a constant value is always reconstructed to the same value regardless of the value of B and C. Only the fourth and the sixth conditions are not guaranteed. To be able to accurately reconstruct a straight line, the value of B should equal $-2C+1$. The derivative at $x=1$ is then automatically fixed to -1/2. Only when this condition is met, the sixth condition is met.

The previously mentioned cubic B-spline is one of the functions in which this condition is met. The corresponding parameters are $B=1$ and $C=0$, resulting in the following function:

$$h(x) = \begin{cases} h(-x) & x < 0 \\ \frac{1}{2}x^3 - x^2 + \frac{2}{3} & 0 \le x < 1 \\ -\frac{1}{6}x^3 + x^2 - 2x + \frac{4}{3} & 1 \le x < 2 \\ 0 & otherwise \end{cases} \tag{2.7}$$

It is clear that this function is not an interpolating but an approximating reconstruction filter. The cubic-spline functions that are interpolating all have a value for B equal to zero. The only interpolating cubic-spline function that is able to reconstruct a straight line is the function with $B=0$ and $C=1/2$, resulting in the following reconstruction filter:

$$h(x) = \begin{cases} h(-x) & x < 0 \\ \frac{3}{2}x^3 - \frac{5}{2}x^2 + 1 & 0 \le x < 1 \\ -\frac{1}{2}x^3 + \frac{5}{2}x^2 - 4x + 2 & 1 \le x < 2 \\ 0 & otherwise \end{cases} \tag{2.8}$$

This cubic-spline interpolation function is also known as the Catmull-Rom spline. Because this is the only interpolating cubic-spline function able to reconstruct a straight line, this function will be referred to as the cubic-spline interpolation function. Although this function has a continuous derivative, the second order derivative is not continuous. The only cubic-spline function that has a continuous second order derivative and is able to reconstruct a straight line is the cubic B-spline.

The Catmull-Rom spline can also be described as a subset of a set of fourth order polynomials with parameter a:

$$h(x) = \begin{cases} h(-x) & x < 0 \\ ax^4 + \left(-2a + \frac{3}{2}\right)x^3 + \left(a - \frac{5}{2}\right)x^2 + 1 & 0 \le x < 1 \\ ax^4 + \left(6a - \frac{1}{2}\right)x^3 + \left(-13a + \frac{5}{2}\right)x^2 + (a12 - 4)x + 2 - 4a & 1 \le x < 2 \\ 0 & otherwise \end{cases} \tag{2.9}$$

For a=0, this function is equal to the Catmull-Rom cubic-spline interpolation function. When a is equal to -1/2, not only the first derivative, but also the second derivative of this function is continuous.

In the same way, polynomials for reconstruction filters using more than four sample points can be derived. This will however significantly increase the complexity of the (3D) algorithm. Besides the piece-wise polynomial functions, it is also possible to derive other functions that also satisfy the six conditions.

Figure 2.5 shows a collection of the reconstruction filters described in this section. Three of these functions are real interpolation functions (linear, three point, and cubic-spline interpolation), while

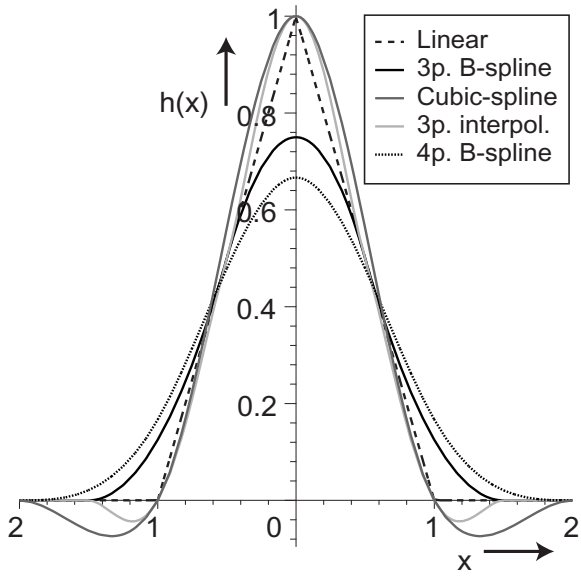


Figure 2.5 Collection of reconstruction filters described

the other two, the third and fourth order B-spline, are approximating reconstruction filters.

As mentioned before, using a reconstruction filter that does not satisfy the six conditions will lead to visible artefacts, especially in 3D visualization algorithms. The effect of these artefacts on the visualized shape will be shown later.

2.4 Volume gradients

In some 3D visualization applications, the volume gradient is used (see section 4.1.2). In a continuous 3D data field *D(x,y,z)*, this volume gradient would be equal to the vector:

$$\vec{G}(x, y, z) = \begin{pmatrix} \frac{\partial}{\partial x} D(x, y, z) \\ \frac{\partial}{\partial y} D(x, y, z) \\ \frac{\partial}{\partial z} D(x, y, z) \end{pmatrix} \tag{2.10}$$

In discrete data fields, the volume gradient at a sample location (n_x,n_y,n_z) is generally determined by the central difference:

$$\vec{G}(n_x, n_y, n_z) = \begin{pmatrix} (D(n_x+1, n_y, n_z) - D(n_x-1, n_y, n_z))/2 \\ (D(n_x, n_y+1, n_z) - D(n_x, n_y-1, n_z))/2 \\ (D(n_x, n_y, n_z+1) - D(n_x, n_y, n_z-1))/2 \end{pmatrix} \tag{2.11}$$

This central difference gradient is used to calculate the gradient on the same grid as the original grey-values. Using a reconstruction filter, this discrete gradient field can be extended to an arbitrary *(x,y,z)* location.

Although this approach is commonly used, the rather large distance of two samples to calculate the central difference gradient will result in unacceptable filtering of the gradient. Much better results can be achieved when instead of the central difference gradient, the intermediate difference gradient is calculated ([BOS95a], [TER96], [BEN95], [BEN96]). With this method, two neighboring samples are subtracted, resulting in the gradient in the middle of two samples:

$$\begin{aligned} G_x\left(n_x+\frac{1}{2}, n_y, n_z\right) &= D(n_x+1, n_y, n_z) - D(n_x, n_y, n_z) \\ G_y\left(n_x, n_y+\frac{1}{2}, n_z\right) &= D(n_x, n_y+1, n_z) - D(n_x, n_y, n_z) \\ G_z\left(n_x, n_y, n_z+\frac{1}{2}\right) &= D(n_x, n_y, n_z+1) - D(n_x, n_y, n_z) \end{aligned} \tag{2.12}$$

Because the three components of the gradient vector are all shifted by half a sample in a different direction, the intermediate difference gradient cannot be calculated at the sample locations. This method results in three separate grids, one for each gradient component. By reconstructing the three components separately however, the volume gradient can be calculated at arbitrary locations. When a linear interpolation function is used, both methods give the same result at the sample locations. In the middle of two samples however, the intermediate difference gradient is much more accurate than the interpolated central differences. The central difference method will result in:

$$G_x\left(n_x+\frac{1}{2}, n_y, n_z\right) = \frac{1}{2}(G_x(n_x+1, n_y, n_z) + G_x(n_x, n_y, n_z)) \tag{2.13}$$
$$= \frac{1}{2}(D(n_x+2, n_y, n_z) + D(n_x+1, n_y, n_z)) -$$
$$\frac{1}{2}(D(n_x, n_y, n_z) + D(n_x-1, n_y, n_z))$$

It can be seen that the central difference gradient at this location is equal to the intermediate difference after pre-filtering the data with [1/2, 1/2].

The volume gradient in a discrete data field can also be calculated as the volume gradient in the reconstructed continuous data field. This volume gradient of the reconstructed data field will be called the congruent gradient of the reconstructed data field. As the reconstruction filter is applied in three directions separately, the derivative of the reconstruction filter can be used to calculate the three gradient components separately. The reconstruction filter is applied in two directions, while the derivative of the reconstruction filter is used to calculate the gradient in the third direction. To calculate the x-direction component of the gradient vector in a linearly reconstructed data field, both the data values a_1, b_1, c_1 and d_1 at $n_x=x_1$ as well as the data values a_2, b_2, c_2 and d_2 at $n_x=x_2$ are bi-linearly interpolated. As the derivative of the linear interpolation function is [-1,1], the x-direction gradient is the difference of these two interpolated values:

$$G_x(x, y, z) = \frac{\partial}{\partial x} D_r(x, y, z) \tag{2.14}$$
$$= L2(a_2, b_2, c_2, d_2, y, z) - L2(a_1, b_1, c_1, d_1, y, z)$$

Because the linear interpolation function is continuous, the x-direction component of the gradient is also continuous in the y- and z-direction. This gradient component is however piece-wise constant in the x-direction, which could be expected because the interpolated data field is linear in the direction of the axes. In the same way, the other components of the gradient vector are calculated. As a consequence, the volume gradient is discontinuous, which leads to visible artefacts when this gradient is used for visualization purposes. When a reconstruction filter with a continuous derivative is used, the congruent volume gradient will also be continuous.

Figure 2.6 shows the derivatives of the reconstruction filters in Figure 2.5. These derivatives have to be used to calculate the congruent gradient. The derivative of the linear interpolation function is shown to be discontinuous. The derivative of the quadratic reconstruction filter is piece-wise linear and continuous. The fourth order polynomial interpolation function that is based on three sample points has a derivative that is also continuous. The strange shape of the derivative at x=±0.5 might however also lead to visible artefacts.

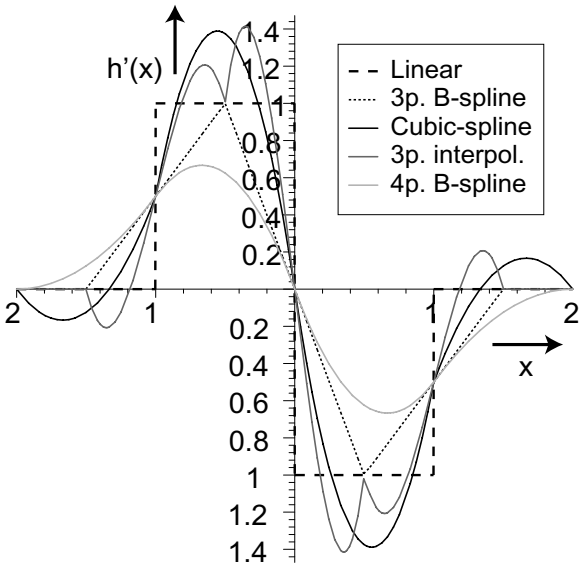


Figure 2.6 Derivatives of the reconstruction filters

The three components of the interpolated intermediate difference gradient are piece-wise linear and continuous in all directions. This volume gradient can also be derived by linear interpolation in two dimensions, while the derivative of the quadratic reconstruction filter is used to calculate the gradient component in the third dimension. It may be concluded that the intermediate difference gradient is a mixture of the congruent gradient of the linear and quadratic reconstruction filters.

2.5 Edge detection

The detection and visualization of edges in volumetric data is a problem that plays an important role in volume rendering algorithms. As mentioned before, the point-spread function of (medical) acquisition devices will pre-filter the input data. A step edge will consequently be blurred over a certain transition area. When the point-spread function is for instance a Gaussian function with s=1, a step edge at x=0 will be filtered to an error function:

$$S(x) = \frac{1}{2} erf\left(\frac{1}{2}\sqrt{2}x\right) + \frac{1}{2} \tag{2.15}$$

Most edge detection algorithms in discrete fields are limited to the sample resolution. The convolution of the object given with the acquisition point-spread function results in the mentioned blurring at the edges, known as the partial volume effect. The resulting encoding of the edge location in the data values makes it possible to determine the edge location accurately between two samples.

In continuous unsampled data, the location of the step edge can be found by finding the maximum of the derivative of the step response. This derivative is equal to the point-spread function of the acquisition apparatus. Because the point-spread function is maximal at x=0, this location will correspond to the location of the step edge. The location of the maximum can be determined by searching the zero crossing of the second order derivative.

This method can however not be easily extended to sampled data fields. Although it is possible to reconstruct the data field, it is generally not possible to get an accurate estimation of the location of the edge based on the second order derivative in this reconstructed field. In 1D, the derivative of linearly interpolated data is piece-wise constant, so there is no unique place at which the maximum is located. The derivative of quadratic reconstructed data is piece-wise linear, with interval boundaries half way two sample locations. The extreme of this piece-wise linear function will hence always coincide with the boundaries of the piece wise linear derivative. The second order derivative of the quadratic reconstruction filter is piece-wise constant, so it is in general not significant to use the location where this second order derivative equals zero as an approximation of the edge location.

A similar problem occurs when the cubic-spline interpolation function is used as a reconstruction filter. Although the cubic-spline interpolation function is a piece-wise third order function and the second order derivative is therefore piece-wise linear, this second order derivative is not continuous. Hence, even the cubic-spline interpolation cannot be used to get an accurate estimation of the location of a step edge based on the second order derivative. A reconstruction filter that can be used to estimate an edge with sub-sample accuracy based on the second order derivative must hence at least have a continuous second order derivative.

Another method to find the edge location is based on the reconstructed value. When the point-spread function is symmetrical, the acquired value at the step edge is the mean of the acquired values at both sides of the edge. When the data is reconstructed, the location at which the reconstructed data equals this mean can be used as an estimate of the step edge location. This approach does not suffer from the ambiguity caused by the discontinuity of the first and/or second order derivatives, which is the case when a derivative based edge estimator is used. Even for a lower order reconstruction filter as the linear interpolation function, a non ambiguous edge location can be found when the two neighboring samples have a different value. This will generally be the case for samples near a step edge. Hence, the continuity of the reconstruction filter itself is generally satisfactory to find a non-ambiguous edge location.

CHAPTER

3

VOLUME VISUALIZATION

The term 'volume visualization' is commonly used to describe all techniques that can be used to visualize volumetric datasets. An important field where these volume visualization methods can be applied, is the visualization of volumetric datasets generated by three-dimensional medical scanners. The same visualization methods are however also applicable to all other fields requiring visualization of three-dimensional data.
Several methods have been developed to visualize such a volumetric dataset. For a certain application, one of these methods may be preferable over the others, while for another application another method might be more useful. This chapter will describe some of the visualization methods of particular relevance for medical applications.

3.1 Definitions

A volumetric dataset consists of a three dimensional set of scalar values. The locations at which these values are given are called voxels, which is an abbreviation of volume element. This is in fact a three-dimensional extension of the definition of a pixel (=picture element), known from two-dimensional image processing. The value of a voxel is often referred to as the voxel value.

Figure 3.1 shows a cube surrounded by eight voxels. This cube will be referred to as a voxel cube or cell. This voxel cube is the smallest sub-volume in which the volume can be divided. A dataset containing $N_x*N_y*N_z$ voxels can hence be subdivided in $(N_x-1)*(N_y-1)*(N_z-1)$ of these basic sub-volumes.

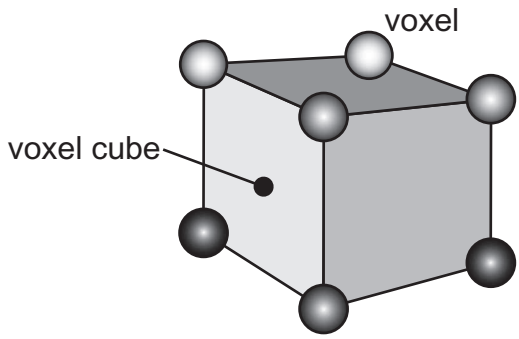


Figure 3.1 Voxel cube consisting of eight voxels

3.2 Multi planar reformatting

In medical scanners, the (grey value) data is often acquired slice by slice. This acquired data is typically observed by also looking at these slices in a slice by slice way. Figure 3.2 shows two slices measured with a MRI scanner. The left image shows a slice through the head in which the brain tissue is very well visible. The right image shows the same slice in which only the blood vessels are visible. While observing these slices might be sufficient to diagnose the AVM (=malformation of arteries) in the right side of the brain (left side of the image), it is very hard to understand the three dimensional structures from just looking at the stack of

slices. Especially in the right image of Figure 3.2, where blood vessels that intersect the slice are depicted as small dots.

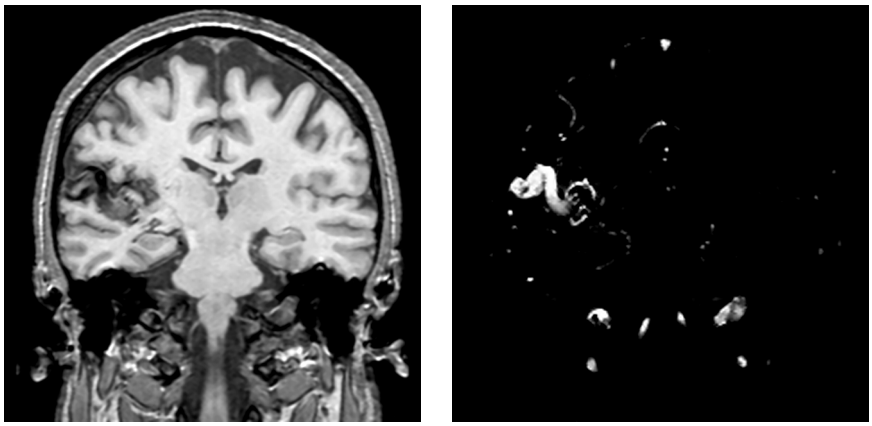


Figure 3.2 Slices of a MRI dataset of a patient with an AVM

Multi planar reformatting (MPR) is a technique that is used to visualize the (re-sampled) grey values in arbitrary cross sections through the volumetric data. The cross sections made through the 3D data will be flat in most cases, however arbitrarily bent cross sections are of special relevance for specific application areas, such as a survey of the spinal chord or a study of the dents. The main advantage of the multi-planar reformatting method is that one is not restricted to viewing in the direction the data was scanned, which makes it possible to visualize data that was measured in different slices in one two-dimensional image. Another advantage of this visualization method is the speed. The main drawback is that the visualized data is, like the original slices, two dimensional. A single image does not give much more insight in the three dimensional structures than the measured slices. Blood vessels intersecting the reformatted plane will still be depicted as small dots. To be able to visualize three dimensional structures in a volumetric dataset in a two dimensional image, the output image should contain more information than just the grey-values on a plane through the dataset.

3.3 Ray casting

More sophisticated volume visualization algorithms take as input the complete discrete three dimensional data field $D(n_x,n_y,n_z)$ and project this data field onto a two dimensional screen. This can be achieved by casting a ray from each pixel *(i,j)* on this screen through the data field, which is known as ray casting. The depth index *k* defines the location on these rays, as can be seen in Figure 3.3. A coordinate transformation is used to link the display coordinates *(i,j,k)* to the object-space coordinates *(x,y,z)*. This transformation can be organized to contain a perspective component.

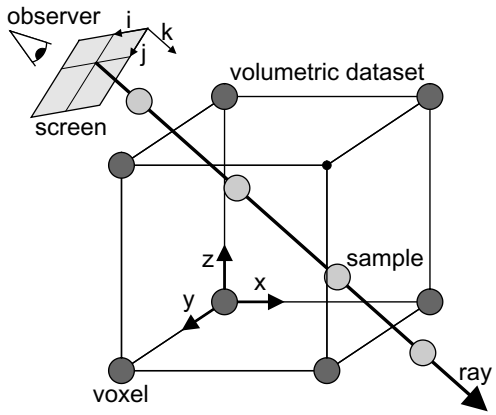


Figure 3.3 Common concepts and terms in ray casting algorithms

3.3.1 Maximum intensity projection

Maximum intensity projection (MIP) [ZUI95] is a visualization technique that projects the highest intensity on each ray onto the corresponding pixel on the screen. The grey values are calculated by re-sampling the data at discrete *k* locations on the rays. Depending on the dataset and the object to be visualized, it may also be desirable to visualize the minimum intensity on a ray. To distinguish between these two projection types, the abbreviations MinIP or mIP may be used for minimum intensity projection, while for maximum intensity projection the abbreviations MaxIP or MIP may be used. Because the underlying visualization algorithms are

almost identical, the abbreviation MIP will be used here for both projection types.

In medical applications, the MIP method is mainly used to visualize blood vessels as shown in Figure 3.4 for the MRI dataset shown in the right image of Figure 3.2. It can be seen in this figure that with this method the blood vessels are projected on the screen. It is hence possible to see the blood vessels in the whole volume in a single projection image.

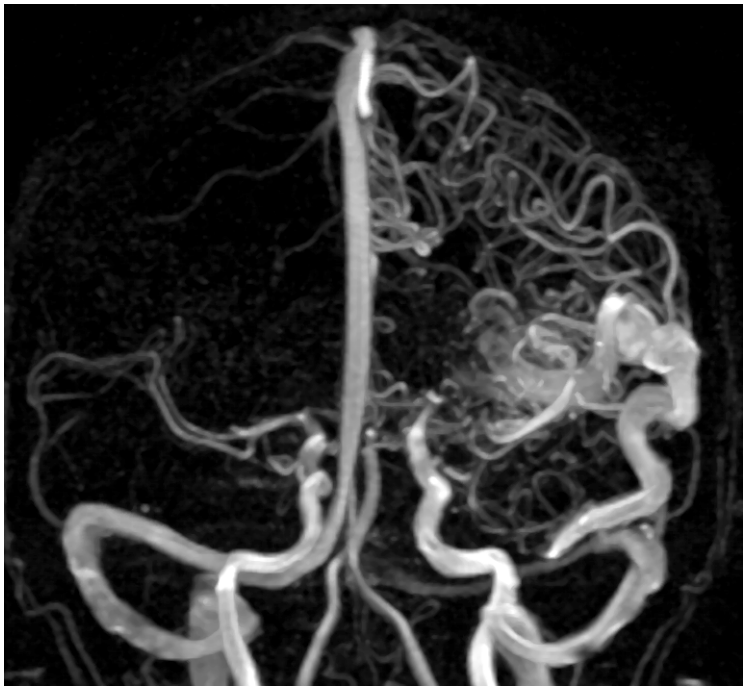


Figure 3.4 Maximum intensity projection

A disadvantage of the MIP method for this purpose is the impossibility to follow a blood vessel in the foreground when blood vessels in the background with higher intensity cross this vessel. Furthermore, it is possible that another object with a high grey value makes it impossible to see the desired object.

3.3.2 Closest vessel projection

To be able to see blood vessels in the foreground regardless of the intensity of vessels in the background, the closest vessel projection

(CVP) [ZUI95] algorithm can be used. This algorithm projects the maximum intensity of the first encountered blood vessel onto the screen as shown in Figure 3.5.

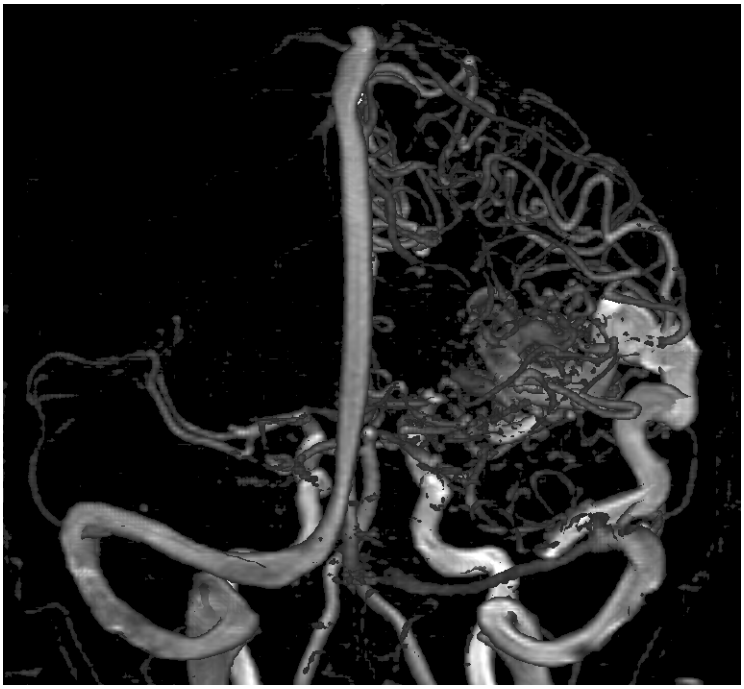


Figure 3.5 Closest vessel projection

Ray casting can be stopped when the first vessel is found and hence this technique can also be significantly faster than maximum intensity projection. Because vessels in the background are obscured by vessels in the foreground, this method also gives more depth information than the MIP method. The main problem with this method is the detection of vessels. In a straightforward approach, the re-sampling locations on a ray where the re-sampled grey-value exceeds a chosen threshold are supposed to be inside a vessel. Choosing a high threshold will make the smaller vessels invisible. Choosing a threshold just above the background grey-value will however result in a noisy image in which the main blood vessels may be obscured by the noise in the foreground. This method is hence sensitive to noise.

Besides projecting the maximum or minimum value on a ray, it is also possible to perform calculations on the whole set of values on a

ray. For instance, the mean grey value on a ray can be projected. This can give results similar to x-ray images. This approach is however not (yet) commonly used in medical image processing.

3.3.3 Volume rendering

Volume rendering is a volume visualization method that projects the shape of objects in the data set onto the screen instead of the grey values. This can be accomplished by calculating volume gradients in the dataset that will be used as the orientation of the object shape. A shading model is used to calculate the observed effect of a light source shining on this shape. By assigning transparency and color based on the grey-values in the dataset, the contribution of this shaded object to the value of a pixel on the screen can be calculated using a composition formula.

Another approach applies the composition formula directly on the (re-sampled) grey-values or after mapping the grey-values to a color value. Yet another approach maps the grey-values to the screen via the frequency domain by using Fourier transforms [MAL93]. Because these volume rendering methods that do not use shading can hardly be compared to the methods that do use shading, the term volume rendering will be used in the remainder of this thesis for the volume rendering methods based on shading.

A detailed description of various volume rendering algorithms will be given in the following chapter.

CHAPTER

4

VOLUME RENDERING

As mentioned in the previous section, volume rendering is a volume visualization method that projects the shape of objects in the data set onto the screen instead of projecting the grey values. Currently, many different volume rendering algorithms exist, while each algorithm may have many more or less different implementations. This chapter will describe some of the most commonly used volume rendering algorithms and implementations.

4.1 Common concepts

Many volume rendering algorithms require the calculation of the color and opacity at an arbitrary location in the dataset and the color of a pixel on the screen. Therefore, these common concepts will be discussed before the description of the actual algorithms and implementations. Although some of these common concepts are mainly applicable to ray-casting based volume rendering methods, they may also be used for some other volume rendering algorithms.

4.1.1 Opacity calculation

Most volume rendering methods calculate the opacity at discrete (sample) locations on the rays. This requires two computation steps. Because the sample locations on the ray will generally not coincide with the voxel locations, some re-sampling filter will have to be used. Furthermore, the grey value data has to be converted to an opacity value. The easiest way to calculate this opacity is to assign an opacity to all possible grey values. In this way, a look-up table can be used to directly convert a grey value to an opacity value. This approach can be used when the object of interest has a grey value that is higher (or lower) than its surrounding. The grey values that occur inside the object that is visualized will generally be assigned a high opacity, while the grey values that occur in the surrounding will be assigned a low or zero opacity (fully transparent). Consequently, we will only see the object of interest.

More complex methods assign an opacity value to the length of the volume gradient. This approach makes regions with a large volume gradient, such as the transition area between two tissues, opaque or semi-transparent, while regions with a small volume gradient, for instance inside an object, will be made fully transparent. The grey value based and the volume gradient based opacity calculation methods can also be combined, for instance by multiplying the two opacity values resulting from each method.

4.1.2 Shading and coloring

To calculate the effect of a light source shining on a surface, the direction of the surface normal has to be known. In most volume

rendering algorithms, the volume gradient is used to approximate the direction of this normal vector.

When the surface normal (or volume gradient) is known, a shading model can be used to calculate the effect of light shining on this surface. A simple shading model contains three components: an ambient, a diffuse and a specular component. The ambient component is used to simulate reflections of the light from surrounding objects. The diffuse component simulates the light reflected equally in all directions. This component depends on the amount of light that hits a unit area on the surface. The specular component is used to simulate the reflection of the light in the direction of the observer.

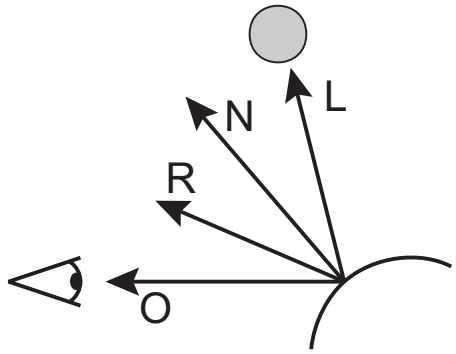


Figure 4.1 Shading model

A very popular shading model, illustrated in Figure 4.1, was derived by Phong [PHO75]. In this shading model, the ambient component is assumed to be a constant A. The inner product of the light vector $\vec{L}$ and the surface normal vector $\vec{N}$ multiplied with a constant D is used as the diffuse component. The specular component is defined as the n-th power of the inner product of the reflected light vector $\vec{R}$ with the observer vector $\vec{O}$ multiplied with a constant S. This results in the following equation for the observed light intensity:

$$I = A + D(\vec{N} \cdot \vec{L}) + S(\vec{R} \cdot \vec{O})^n \tag{4.1}$$

This equation only gives a single value that can be used for grey value rendering. When objects are rendered in full color, each object can have a different color, and even the light source can have a color. In this case, the reflected light will have the color of the

light source, while the color of the ambient and diffuse components will depend on both the light and the object color.

4.1.3 Composition

Most volume rendering algorithms calculate the opacity and color of an object along the rays at discrete k values that run from either the position of the viewer to infinity (known as front-to-back rendering), or the other way around (back-to-front rendering). A composition formula [LEV88] is used to calculate the color $C^*(i,j)$ of a pixel on the screen from the opacity and color samples on a ray:

$$C^*(i,j) = \sum_{k=1}^{n} \left\{ \alpha(i,j,k) \cdot C(i,j,k) \cdot \prod_{l=1}^{k-1} (1 - \alpha(i,j,l)) \right\} \tag{4.2}$$

The product in the right hand side of the equation is equal to one minus the accumulated opacity up to point *k-1*. Calculation of this product for each sample point would be extremely expensive. Depending on the direction, one of two different algorithms can be used. The least complex formula is the composition formula for back-to-front rendering:

$$C^*(i,j,k) = \alpha(i,j,k) \cdot C(i,j,k) + (1 - \alpha(i,j,k)) \cdot C^*(i,j,k+1) \tag{4.3}$$

In this formula, $C^*(i,j,k)$ is the composited color from sample n to k (with $k<n$). This can be easily implemented by storing only the composited color.

Most volume rendering methods however use the front-to-back rendering method. It is obvious that when the accumulated opacity is close to one, one minus the accumulated opacity, and hence the right hand side product in equation 4.2, will be close to zero. Hence, the color of samples behind this point will not (or hardly) contribute to the color of the corresponding pixel on the screen. Casting a ray can hence be stopped at a point where the accumulated opacity is (almost) one. This is called early ray termination [LEV88]. By storing the accumulated opacity, the computation of the product is reduced to a single multiplication at each non-transparent re-sampling location k. The following equation shows the composition formula for front-to-back rendering:

$$\begin{aligned} \Delta\alpha &= \alpha(i,j,k) \cdot (1 - \alpha^*(i,j,k-1)) \\ C^*(i,j,k) &= \Delta\alpha \cdot C(i,j,k) + C^*(i,j,k-1) \\ \alpha^*(i,j,k) &= \alpha^*(i,j,k-1) + \Delta\alpha \end{aligned} \tag{4.4}$$

Because the contribution of the color at a sample point now depends on the opacity of the samples in front, not only the composited color, but also the composited opacity $\alpha^*(i,j,k)$ has to be stored.

When the distance between samples on a ray is changed without adapting the opacity value at the sample locations, the perceived transparency of the material will change. This change in perceived transparency is in general not desirable. It is therefore better to use an opacity setting that is independent of the sampling distance. This means that the value of $\alpha(i,j,k)$ has to be adapted to correct for the sampling distance ([BOS95a],[TER96]). As a result of this sampling correction, the composited opacity over a fixed distance should be unaltered when the sampling density is changed. Hence, also one minus the composited opacity, which equals the right hand side product in equation 4.2, should remain unaltered:

$$1 - \alpha^*(i,j) = \prod_{k=1}^{N_1 \cdot n} (1 - \alpha_{N_1}(i,j,k)) = \prod_{k=1}^{N_2 \cdot n} (1 - \alpha_{N_2}(i,j,k)) \tag{4.5}$$

When considering an area with a constant opacity, equation 4.5 can be simplified to:

$$(1 - \alpha_{N_1})^{N_1} = (1 - \alpha_{N_2})^{N_2} \tag{4.6}$$

When the factor N_1 is chosen such that the sampling distance equals a reference distance for which α is given, the corrected opacity for a oversampling factor $N=N_2/N_1$ can be found by:

$$(1 - \alpha) = (1 - \alpha_N)^N \tag{4.7}$$

Hence,

$$\alpha_N = (1 - \sqrt[N]{1 - \alpha}) \tag{4.8}$$

Consequently, α can for instance be chosen to be the opacity per meter. Given a sampling distance *sd* in meters, and hence a sampling rate of 1/*sd* samples per meter, the corrected sample

point opacity will than be $\alpha_N = 1-(1-\alpha)^{sd}$. Although this seems to be a rather expensive function to be evaluated at every sample point, the table look-up approach to calculate the opacity based on the grey values makes it possible to perform this operation only once on each cell of the opacity look-up table and only when the opacity value or the sampling distance is changed.

4.2 Voxel space volume rendering

Many of the current volume rendering algorithms are based on the method introduced by Marc Levoy [LEV88]. This method pre-computes a discrete opacity field $O(n_x,n_y,n_z)$ and a discrete color field $C(n_x,n_y,n_z)$ from the data field. The opacity field is computed by applying a (non-linear) opacity function to the data value at each voxel. The shading is based on the central difference volume gradient at each voxel location. The object color is calculated by applying a (non-linear) color function to the data value at each voxel. For a fixed light direction, the color field is calculated from the shading and the object color at voxel locations. The opacity and color at the re-sampling locations on the rays are calculated through re-sampling of the corresponding discrete fields. A composition step is subsequently used, using a front-to-back order, to compute the color of the pixels (i,j) on the screen.

In this way, the 2-D view of the 3-D object can be calculated from an arbitrary viewing location, without the need to re-compute the opacity field and the color field for each viewing angle. When the position of the light source is changed however, the expensive color calculation has to be redone for the whole dataset. Furthermore, it is almost impossible to apply the full shading model derived by Phong [PHO75], as the position of the observer and the light vector with respect to the surface normal are both needed for the specular shading. Specular shading can therefore only be used when the color dataset is re-computed for every viewing angle. As this computation is much more expensive than the rendering itself, one is constrained to use the diffuse shading technique, that is independent of the position of the observer, to compute the additional color field.

Another disadvantage of this ray casting based voxel space volume rendering is the high memory requirement. Besides the original

dataset, two additional datasets are required. When a full-color color field is used, the total memory needed can be five times the size of the original dataset.

Because direct implementation of the ray casting algorithm just described is not capable of rendering volumetric datasets with interactive speed (about one frame per second) on current workstations, several optimizations have been developed to increase the rendering speed. Some of these optimizations, such as early ray termination (see section 4.1.3), skipping large transparent sub-volumes and octree based optimizations [WIL92], [SHE96], have little to no influence on the rendered scene. With these optimizations however, interactive rendering is still not possible. Using a shear-warp factorization of the viewing transform [LAC94], interactive speed is feasible for small sized datasets. In this algorithm, the data is pre-processed (sheared) such that during ray casting the samples on the rays coincide with an intermediate grid so no re-sampling is necessary. This is achieved by re-sampling (in 2D) each slice in the dataset at the locations where the rays intersect the slice. The deformation of the resulting image is corrected in a post-processing (warping) step. This method however significantly affects the rendered image under certain angles, making the algorithm view-point dependent.

4.3 Super resolution volume rendering

Due to the interpolation of voxel location opacities in Levoy's algorithm and the voxel projection in splatting based algorithms (described in section 4.5), the perceived resolution is equal to the voxel size. By oversampling the data and applying the classification (opacity and color) functions on the higher resolution data, renderings with a perceived resolution that is much higher than the voxel resolution is possible. Methods that are based on this approach are therefore referred to as super resolution volume rendering methods [BOS95a], [BOS95b], [TER96].

An important feature of these methods is that they apply the non-linear opacity function at the sample locations on the rays, after the application of a re-sampling function on the raw voxel data. In this way, the discrete data field is (implicitly) re-sampled to a

higher resolution transformed data field and then subjected to the opacity function.

Moreover, the shading is calculated at the re-sampling locations as well. This implies that a technique should be used that is able to compute the volume gradient vector $\hat{G}(x, y, z)$ at arbitrary *(x,y,z)* locations. The voxel location (central difference) volume gradients can be re-sampled to find the gradient vector at the re-sampling location, requiring three (tri-linear) interpolations, as described in section 2.4. This approach is a three-dimensional extension to Phong normal interpolation [PHO75], where the surface normals are calculated at the three vertices of a triangle and interpolated to find the normal at arbitrary positions on this triangle. In this respect, the color interpolation scheme in Levoy's algorithm can be seen as the three-dimensional extension to Gouraud shading [GOU71], where the colors are calculated at the vertices of triangles and interpolated to find the color of arbitrary points.

A preferred method calculates the intermediate difference gradients of the data values in between the voxels, as described in section 2.4. Because the three gradient component fields are shifted in a different direction, this re-sampling has to be done with different coefficients, while the re-sampling of the three central difference gradient components can be done with the same coefficients. The calculation of these coefficients makes the intermediate difference method slightly more expensive. However, the intermediate difference gradient calculation results in a higher resolution gradient field than the central difference gradient method. Furthermore, the central difference method uses four data points in addition to the intermediate difference method, for each gradient component.

Figure 4.2 shows a cross section of a cone that contains sharp edges and three slits with variable width. This object is filtered and sampled comparable to the filtering and sampling in the data acquisition process. The object is sampled on a 32×32×32 voxel grid such that the slits have the width of a half, one and two voxel distance respectively. The resulting grey-value dataset will be referred to as the cone dataset. This dataset represents a small object with fine details that is sampled on a rather coarse grid. Therefore, this dataset will be used as a reference dataset to visualize artefacts and limitations of different volume rendering algorithms.

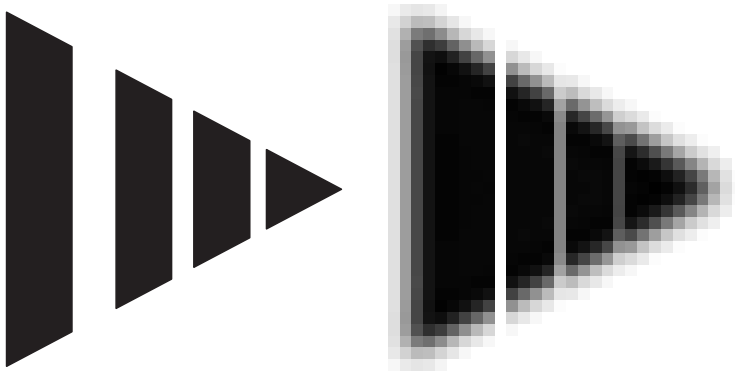


Figure 4.2 Cross section of a cone with slits (left) and the 32×32×32 voxel cone dataset that will be used to compare the artefacts in different volume rendering algorithms

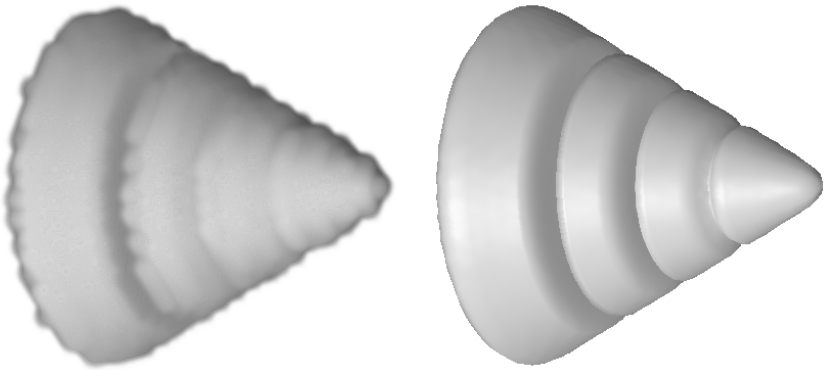


Figure 4.3 Rendering of the cone dataset using Levoy's algorithm (left) and the super resolution volume rendering algorithm (right)

Figure 4.3 shows two renderings of this cone dataset. To visualize the surface of the cone, the opacity function chosen is zero below a certain threshold (fully transparent) and one above this threshold (opaque). Note that although the surface of the object is rendered, the visualization algorithm stays the same. Visualization algorithms that are based on direct visualization of the volume will therefore still be called volume rendering algorithms. When such a direct volume rendering algorithm is used to visualize surfaces, it is often called a surface rendering algorithm. This term will however be reserved for methods that do not directly visualize the

volume, but a surface description extracted from the volume, as will be discussed in section 4.4. All visualization algorithms that take the volume data as input will hence be called volume rendering algorithms, while algorithms that take a surface description as input will be called surface rendering algorithm, regardless of the ability to visualize surfaces or transparent volumes.

The left image of Figure 4.3 shows the result of interpolating voxel location opacities as used in Levoy's volume rendering algorithm. At the voxel locations, the opacity is zero or one, while in between the voxels at the surface the opacity falls of from one to zero. This is visible in the rendering as semi-transparent cubes. Hence, even with this extreme opacity function, the visualization of a sharp surface is not possible. Using the super resolution algorithm however, visualization of sharp surfaces located between voxels is possible, as shown in the right image of Figure 4.3.

Although calculating and shading the intermediate difference volume gradient at the re-sampling location is much more expensive than just interpolating the color of surrounding voxels, the gradient is only calculated at re-sampling locations with non-zero opacity. When an opaque surface is visualized with the super resolution method, this gradient calculation is done only once for each ray that hits the surface. When for instance an opaque surface is rendered in a 256×256×256 voxel dataset, Levoy's method would require 16 million gradient calculations. When the dimension of the dataset grows with a factor N, the number of pre-computations grow with a factor of N^3. When for instance all rays in a 256×256 image hit the surface, the super resolution algorithm would require only 64,000 gradient calculations regardless of the size of the dataset. Because typical renderings may require over a million interpolations, the rendering speed is only marginally influenced by the gradient calculation when an opaque surface is rendered.

4.4 Voxel based surface rendering algorithms

Surface rendering is a technique to render a set of simple graphical primitives that describe a surface, like a set of polygons. The advantage of this technique is the availability of low cost hardware

to speed-up the visualization. Although this hardware can not be used directly to visualize surfaces in volumetric data, it is possible to translate the volumetric dataset into a collection of polygons such as triangles. This translation is known as triangulation. Algorithms that translate a dataset into such a surface description to be able to use surface rendering hardware to render this surface description, will be referred to as voxel based surface rendering algorithms.

The most popular approach to convert the volumetric data into a set of polygons is the marching cubes method introduced by Lorensen and Cline [LOR87]. In this algorithm, each unit cell consisting of eight voxels is examined for the presence of a surface. A surface intersects a cell when one of the voxel values is above and another is below a certain value. This value is called the iso-value. When a surface is found to intersect a cell, the intersection x of the surface with the edges of the cell are calculated by solving $a+x(b-a)=Iso$, with a and b the grey value of the voxels connected by the edge. Polygons in these cells are constructed by connecting the intersection points. The binary decision at the eight voxels leads to 256 different possible cases, of which 15 have been shown to be topologically distinct. The surface that is extracted by connecting the polygons is called the iso-surface.

This method can generate multiple triangles for each cell. The total number of triangles generated with this method can therefore become extremely large for large sized datasets with complicated iso-surfaces, which is typical in medical datasets. This huge amount of triangles can prohibit a high rendering speed, even on high performance workstations. Furthermore, the generation of the iso-surface using this approach is slow. Therefore, several algorithms have been developed to reduce the number of polygons created on one hand and the surface extraction time on the other.

One of the approaches to reduce extraction time is based on octrees to decrease the time needed to traverse the cells in the volume [WIL92]. This approach does however not influence the number of polygons created. Although this algorithm does significantly increase the surface extraction speed, the reported calculation times were well above 100 seconds for rather small datasets. A significant reduction in CPU-time is possible when a modern, faster workstation is used. Interactive speed is however still not feasible. Therefore, a lot of research is still done to optimize this

algorithm especially when the iso-value should be changed and the iso-surface should be visualized interactively.

One of the approaches that is able to reduce the number of polygons is the adaptive marching cubes algorithm [SHU95]. This reduction is achieved by adapting the size of the triangles to the shape of the surface. The reported reduction is however small compared to other methods such as octree-based decimation [SHE96]. This approach uses adaptive downsampling of the volume data and applying the marching cubes algorithm on larger cells in regions where the iso-surface is mostly flat. By using a surface tracking algorithm, both the number of polygons and the extraction time is further reduced. Although this has some advantages, only one (connected) surface is extracted, while the actual iso-surface in the data may exist of multiple separate surfaces. Other methods are based on interval volumes [FUJ96], a multiscale method [GUO95], a modified branch on need octree [CHU95] and a method based on spatial error bounds [KLE96]. Wilhelms and Van Gelder [WIL95] introduced multi-dimensional trees for controlled volume rendering with fewer triangles. An approach, in which a deformable surface is extracted, with considerable spatial error, from volume data is given in [LUR98].

While reducing the number of polygons is necessary for high speed visualization, the opposite approach will frequently be needed to enhance the spatial resolution. The discrete data field is made continuous in this case using a reconstruction filter of choice. Triangulation of the implicit surface defined by $D_r(x,y,z) = Iso$ can generate a lot of triangles within one unit cell [BLO88]. As the marching cubes algorithm can already create a huge set of triangles, this approach will however be very slow, both in terms of surface extraction as well as in terms of rendering speed. The rendering speed can however be improved by reducing the number of polygons using methods comparable to the ones described before.

4.5 Splatting

A completely different approach is based on projecting voxels from the dataset onto the pixels on the screen, such as splatting [WES89]. The order of traversal of the index-set (n_x,n_y,n_z) is chosen such that the object space is traversed either in front-to-back or

back-to-front order. A projection of a voxel cube is pre-computed as splatting footprint. The composition process described in section 4.1.3 is modified to use a complete RGBα image as accumulator and each splatting footprint as local contribution to the final 2D image. The opacity and gradient are either computed at the voxel-locations or the center of a voxel-cube.

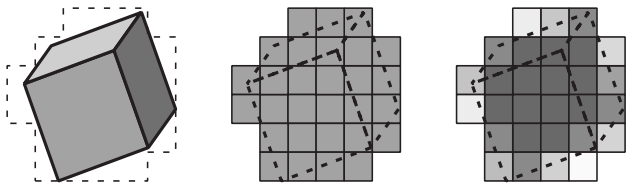


Figure 4.4 Voxel cube and corresponding normal and anti-aliased splatting footprints

Figure 4.4 shows a possible splatting footprint measuring 6x6 pixels. The left image shows a voxel cube from a certain angle. Given this orientation and the size of the cube, the size and shape of the splatting footprint can be determined. All pixels in the footprint have the same color, as shown in the middle image. This color can be pre-computed for each voxel given in the dataset in the same way as in direct volume rendering. In a similar way, each voxel is assigned an opacity. The splatting footprint may be anti-aliased at its edges by modulating the opacity of a pixel with the area of the pixel that is inside the contour of the actual voxel cube, as shown in the right image of Figure 4.4. The pixels in the center have an opacity equal to the voxel opacity (dark grey), while the pixels at the edge are made more transparent (light grey).

The main advantage of the splatting algorithms is that they are much faster than ray casting algorithms [CRA93]. A technique to further speed-up this algorithm [CRA96] uses a subset of all data, defined by a range of data values. Some optimizations, such as early ray termination, can not be applied to this method, as it is not ray-driven. An alternative implementation of the splatting algorithm that uses a ray-driven approach [MUE96] is however capable of using the same optimizations as used in ray casting algorithms.

The algorithm introduced by Gross and Lippert [GRO95], [LIP95] computes analytic solutions of the ray intensity integral through a single wavelet by slicing its Fourier transform and by back-

projecting it into the spatial domain. The resulting slices are RGBa textures. Due to the similarity of the basis functions, the computation of the texture map has to be figured out only once for each 3D-mother wavelet. Hence, the final volume rendering procedure turns out to be a superposition of self-similar, transparent and colored textures, supported by modern hardware accumulation buffers.

CHAPTER

5

ISO-SURFACE VOLUME RENDERING

A new technique as far as volume rendering is concerned is iso-surface volume rendering [BOS98]. This algorithm has some conceptual similarity with voxel based surface rendering algorithms in the sense that both methods intend to visualize an iso-surface in the 3D data field. Due to the lack of a clear definition of this iso-surface however, it is hard to compare the accuracy of these methods. Hence, a definition of the iso-surface in a discrete three dimensional data field will be introduced first in the following section.

5.1 Iso-surfaces in volumetric data

An iso-surface is, by definition, the surface in a (continuous) field *F(x,y,z)* at which the value equals a pre-defined iso-value. When an iso-value *Iso* is given, the equation *F(x,y,z) = Iso* will give a relation in *x*, *y* and *z*, which describes the corresponding iso-surface.

In the case of a volumetric dataset however, we have to define an iso-surface in a discrete data field $D(n_x,n_y,n_z)$. A reconstruction filter *R(x,y,z)*, that might be interpolating or approximating as described in section 2.3, can be used to compute the values of the discrete data field at arbitrary *(x,y,z)* locations. This implies that this (continuous) reconstruction filter implicitly extends the discrete data field to a continuous data field $D_r(x,y,z)$. Hence, it is possible to define the iso-surface in a discrete field $D(n_x,n_y,n_z)$ as the iso-surface in this extended continuous data field $D_r(x,y,z)$. This extended field, and hence the iso-surface, is determined by the data values in the discrete field on one hand and the reconstruction filter of choice on the other hand. This iso-surface is equal to the implicit surface defined by $D_r(x,y,z)=Iso$. It is clear that this iso-surface is not the same as the 'iso-surface' extracted with the marching cubes algorithms. When a linear reconstruction filter is used, only the nodes of the triangles lie exactly on the real iso-surface.

Figure 5.1 shows two iso-surfaces in a dataset with 3x3x3 voxels (8 voxel cubes) in which the center voxel has a value of one and the other voxels are zero. This dataset will be called the single voxel dataset. The iso-surfaces shown are iso-surfaces in the 3D impulse response of the reconstruction filters. The left image shows the iso-surface in the data field reconstructed with the tri-linear interpolation function and the right image shows the iso-surface in the data field reconstructed with a quadratic reconstruction function.

Although the linear reconstruction function is linear in one dimension, the iso-surface in a tri-linearly interpolated data field is a third order function within a voxel cube. While the iso-surface is continuous, the orientation of the surface normal of the iso-surface is discontinuous at the boundaries of a voxel cube of the discrete data field. This is caused by the discontinuity of the derivative of the linear re-sampling function.

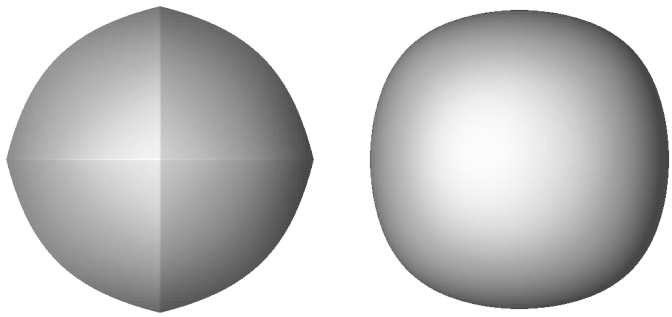


Figure 5.1 Iso-surfaces in the single voxel dataset created with tri-linear interpolation (left) and a quadratic reconstruction function (right)

The quadratic B-spline reconstruction function (described in section 2.3) has a continuous, piece-wise linear, derivative in one dimension. When applied in three dimensions on a discrete data field, this function results in a continuous iso-surface with a continuous surface normal even at the voxel boundaries as can be seen in the right image of Figure 5.1. This response gives a natural surface gradient for a single voxel as well as for iso-surfaces in arbitrary data fields.

Figure 5.2 shows three iso-surfaces in the cone dataset for three different reconstruction functions. In the left image, the discontinuity of the congruent gradient of the tri-linear interpolation function is clearly visible. In the middle image, the gradient is continuous because the derivative of the Catmull-Rom spline interpolation function is continuous. The overshoot generated by this interpolation function is also clearly visible. The right image shows the iso-surface of a cubic-spline interpolation function with parameters B and C equal to zero (see equation 2.6). This function does not satisfy the sixth condition as described in section 2.3. It can be seen that an interpolation method that does not satisfy this condition is unsuitable for (three-dimensional) visualization.

By choosing a reconstruction function and an iso-value, the location of the iso-surface is uniquely defined and independent of the viewing transform, even in perspective view or with an extreme zoom factor. This surface does hence not suffer from the ambiguity that is a big problem in the marching cubes algorithm when multiple triangles lie in the same voxel cube. For these reasons, the

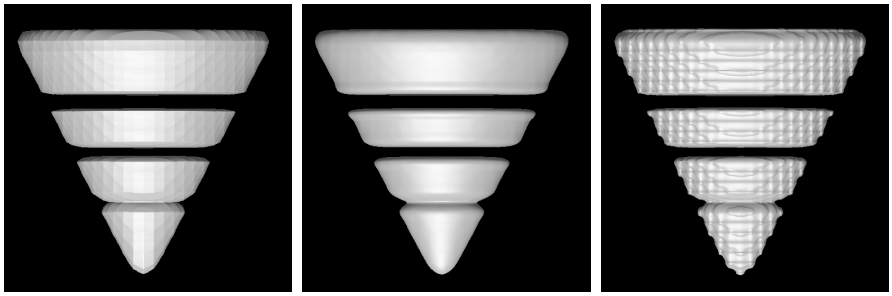


Figure 5.2 Iso-surfaces of the cone dataset with tri-linear interpolation (left), Catmull-Rom spline interpolation (middle) and an unsuitable cubic-spline interpolation function (right)

iso-surface will be used as the reference surface to investigate spatial errors.

5.2 Iso-surface volume rendering

The main difference between iso-surface volume rendering and traditional volume rendering methods is that iso-surface volume rendering aims to find the intersection point of a ray with the iso-surface between the re-sampling locations on this ray instead of calculating the opacity at re-sampling locations. The opacity and gradient are calculated at the location of the intersection.

When tri-linear interpolation is used, the intersection points can be found by finding the roots of a third order polynomial function. The grey-value at an arbitrary (*x, y, z*) location inside a basic voxel cube can be described as $L_3(A..H, x, y, z)$, in which *A..H* are the voxel values. The *x* location on a ray through this sub-volume can be parameterized as $x = x_0 + \lambda(x_1 - x_0)$, in which x_0 is the *x* coordinate of the location where the ray enters the sub-volume and x_1 is the *x* coordinate of the location where the ray exits the sub-volume. In a similar way, also the *y* and *z* locations can be written as a linear function of λ. These values can be used to rewrite the tri-linear interpolation function as a third order polynomial for each line segment. By calculating the roots of the difference of this polynomial and the iso-value, the exact intersection locations can be calculated. Calculating the roots of a third order function is however a extremely computational intensive operation.

An alternative approach can find the roots of the polynomial by using an iteration algorithm. Although the finding of the roots can be much faster, the parameters of the polynomial will still have to be calculated, which will now become the bottleneck. When higher order reconstruction filters are used, the complexity will increase dramatically, making also this approach not suitable for high speed applications.

A combination of a sampling algorithm with iteration can be used to find a very accurate approximation of the intersection point without the need to re-sample the data densely. The iteration algorithm can be started with any pair of samples: (x_1, y_1, z_1) and (x_2, y_2, z_2), at subsequent locations k_1 and k_2 on a ray (i, j), provided that the iso-value *Iso* is enclosed between the re-sampled data values at the locations k_1 and k_2 as shown in Figure 5.3. The (x, y, z) location of the intersection of a ray (i, j) with the iso-surface location, can be estimated within a small error-bound through (repeated) calculation of the intersection point of the iso-value *Iso* with a straight line connecting the two re-sampled data-values, as shown in Figure 5.3. This iteration-process is known as the regula-falsi iteration method.

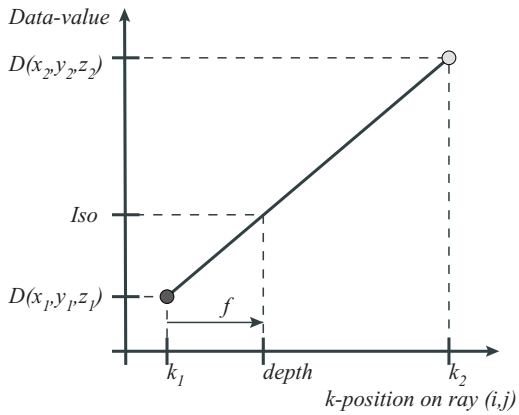


Figure 5.3 The location of the intersection of a ray with the iso-surface is estimated from two samples of which one has a (re-sampled) data-value below the iso-value, and the other has a data-value above the iso-value

A straightforward implementation of this algorithm searches for two adjacent samples on a ray, such that one has a data value below the iso-value and the other one has a value above the iso-

value. The iso-surface will be located between these two samples. From the data values and the iso-value, the depth can be calculated using the straight-line approximation. This approach gives the following equations:

$$f = \frac{Iso - D(x_1, y_1, z_1)}{D(x_2, y_2, z_2) - D(x_1, y_1, z_1)} \tag{5.1}$$

$$depth = k_1 + (k_2 - k_1) \cdot f \tag{5.2}$$

In these equations, f is the location in between the two samples where the intersection is found ($f \in [0, 1]$).

Note that the 1D variant of this formula is used in the marching cubes algorithm to find the intersection points of an iso-surface with the edges of an elementary cell of the voxel-grid. In the regula-falsi iteration method, this formula is used for the calculation of the new base point.

When the distance between k_1 and k_2 is large, this straight-line approximation is not reliable. Using a very small distance between samples on the ray will however significantly increase the rendering time. Hence, it is important to select the largest initial re-sampling distance that guarantees convergence to any iso-surface present. When the two points k_1 and k_2 are found, an iterative algorithm can be used to find two points k_3 and k_4 nearer to the iso-surface. Although the regula-falsi method will converge quickly when the distance between two points is sufficiently small, the performance can be poor when this distance is large. Therefore, another iterative algorithm with a guaranteed convergence factor, such as bi-section, can be used to derive the points k_3 and k_4 from k_1 and k_2 to come close enough to the iso-surface to take advantage of the fast convergence of the regula-falsi iteration algorithm. A speed advantage, available on most computers, is related to the fact that the bisection steps do not need division operations, whereas a regula-falsi step uses one division.

Any other kind of iteration can be used as well to narrow down the initial interval. By defining a maximum distance between k_3 and k_4, the accuracy can be chosen freely. In this way, a trade-off can be made between accuracy and rendering speed.

The initial re-sampling distance $(k_2\text{-}k_1)$ should be chosen with care. As mentioned, a small initial distance would lead to increased

rendering time. When a re-sampling distance equal to or even larger than the minimal sampling distance of the voxel grid is chosen, it is possible that image details will be missed. This can also occur in the super-resolution method when the re-sampling distance on a ray is too large. Levoy's method will suffer less from this problem because even image details smaller than a voxel will be visualized as a vague cube of 2x2x2 voxels. Although it is not possible to miss these details, this visualization will not give any insight in the shape of these small objects.

In [TIE98], an algorithm is described that also aims to find the intersection point of a ray with the iso-surface. In this algorithm, the possibility to miss fine details is reduced by oversampling locally. Oversampling can however not guarantee that small details will never be missed, while the computation time is increased significantly. Another problem with this algorithm is that the root is searched in an interval that may contain multiple different functions. When for instance two samples are taken at both sides of a boundary of a cell using tri-linear interpolation, the function on the interval between the samples is a piece-wise third order polynomial with a discontinuous derivative at the boundary of the cell.

In the iso-surface volume rendering method, these problems are solved by taking samples at the intersection points of the rays with a unit cell consisting of eight voxels. In this way, even the smallest image details will not be missed as long as one of the rays intersects these details. When using a low image resolution, it is hence possible that none of the rays will hit a very small object (smaller than the pixel size). In this case, this small object will not be visible. This means that the smallest object visible is about the projected pixel size in the volume. Increasing the image resolution or zooming in on details will make even the smallest objects visible, as will be shown with the rendering of a single voxel.

Because the samples are taken at the boundaries of the cell, the function on the interval between these points and the derivative of this function are both continuous functions. This makes this method more suitable for an iterative search for the root of this function.

When the volume is sampled at the intersection points of the rays with the unit cells, the initial sampling distance within each cell

will vary from ray to ray, while the initial sampling distance on a ray will vary from cell to cell. Hence, it is possible that the number of iterations necessary to guarantee the maximum distance between k_3 and k_4 will vary as well. The average initial sampling distance will be in the order of the voxel distance.

Besides the guarantee that image details will not be missed when a ray intersects these details, the sampling at the intersection points with the unit cells also has a speed advantage. Because these intersection point lie on a voxel plane, the initial re-sampling function is a two-dimensional function, which is more than twice as fast as a three-dimensional re-sampling function in the case of a tri-linear interpolation and up to more than four times as fast in the case of cubic-spline interpolation. Only the re-sampling in the iteration phase has to be done using the full three-dimensional re-sampling function.

When the iso-surface location is known for all rays, true surface-gradients can be calculated instead of volume gradients. In this way, the surface will be shaded in accordance with the estimated shape. When the shape is estimated accurately, this surface gradient will be equal to the congruent volume gradient at the surface.

More details of the iso-surface volume rendering algorithm will be given in section 7.2.5 about spatial accuracy of the method. Details about implementation will be given in Chapter 8, while some examples of applications on medical data will be given in Chapter 9.

CHAPTER

6

SIGNAL THEORETIC ASPECTS OF DATA ACQUISITION

As discussed in section 2.2, the grey-value data from medical scanners is sampled in the acquisition process. According to the Nyquist criterion, the data should be sampled correctly to avoid aliasing. In most (medical) acquisition devices, the acquired data is band-limited by the point-spread function (PSF) of the acquisition device. Using the modulation transfer function (MTF), which is the Fourier transform of the PSF, the highest signal frequency and hence the lowest sample frequency to satisfy the Nyquist criterion without additional low-pass filtering can be determined.
In this chapter, the point-spread function, the modulation transfer function and the noise properties of two medical data acquisition devices will be described: the CT and the MR scanner.

6.1 Computed Tomography (CT)

In CT imaging, data is acquired by shooting x-ray beams through the object being measured. The number of photons that are absorbed by the object can be calculated from the number of detected photons at the other side of the object. This detection is achieved by a row of detectors that transform the number of photons into a measurable voltage. In this way, a set of 512 to 1024 measurements, called a profile, is obtained for each angular position. To be able to reconstruct a single slice, 600 to 1500 profiles per rotation are measured, resulting in a total of 307,000 to 1,536,000 measurements per slice.

Figure 6.1 shows schematically the commonly used third generation CT scanner in which the (one-dimensional) detector array and the x-ray tube rotate together around an object in a single plane, acquiring one slice of data.

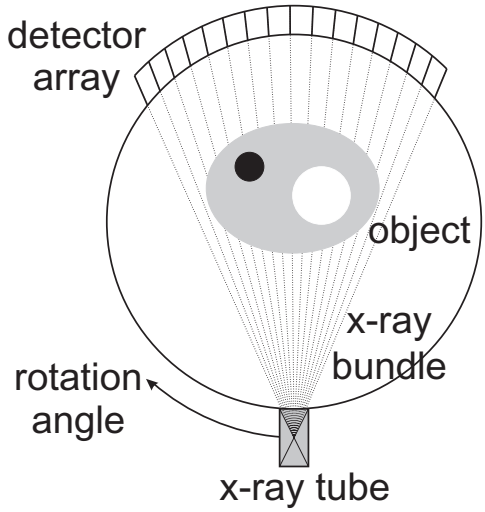


Figure 6.1 Third generation CT scanner

When all measurements in a slice are depicted in a two-dimensional image with the profiles in one direction and the angles in the other, this will result in an image in which a point in the object is depicted as a sine shaped line, as can be seen in Figure 6.2 for a full rotation over 360 degrees. Such a 2D image showing all profiles in a slice is called a sinogram. One profile can be

represented by the grey-values on a sloped line through this sinogram.

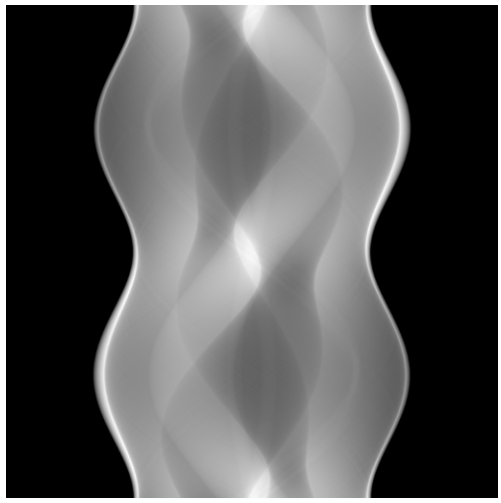


Figure 6.2 Sinogram of one acquired slice from a CT scanner

This figure shows a sinogram that is generated from a Shepp-Logan head phantom image that is often used in CT and other tomographic reconstructions [SHE74]. This greyscale phantom image is a model of the human head that consists of two large ellipses (representing the skull and the brain) containing several smaller ellipses (representing anatomical structures inside the brain). The corresponding original greyscale data is shown in Figure 6.3.

The projection of the image intensity along radial lines is known as the radon transform [HEL99]. The acquired profile image shown in Figure 6.2 is hence the radon transformed grey-value data of the slice shown in Figure 6.3. Different approaches exist to reconstruct the original grey-values in a slice from the measured profiles [KAK88]. One of these approaches is the inverse radon transform.

A commonly used inverse radon transform algorithm is the so called filtered backprojection [RAM71]. This algorithm first filters the profile data using a high-pass ramp filter, the so called Ram-Lak filter. This filter can be windowed to reduce high frequency noise. The grey value data at an output pixel location can then be reconstructed by calculating the integral on a sine shaped path in

Figure 6.3 Shepp-Logan phantom

the filtered sinogram. Because the points on these paths do not coincide with the measured locations, a resampling filter will be necessary. Figure 6.4 shows the sinogram after filtering with a ramp filter.

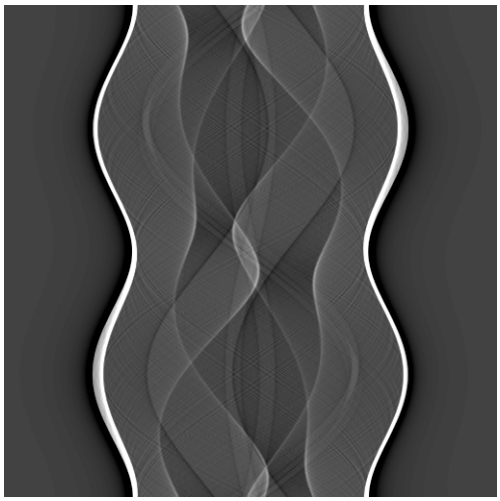


Figure 6.4 Filtered sinogram

Figure 6.5 shows the resulting output image after filtered backprojection. In this example, a low resolution phantom image of 512×512 pixels is transformed to 512 profiles containing 512

samples. This rather low resolution might result in visible artefacts in the reconstructed image. In this example however, the ramp filter is windowed by a hanning window to reduce high frequency artefacts.

Figure 6.5 Result after filtered back-projection

6.1.1 The point-spread function

The (in-slice) point-spread function of a CT system is determined by a number of factors: the size of the focal spot on the x-ray tube anode, the width of the collimator that determines the slice thickness, and the size of the detectors. Due to the divergent nature of the radiation, and also due to the reconstruction method, the PSF is spatially variant with respect to the distance from the rotation centre [RAT92].

Besides this, additional variance is introduced by the scanned object (patient) itself, because scatter (redirected radiation) and beam-hardening (wavelength dependant absorption) locally vary with the material inside the scanned object. In the remainder of this thesis, the spatial variance of the PSF will be neglected.

The PSF of the CT scanner can be described as the convolution of a number of optical transfer functions with a square shape. As discussed in section 2.3, repeated convolution of a square function

will result in a Gaussian function. Hence, the PSF can be approximated by a Gaussian function:

$$PSF(x) = e^{-\frac{x^2}{2\sigma^2}} \tag{6.1}$$

The width w of the PSF is defined as the distance between the two points that have a value that is half the maximum of the PSF. Hence, the width has to satisfy the following relation:

$$e^{-\frac{(w/2)^2}{2\sigma^2}} = \frac{1}{2} \tag{6.2}$$

And therefore,

$$w = 2\sigma\sqrt{2\,ln(2)} \approx 2.35\sigma \tag{6.3}$$

When the width of one of the square optical transfer functions is significantly larger than the others, for instance due to a large focal spot size, the PSF cannot be described by a Gaussian function. In this case, the width w of the PSF will increase, while the parameter σ cannot be used. Therefore, the width w will be used as a parameter to describe the CT point-spread function.

The point-spread function in the scan direction is usually much wider than the point-spread function within a single slice. This is mainly caused by the shape of the detectors in the detector array. The size of such a detector in the direction perpendicular to scanned slice is much larger than the width. The shape of the point-spread function in the scan direction can be controlled by adjusting the width of the x-ray beam collimator. The width of this scan direction point-spread function is also known as the slice thickness, while the sampling distance between two acquired slices is known as the slice distance.

As a result of this deviation of the point-spread function in the slice direction, the three dimensional point-spread function is not point symmetrical. This may be a problem when the acquired volume is used in three-dimensional (visualization) applications. A reason for acquiring thick two-dimensional slices is to increase the signal to noise ratio and contrast. The resulting data is however mainly useful for observing the original acquired slices. The asymmetrical point-spread function makes the data less useful for observing perpendicular planes.

A new generation of CT scanners solves this problem by using a two-dimensional detector array. In a single rotation of the x-ray source, many slices are acquired. This approach can give a much more symmetrical three-dimensional point-spread function. The main drawbacks of this method are the increased noise and the large amount of acquired data. To reduce the noise, the data can be low-pass filtered. When only slices perpendicular to the scan direction are observed, this filtering can be done in the scan direction, again resulting in an asymmetrical point-spread function. For three-dimensional applications however, it might be better to filter the data with a symmetrical three-dimensional low-pass filter to preserve the symmetry of the point-spread function. Because of the symmetry and flexibility of this type of scanners, these scanners are preferable when the acquired data is used for three-dimensional visualization.

6.1.2 Modulation transfer function

Because the Fourier transform of a Gaussian function is also a Gaussian shaped function, the MTF of a scanner with Gaussian PSF can also be described by a Gaussian function, with:

$$PSF(f) = e^{-\frac{f^2}{2\sigma_f^2}} \tag{6.4}$$

The product of the standard deviations of the place and frequency domain satisfy the following equation:

$$\sigma_f \sigma_x = \frac{1}{2\pi} \tag{6.5}$$

Because both the PSF and the data are continuous signals in place domain, the frequency spectrum before sampling is also continuous and has no repetition. Sampling will result in a discrete frequency spectrum that is replicated at every multiple of the sampling frequency. Consequently, signal data above the Nyquist frequency will be aliased. Hence, according to the Nyquist criterion, the sampling frequency has to be at least twice the highest signal frequency.

Because the data is in general not band-limited, the highest signal frequency before sampling is determined by the MTF. The value of the MTF is very small (lower than one percent) for frequencies

higher than $3\sigma_f$. This means that, according to the Nyquist criterion, the sampling frequency has to be twice this frequency, or $f_s \geq 6\sigma_f$. Using equation 6.5, we can calculate the sample distance sd as a function of σ_x:

$$sd = \frac{1}{f_s} \leq \frac{1}{6\sigma_f} = \frac{2\pi\sigma_x}{6} \approx \sigma_x \tag{6.6}$$

This means that for a sampling distance smaller than σ_x, the Nyquist criterion is met. Using equation 6.3, the resulting sampling distance should be at most 0.43 times the width of the point-spread function.

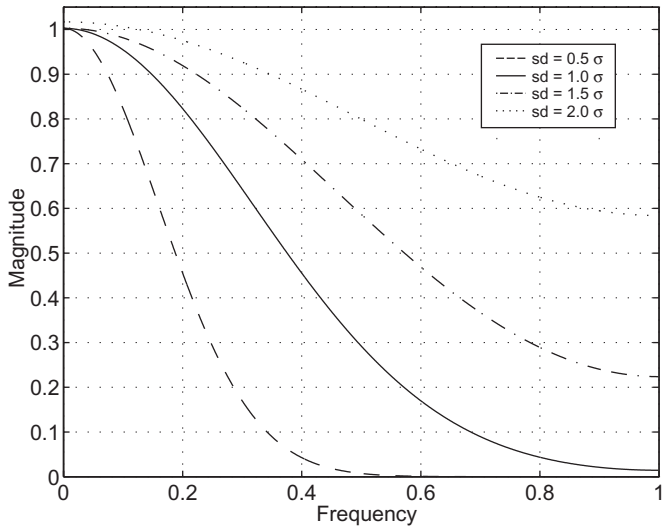


Figure 6.6 Modulation Transfer Function of a CT scanner with a Gaussian PSF after sampling

In Figure 6.6, the MTF after sampling (MTF') is plotted, using the sampling distance as a parameter. The frequency is normalized to the Nyquist frequency, which is half the sampling frequency. Due to folding, the value of the MTF' at half the sampling frequency is twice the value of the MTF at the same frequency. It can be seen that when the condition $sd=\sigma_x$ is met, the MTF has a negligibly small value at the Nyquist frequency. For larger values of the sampling distance, the rather large value at the Nyquist frequency will result in aliasing. When $sd=2\sigma_x$, the MTF has a nonzero value at the sampling frequency ($f=2$). Consequently, even the lowest

frequency components will be influenced by folding, which is visible as the small offset at f=0 in Figure 6.6.

As mentioned before, the PSF cannot be described by a Gaussian function when one of the square optical transfer functions is significantly larger than the others. In this case, the MTF will also be different, as can be seen in Figure 6.7.

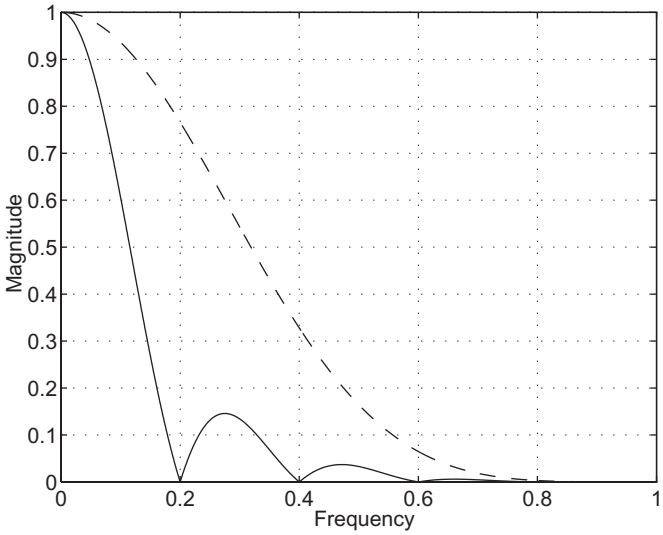


Figure 6.7 Modulation Transfer Function of a CT scanner with a non Gaussian PSF

The resulting MTF is the product of a sinc function and a Gaussian. It can be observed that high frequency data will be more attenuated. This is commonly the case for the point-spread function in the slice direction. In order to obtain maximal spatial resolution, all the square optical transfer functions have to be chosen as small as possible.

6.1.3 Noise

The process of counting photons at the detectors is the source of the noise in CT acquisition, which means that the noise is not affected by the (optical) point-spread function. Consequently, the noise in the measured profiles has a Poisson distribution. This noise is not correlated and therefore has an amplitude that is more or less independent of the sampling frequency, which is also called white

noise. Due to the interpolated filtered back-projection, the noise will become frequency dependent with a Gaussian distribution.

The ideal ramp filter has a frequency response that grows linearly with the frequency. After filtering the profiles, the data is back-projected to reconstruct the final two-dimensional grey-value slices. Different interpolation functions are used to re-sample the measured profiles. For instance, the frequency response of filtering with an ideal ramp filter and re-sampling using linear interpolation is shown in Figure 6.8.

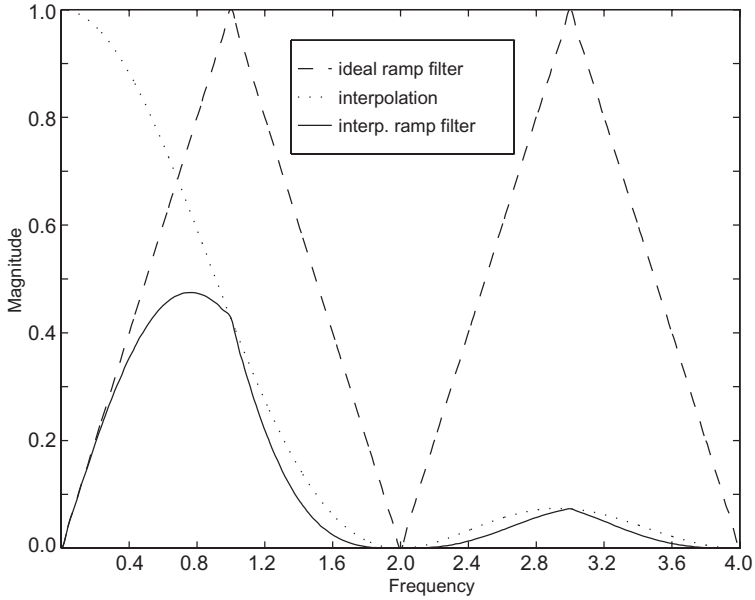


Figure 6.8 Frequency response of filtering the profiles with an ideal ramp filter and linear interpolation

In this figure, the frequency axis is normalized to the Nyquist frequency (π rad/s). Except for the frequency response of the linear interpolation function, the amplitude axis is scaled with the same number. In this example, the data was re-sampled at a four times finer grid, resulting in a four times higher sampling frequency. The re-sampling rate can be very high when a zoom reconstruction is used, which means that small objects are reconstructed to a full image.

The frequency response of the ideal ramp filter shows that lower frequency noise in the profiles is attenuated, while the higher

frequency noise is amplified. To reduce the high-frequency noise, different types of filters are used. Consequently, the frequency spectrum of the noise in the profiles after filtering has a band-pass behavior, as shown in Figure 6.9.

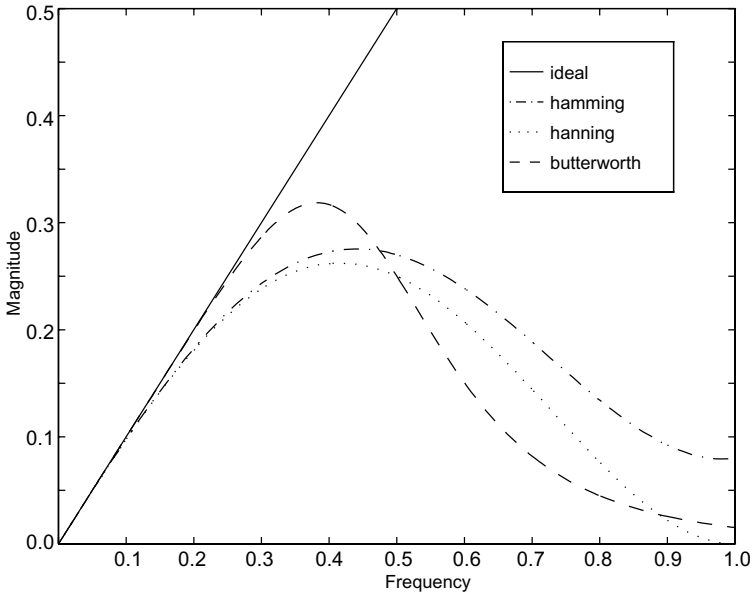


Figure 6.9 Frequency response of some different (windowed) ramp filters

As mentioned before, the data in the profiles is interpolated during the back-projection. Figure 6.10 shows the noise spectrum after filtered backprojection using, for example, a Butterworth ramp filter in combination with linear interpolation.

The high-frequency noise behavior, and consequently the spatial resolution, can be influenced by the choice of the ramp filter. More suppression of the high-frequency noise will also result in a lower spatial resolution, and vise versa.

Figure 6.11 illustrates the noise filtering in the filtered backprojection algorithm. The upper left image shows the acquisition noise in the acquired profiles, that has a Poisson distribution and is not correlated. The upper right image shows that the FFT of this noise is flat. The lower left image shows the filtered backprojection of the noise image. It can be clearly seen that this noise is correlated. Furthermore, the noise is heavily

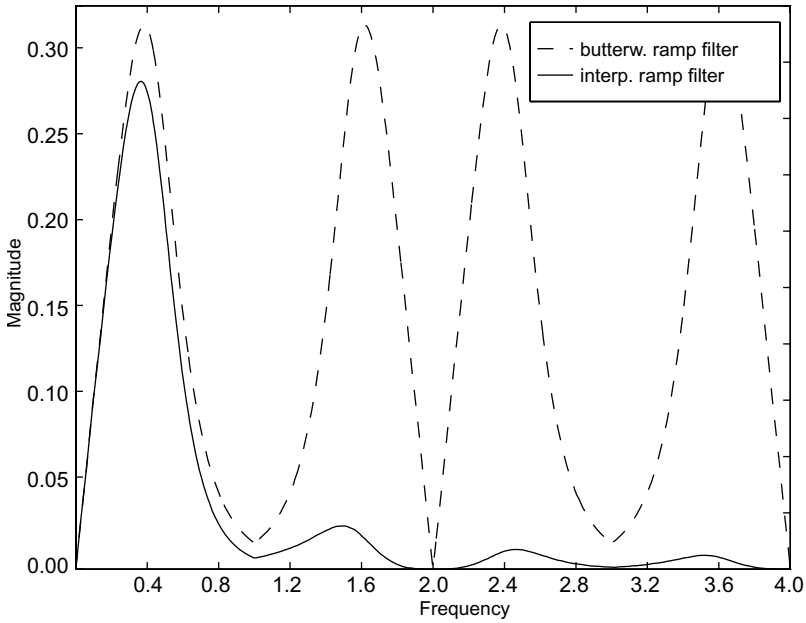


Figure 6.10 Frequency response of filtering the profiles with a Butterworth ramp filter and linear interpolation

frequency dependent, as can be seen in the lower right image that shows the FFT of the filtered backprojected noise. It is clearly visible that the noise after filtered backprojection has a low amplitude at low frequencies (center of the image), and that the amplitude grows with the frequency (towards the edges of the image). As mentioned earlier, an additional low-pass filter can be used to reduce the high frequency noise. In this example, an unwindowed Ram-Lak filter is used.

It is clear that the spatial resolution is limited by the sampling in the profiles and therefore by the detector width. Using zoom reconstruction to enlarge small objects will not enhance the detail of those objects.

6.2 Magnetic Resonance Imaging

Although a MRI scanner [FRI89], [VLA99] uses a completely different acquisition principle, a similar approach can be used to describe the signal theoretic background of this scanner. First, some principles of MRI will be explained.

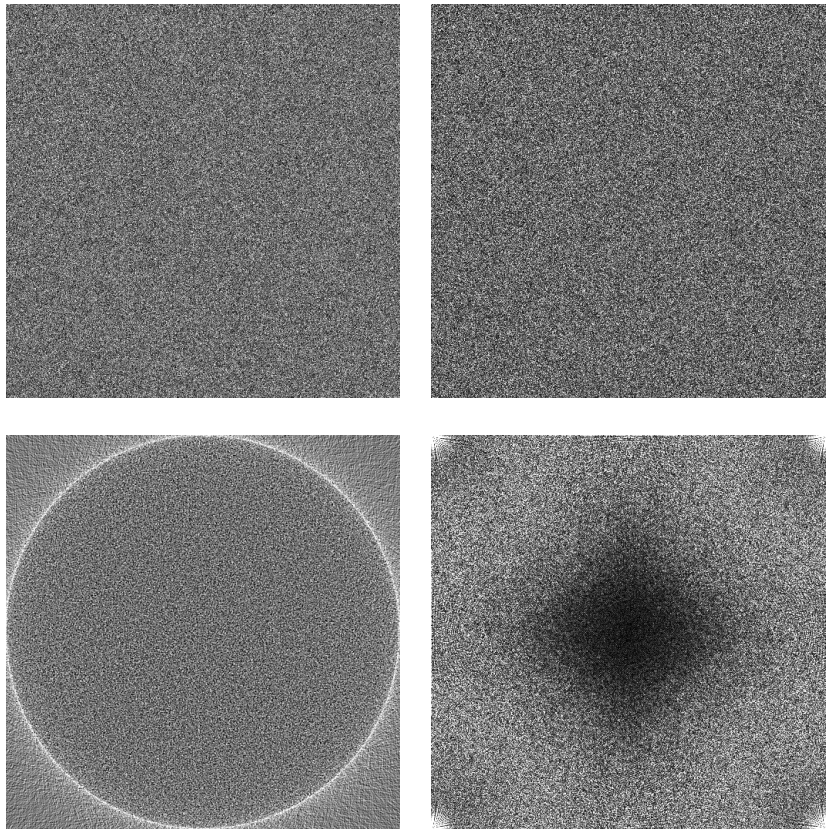


Figure 6.11 Noise in CT reconstruction: poisson noise in the profiles (upper left), FFT of the profile noise (upper right), noise after filtered backprojection (lower left) and FFT of the noise after backprojection (lower right)

The magnetic resonance imaging acquisition method is based on the magnetic field that is caused by spinning nuclei like ^{1}H, ^{13}C, ^{19}F and ^{31}P. The hydrogen nuclei (^{1}H) are commonly used because most tissues in the human body contain high concentrations of these nuclei. When the nuclear spins are directed in a random direction, the magnetic fields of the nuclei will cancel each other out. When a patient is placed in a magnetic field, the nuclear spins will become orientated parallel or anti-parallel to the magnetic field.

A small majority of nuclear spins will be aligned in the low energy parallel direction. This results in a net magnetization in this direction. When energy in the form of radio waves is added to the

patient, the direction of the nuclear spins can change from the low energy parallel direction to the high-energy anti-parallel direction. A radio frequency (RF) pulse that displaces the net magnetic field 90 degrees is called a 90-degree pulse. The frequency of this pulse depends on the gyromagnetic ratio of the nucleus and the strength of the magnetic field. For hydrogen nuclei, the gyromagnetic ratio is 42.6 MHz/Tesla. In a 1 Tesla field, the frequency of the RF pulse should hence be 42.6 MHz.

By adding a gradient field to the main magnetic field, the strength of the magnetic field becomes place dependent. As a result, only the nuclei in a single slice will respond to a certain RF pulse.

Besides the spin around their own axis, the nuclei also turn around the axis of the main magnetic field. Due to a 90 degree pulse, the net magnetization vector rotates in a plane perpendicular to the main magnetic field (transverse magnetization), which can be measured by an antenna. When the RF pulse is turned off, the net magnetic field will realign in the direction of the main magnetic field. This is called T1 or spin-lattice relaxation. T2 or spin-spin relaxation occurs when spins in the high and low energy state exchange energy, which results in a loss of transverse magnetization. Finally, T2* relaxation is caused by dephasing of individual magnetizations. This dephasing is caused by magnetic field inhomogeneity and results in a loss of transverse magnetization at a rate higher than T2 relaxation.

Because the rotation speed also depends on the strength of the magnetic field, a second gradient field can be applied such that the rotation speed depends on the location in the slice. When this gradient field is applied in a short time interval, the difference in rotation speed will cause a place dependent phase shift. In this way, different lines in the slice are phase encoded.

Figure 6.12 shows a multi-echo spin-echo sequence. The 180 degrees RF pulses used to generate the echo are rephasing the spins that have undergone T2* decay. The decline in signal from subsequent echoes reflects T2 decay. The time between the 90 degrees pulse and the moment when the spins are completely rephased is called the echo time (TE). This time is equal to two times the time between the 90 degrees pulse and the first 180 degrees pulse.

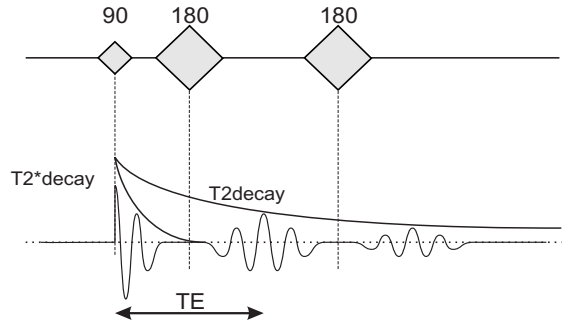


Figure 6.12 Multi-echo spin-echo sequence

The frequency of the signal emitted by a nucleus also depends on the strength of the magnetic field. A third magnetic field is applied to frequency encode the position of a nucleus on a line. In this way, the position of a nucleus in a slice is encoded in the phase and frequency of the signal emitted. Because the first gradient field selects a slice, the second (phase) selects a row and the third (frequency) selects a point on a row, these gradient fields are also referred to as the z, y, and x gradient fields respectively.

In general, different materials have different T1 and T2 relaxation times. Hence, the value of the transverse magnetization at TE depends on the material. Furthermore, the magnetization of a certain material depends on TE. Besides the echo time, the repetition time (TR) also has an influence on the magnetization. The combination of TR and TE determines the contrast and the signal to noise ratio (SNR).

6.2.1 Modulation transfer function

In MRI, the measurements are done in the frequency domain (also known as k-space). This frequency domain data can be transformed to place domain data using a Fourier transform (FT). The grey-value data is obtained by taking the modulus of the resulting complex data.

Consequently, the MTF is flat, up to the Nyquist frequency. To reduce the measuring time, the number of measurements can be reduced, which means that the number of measurements can be smaller than the number of voxels in the grey-value data. In this case, zeros are added to the measured data. Hence, the MTF has

the same shape, although the highest frequency is lower than the Nyquist frequency.

6.2.2 The point-spread function

As the measurements are done in frequency domain, a MR image is obtained by taking the FFT of the measured data. Because the MTF can be described by a square function, the ideal space domain PSF will be a sinc-like function described by:

$$PSF(x) = \frac{sin(\pi \cdot x/ps)}{\pi \cdot x/ps} \tag{6.7}$$

In this equation, *ps* is the pixel size, which is defined as half the distance between the first two zeros of the PSF. The width *w* of the PSF is 1.2 times the pixel size. However, because the pixel size is a commonly used measure for spatial resolution in MRI, it will be used as acquisition parameter for MRI systems.

When the data is acquired in slices, the PSF will have a sinc form in the two directions within the slices, while the PSF in the slice direction can be completely different. In case the data is acquired in 3D, the point-spread function will have a sinc form in all three dimensions. The latter method seems to be favorable for volume rendering.

6.2.3 Noise

The noise in the measured data of MR scanners has a Gaussian distribution. The complex parts, resulting from the two-dimensional fourier transform, also contain (independent) Gaussian distributed noise. Due to the modulus operator, the distribution of the MR image noise depends on the signal value. The background noise for instance, will have a Rayleigh distribution, while the noise for higher signal values tends towards a Gaussian distribution. For lower signal values, the noise distribution is a mixture of Rayleigh and Gaussian distributions.

Because the complex parts after the fourier transform contain noise with a flat frequency spectrum, the frequency spectrum of the MR image noise is also flat. Although the distribution of the noise is signal dependent, the noise is still uncorrelated, and therefore,

the noise spectrum is independent of the signal value and the noise distribution.

Consequently, it is possible to see structure in the MR image noise due to the different distribution properties. This however does not imply that this noise is correlated. In CT, the noise distribution is independent of the signal value, and therefore the same in the whole image. However, the data is correlated due to the filtered back-projection. In this case, it is also possible to see structure in the image. However, this structure is the result of the correlation between the pixels.

As mentioned previously, the signal to noise ratio (SNR) can be influenced by the TR and TE parameters. A larger contrast (signal) means a higher SNR. In this way, the TR and TE times can be chosen to give an optimum SNR for a certain tissue.

CHAPTER

7

ACCURACY

The main objective of volume rendering and voxel based surface rendering methods is the two-dimensional (2-D) visualization of the shape of three-dimensional (3-D) objects in volumetric datasets. When volume visualization methods are used for medical applications, it is highly desirable that the visualized shape is an accurate reconstruction of the object under study. First the accuracy of estimating the shape with an iso-surface will be investigated. This makes it possible to investigate the spatial errors of different visualization algorithms separately.

7.1 Edge reconstruction

The grey-value dataset is the result of convolution of the point-spread function of the acquisition device with an object at discrete (voxel) locations. Due to this convolution, the exact location of the surface of objects larger than the psf width is 'encoded' in the grey-values at voxel locations, while very small objects may be lost. When the data was not sampled in the acquisition process, the location of a step edge can be exactly found using the iso-value method described in section 2.5. When the data is sampled however, an error will be made which will be shown to depend on the location of the samples relative to the edge. In [SMI96], this sample location dependent error was examined for a square point-spread function.

Because in this chapter only the response of a step edge is investigated, the results are not directly applicable to other types of transitions between two tissues. The transition between two soft tissues for instance may also be spreaded over multiple voxels, regardless of the psf. In these cases it is not possible to define a surface that separates the two tissues. Using the same edge detection algorithm on this type of transitions will however give an approximation of a surface in the middle of the transition. This surface may still be useful to analyze the transition. In soft tissues, also the measured values might change within an object. This will also affect the shape of the surface. In this section, the accuracy of the estimated location of an edge in one dimension will be investigated for point-spread functions typically found in practical (medical) scanners.

7.1.1 Edge location estimation accuracy

In Figure 7.1, the edge response of three re-sampling functions on sampled data acquired with a Gaussian psf is plotted for a fixed acquisition sample distance. The acquired samples are marked with an ‘o’.

From this edge response, it is possible to calculate the error made by subtraction of the re-sampled data from the original Gaussian function. This is comparable with the evaluation of interpolation functions by interpolating sampled 2D grey-value images and subtract the result from the original image [BEN95], [MAC96],

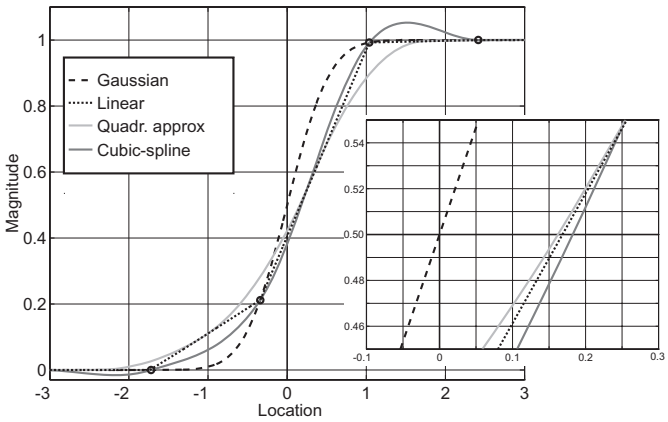


Figure 7.1 Step edge response of the data acquisition and re-sampling chain for a Gaussian point-spread function

[GRE97]. This approach will lead to the conclusion that cubic-spline interpolation is the best and quadratic is the worst image re-sampling technique. However, it is not our goal to make the best reconstruction of the grey-values. Instead, we are interested in the accuracy of the estimated location of the step edge. When the edge location is estimated by the location at which the re-sampled grey-value equals the mean of the values at both sides of the step, as shown in the Figure 7.1 where the mean is 0.5, the largest edge location error is made when cubic-spline interpolation is used, while the edge is best estimated by the quadratic re-sampling function. Linear interpolation performs better than cubic-spline but worse than quadratic.

The previous example showed the results for acquisition with a fixed sample location and a fixed sample distance. When the sample locations are chosen differently, the results will be different. When for instance one of the samples is taken at x=0, the error will be zero for all re-sampling functions. Figure 7.2 shows the edge location error as a function of the sample location for a sample distance equal to the width of the point-spread function. The error is a percentage of the width of the psf. Again, the largest error is made when cubic-spline interpolation is used.

Besides the sample position, the error also depends on the sample distance. To investigate this dependency, only the amplitude of the

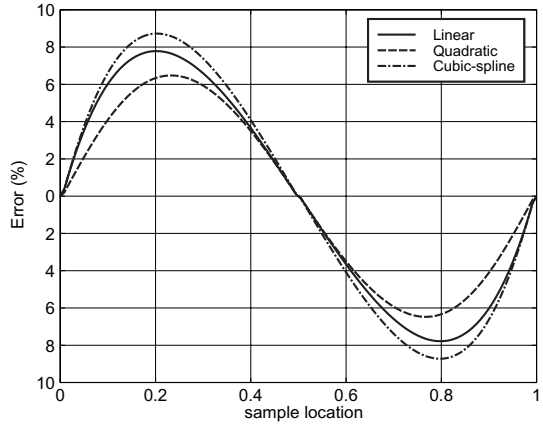


Figure 7.2 Error of the estimated location of the step edge as a function of the sampling location for a fixed sample distance

error will be discussed. Figure 7.3 shows the amplitude of the error as a function of the ratio sample-distance/psf-width.

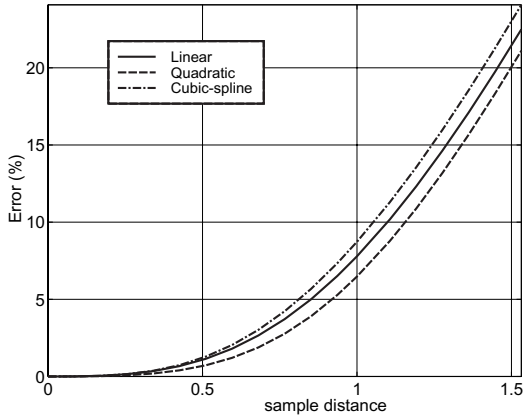


Figure 7.3 Amplitude of the edge location error for different sample distances

It is obvious that the error approaches zero for very small sample distances. The error grows approximately with the third power of the sample distance. In the CT acquisition process, the sample distance can be chosen very small within slices. In the scan direction however, the sample distance is often rather large. Consequently, the amplitude of the edge location error is much

larger in the scan direction than within the slices. This amplitude will become visible in the renderings as ripples on oblique surfaces, as shown in Figure 7.4.

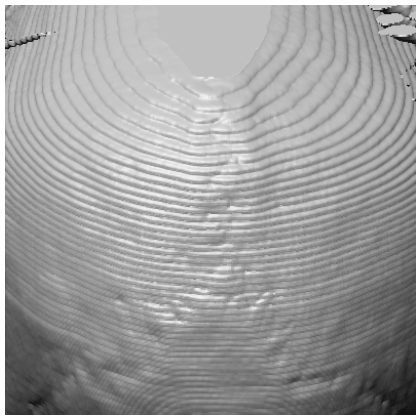


Figure 7.4 Ripple artefacts due to too large sample distance

As discussed in section 6.1.2, the sample distance should be at most 0.43 times the width of the point-spread function to meet the Nyquist criterion. At this sample distance, the systematic edge location error is about one percent of the width, which corresponds to 2.5 percent of the sample distance.

A similar approach can also be used to evaluate the accuracy of the edge reconstruction in MR images. As described in section 6.2.2, the psf in MR acquisition has a sinc shape, while the sample distance is commonly fixed to the pixel size. Figure 7.5 shows the step response of acquisition with a sinc point-spread function and three different re-sampling filters.

Figure 7.6 shows the sample location dependent error for a sample distance equal to the pixel size.

As can be seen in these images, the results for a sinc point-spread function are quite similar to the results for a Gaussian psf. The difference between the re-sampling functions is however slightly larger. Although the sample distance is commonly equal to the pixel size, it is possible to investigate the effect of using a smaller sample distance. The amplitude of the step edge location error as a function of the sample distance is shown in Figure 7.7.

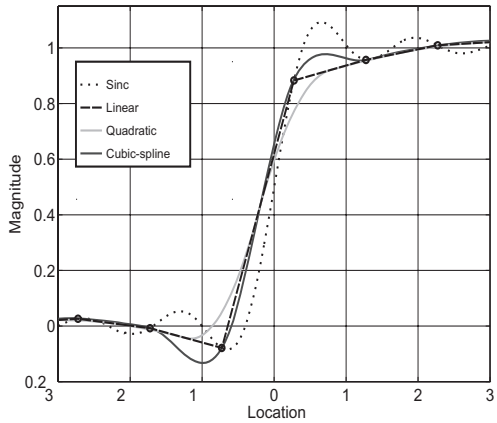


Figure 7.5 Step edge response of the data acquisition and re-sampling chain for a sinc psf

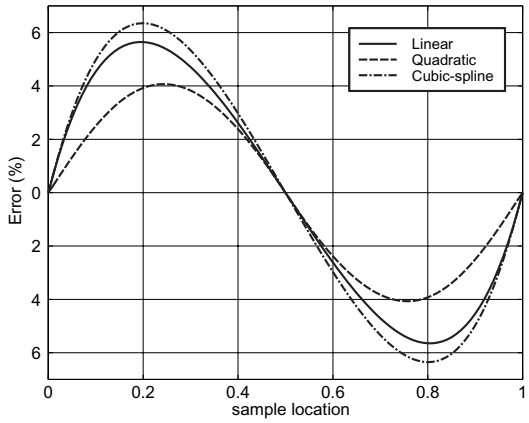


Figure 7.6 Error of the estimated location of the step edge as a function of the sampling location for a fixed sample distance

Although the Nyquist criterion is always met in the case of data acquisition in the frequency domain, the step edge location error is four to six percent of the pixel-size when the sample distance equals the sample size. This error is hence about five times larger than in CT when the Nyquist criterion is met. The Nyquist criterion can hence not be used as a criterion for the determination of the step edge location error.

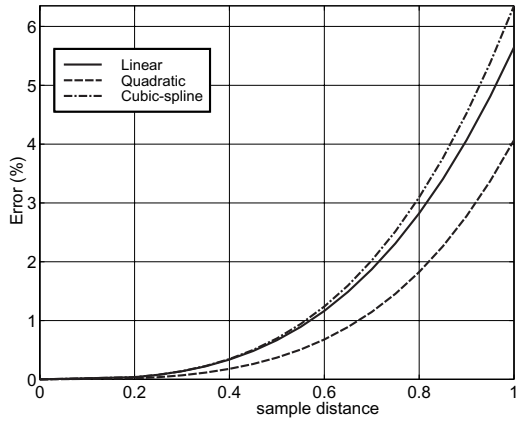


Figure 7.7 Amplitude of the edge location error for different sample distances

When the sample distance is reduced by a factor two, the error will be reduced by a factor 8 to 10 of the pixel-size, which corresponds to a factor 4 to 5 of the sample distance. This can be achieved in MR imaging by zero padding the frequency domain data. When the frequency domain data is zero-padded, not only the ripple in the surface due to the systematic error becomes smaller, but also the frequency response of the total acquisition and re-sampling chain will improve, resulting in much more detail. This is mainly caused by the low-pass filtering of the re-sampling function. Due to the zero-padding, the sampling frequency and hence the Nyquist frequency will become higher and less high frequency data will be filtered out.

Figure 7.8 shows two iso-surfaces of an MR phantom. The left image shows the iso-surface in the original data, while the right image shows the result after zero-padding the frequency domain data. The severe ripples in the left image due to the systematic surface estimation error are clearly visible. In the right image, this error is almost zero and hence hardly visible. The small ripples in the surface that are visible in the right image are caused by the sinc-shaped point-spread function. This same ripple is also present in the left image, but is hardly visible here due to the much larger surface estimation error ripple. Furthermore, the higher frequencies are much more attenuated in this image resulting in a slightly smaller ripple due to the point-spread function. Small,

high frequency, details are however also filtered out which is very well visible at the loss of detail at the screw holes.

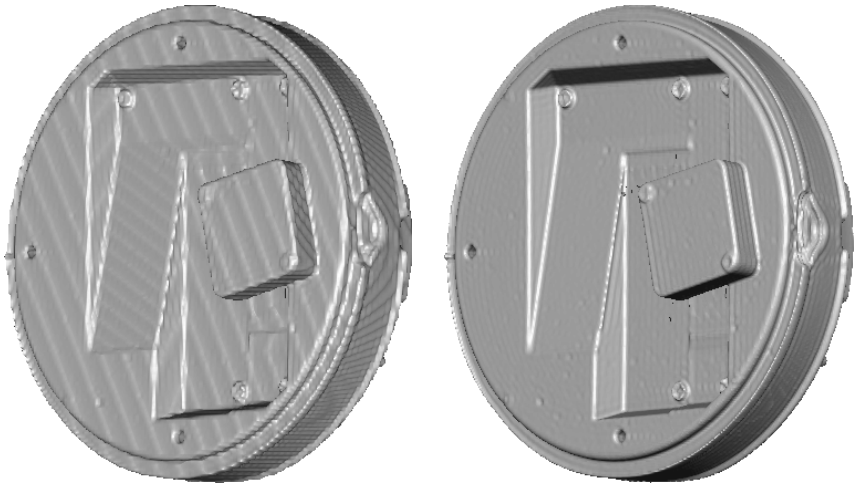


Figure 7.8 Iso-surface before (left) and after zero-padding

It is not possible to use an ideal low-pass filter to re-sample the place domain data. This is caused by the complex modulus operator in the reconstruction process. In this process, the phase information is lost, which means that the original frequency domain data can not be reconstructed. Figure 7.9 shows the response of a step edge before and after zero-padding. The black circles indicate the locations at which the samples are taken. Even when sinc-interpolation is used, the original data cannot be recovered. The grey circles indicate the extra samples that are available when zero-padding with a factor of two is applied. In this case, even a strong low-pass filtering re-sampling filter might yield better high-frequency results than any re-sampling filter without zero-padding.

In this example, a step is shown from zero to one. Due to the overshoot, negative values are possible, as shown in Figure 7.5 and Figure 7.9. This is however not possible in MR data. The results shown are hence only usable for step edges at which the acquired value can not be negative. The influence of the modulus operator on the estimated step edge location is shown in Figure 7.10.

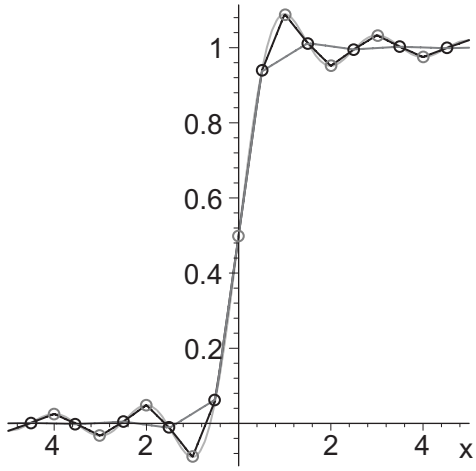


Figure 7.9 Step response of data acquisition and re-sampling

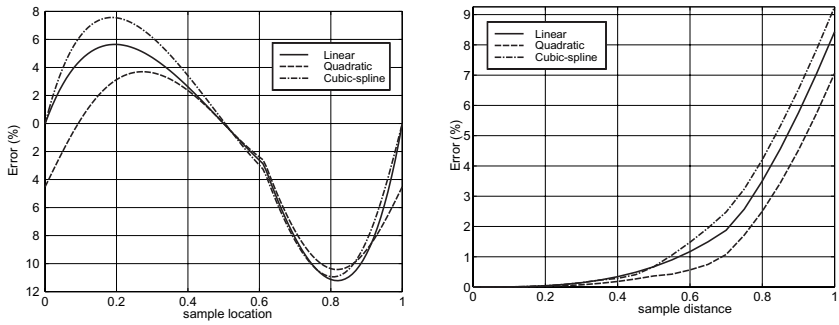


Figure 7.10 Error of the estimated location of the step edge as a function of the sampling location for a fixed sample distance (left) and amplitude of the error for different sample distances (right)

Besides the smaller error, the quadratic re-sampling function also has the advantage of being less sensitive to noise in the data compared to the linear and cubic-spline functions.

7.1.2 Gradient accuracy

In the previous section, only the accuracy of the estimated location of a one dimensional step edge was considered. When these edge detection algorithms are applied to surfaces in 3D, the sample location dependent error will result in visible artefacts. In many volume rendering algorithms however, not the estimated surface,

but the shading of this surface using the volume gradient is visualized. The desired value of the gradient is the derivative of the step response at the actual step edge location. This derivative is equal to the maximum value of the point-spread function that occurs at x=0. When the surface gradient is estimated using one of the gradient calculation methods described in section 2.4, the estimated gradient will in general not equal the desired gradient. Similar to the estimated edge location, the estimated gradient will depend on the sampling distance, the sample location and the chosen function.

Figure 7.11 shows the desired and estimated gradients using three different gradient estimation algorithms for a Gaussian point-spread function. The sampling distance chosen is equal to the standard deviation.

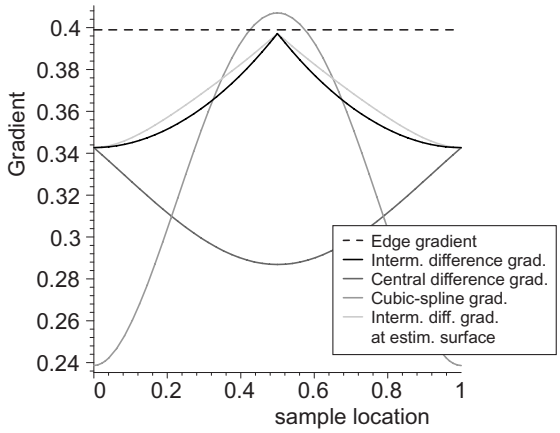


Figure 7.11 Estimated gradients at different sample locations

As mentioned, the desired gradient is equal to the value of the point-spread function at x=0, in this case $1/\sqrt{2\pi}$. This value is depicted in Figure 7.11 as a red line. The three gradient estimation methods that are shown in this figure are the central difference method, the intermediate difference method, and the cubic-spline gradient method. In case of the central and intermediate difference methods, the gradient at an arbitrary location is found by linearly interpolating the calculated gradient at (or between) the samples.

Because the central difference and the intermediate difference gradient methods give the same results at the sampling locations,

the estimated gradient at the edge location is identical when a sample is taken at the edge location. The value of the gradient found in this case is however not the correct derivative of the (continuous) step response. The cubic-spline based gradient algorithm results in a very inaccurate approximation of the derivative.

When the edge location lies exactly between two samples (sample location = 0.5), the gradient estimated by the intermediate difference methods is a much more accurate approximation of the derivative, while the central difference method results in a gradient estimation that is a worse approximation of the derivative. The cubic-spline gradient algorithm also gives much better results. Due to the overshoot of this function, the estimated gradient can become larger than the desired derivative.

Because in applications the exact edge location is unknown, the gradient will be calculated at the (wrong) estimated edge location. Figure 7.11 also shows the result of calculating the intermediate difference at the estimated edge location using linear interpolation. It can be seen that the results are identical when the edge location coincides with a sample or lies between two samples. In these cases the estimated edge location is identical to the exact edge location. At other sample locations, the error in the estimated edge location results in a better approximation of the derivative of the step response.

Because the estimated gradient also depends heavily on the sample location, visualization using the estimated volume gradient will also result in severe ripple artefacts when the sample distance is large.

When a fixed volume gradient is used for shading, the choice of the re-sampling function to find the surface location hardly influences the visual appearance. Only the shape of the surface will differ noticeably, while the shading of the surface will be almost identical.

7.2 Spatial error bounds for the visualization of iso-surfaces

To be able to evaluate the accuracy of the many different rendering algorithms, an error bound has to be defined. As mentioned in the previous section, several error criteria introduced in the past are based on the reconstruction of the grey-values at sub-voxel locations [BEN95], [MAC96], [GRE97]. The common approach is to sub-sample a given two-dimensional image and re-sample this sampled image to recover the original image size. Hereafter, the grey-values of the original image are compared with the grey-values of the re-sampled image for instance by calculating the mean square error. This is a valid approach to compare the accuracy of re-sampling functions in applications where the grey-values have to be displayed at a higher resolution than the image size, details of the image are blown up or the image is subjected to a transformation.

Another approach to compare the re-sampling functions used in different visualization methods is based on frequency domain criteria [BEN95]. In this approach, the ideal low-pass filter is used as the point of reference. The re-sampling functions considered are transformed into the frequency domain and compared with the ideal low-pass filter.

However, these error criteria do not take into account the way the re-sampling functions are embedded in different volume visualization algorithms. This aspect has much more impact on the visual appearance than the choice of the re-sampling function. Hence, these error criteria are not generally usable to evaluate the accuracy of the visualized shape.

As shown in the previous section, the amplitude of the sample location dependent error and hence the size of the ripple in the surface is smaller for the quadratic B-spline re-sampling method than for linear interpolation, while cubic-spline interpolation has an even larger error. Because grey-value and frequency domain based error criteria normally predict the opposite, it is clear that these error criteria can not be used in the case of finding the location of a step edge.

An alternative approach is to compare the final two-dimensional rendered image with a reference image [COH98]. This is a good

approach to evaluate the perceived image quality when a lossy simplification is used, when a reference lossless image is available. In this case, the best method is the method that has the least impact on the rendered image. In the case of volume visualization however, there exist many visualization algorithms and a reference is not (yet) available.

Therefore, a new spatial error bound will be used that uses the maximum distance between the location where the surface is calculated and a reference surface as a measure for spatial accuracy. This allows an evaluation of the error bound, independent of the observation angle. The errors in traditional volume rendering algorithms are mainly caused by calculating the gradient at the wrong location. This gradient should be calculated on the reference surface. By using the definition of the spatial error bound, the maximum distance between the location where the gradient is calculated and the reference surface can be determined.

Because these error bounds are defined in relation to the voxel distance, perspective projection does not influence the spatial error bounds. Although voxels near the viewpoint can be enlarged extremely in perspective projection, a visualization method with a negligible spatial error is able to visualize an iso-surface in these voxels without visible artefacts.

The errors made in the different volume visualization methods described in the previous section have a different effect on the visualization of a given object. This will be illustrated with the rendering of the isometric 32×32×32-voxel data set of a cone with slits of ½, 1 and 2 grid units that was shown in Figure 4.2. When such a small dataset is rendered at high resolution, errors as small as the voxel distance can be easily visualized. To be able to see the effect of a spatial error much smaller than the voxel size, a dataset containing a single voxel will be used for the visualization of this error.

Because the location where the re-sampled data is equal to a fixed (iso-) value is a good estimation of the location of a step edge, the extension of this definition to the three-dimensional case, which is known as an iso-surface (see section 5.1), will be taken as a point of reference for the shape of the object.

Figure 7.12 shows a two-dimensional example of the acquisition and reconstruction process. Although the shape of the test object

int his figure looks like the cone in Figure 4.2, these are not identical. Figure 7.12 is merely used to illustrate the whole traject of data acquisition and reconstruction. The renderings in this chapter are all based on the dataset in Figure 4.2.

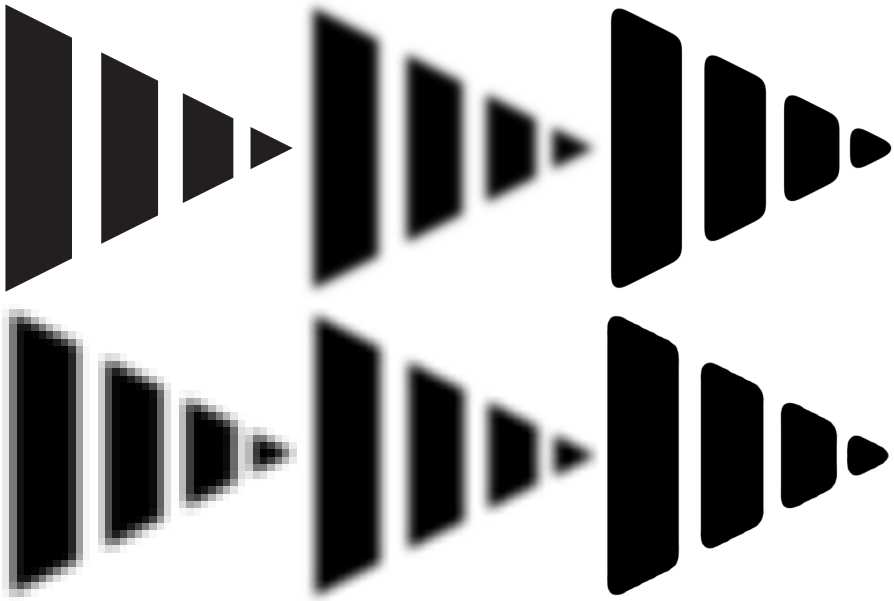


Figure 7.12 Test object before acquisition, after filtering with a Gaussian psf, after thresholding without sampling, after sampling, after re-sampling, and after iso-surface reconstruction

In the upper left image of Figure 7.12, the original test object is shown. In the upper middle image, the same object after filtering with a Gaussian point-spread function is shown. The upper right image shows the result of thresholding this filtered object. The difference between the upper left and the upper right image is entirely caused by the low-pass character of the point-spread function. As a result, sharp edges are rounded. When the data is sampled, a discrete data field, as shown in the bottom left image, will be the result. This data is the input for the various visualization algorithms. By re-sampling and thresholding the data to find the edges of the object, as described in the previous section, a good estimation of the thresholded object before sampling can be made. The lower middle image shows the data after re-sampling, while the lower right image shows the final approximation of the object. Any difference between the upper right and the lower right image is caused by the sampling/re-

sampling process. It was shown in the previous section that these errors can be made sufficiently small by increasing the sampling frequency in the acquisition process. Because these errors have been investigated separately, it is now possible to take the resulting iso-surface after re-sampling as a reference object. The differences between the original object and the reference object are caused by the acquisition process. The goal of the spatial error bound in this section is to determine the maximum distance between the boundary of the reference object (the iso-surface) and the boundary of the visualized object for all visualization algorithms described in Chapter 4.

Due to the sampling in the acquisition process, sampling artefacts may be introduced. Small details in two adjacent acquired slices may for instance be unconnected in reality, while in the visualization process these details may be connected due to non optimal sampling and interpolation. Oversampling the data will therefore not help in reducing these sampling artefacts. These artefacts are caused by the acquisition process and can hence be reduced by adjusting the sampling parameters during acquisition. Besides the sampling artefacts caused by the scanner, the re-sampling used in many visualization algorithms introduces additional re-sampling artefacts. Some of these artefacts are directly related to the spatial error bound described in this chapter.

As not all volume visualization methods are able to visualize a sharp surface, the spatial error bound has to be extended to handle these cases as well. In this case, the location where the non-linear opacity function is applied can be considered the location of the surface. This definition is easily handled and characteristic for the spatial error found in all the rendering methods described. Provided that the iso-surface defined in section 5.1 is accepted as the golden reference, the errors produced by the basic volume rendering methods are all related to the fact that the opacity and the gradient were calculated at locations close to the iso-surface, but not exactly on the iso-surface.

7.2.1 Voxel space volume rendering

Due to the interpolation of the opacity in the ray casting algorithm described in section 4.2, sharp transitions will be visualized as a vague transition from opaque to transparent within a unit cell. In

the rendering, this will become visible as small vague cubes with a size equal to the voxel size. Figure 7.13 shows a rendering of the cone dataset with the voxel space volume rendering method using a very small re-sampling distance of 1/10th of the voxel distance.

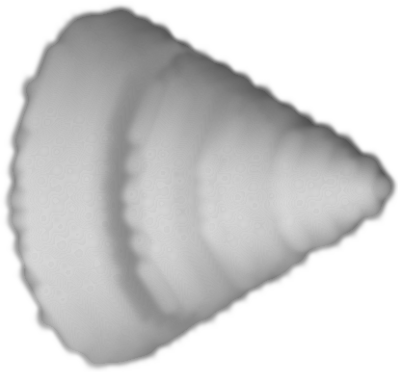


Figure 7.13 Voxel space volume rendering of the 32×32×32 voxel dataset of a cone with small slits

It can be seen clearly that even with this very small k distance this method is not able to accurately render surfaces. Because decreasing the re-sampling distance both on the rays as well as between rays has little effect on the perceived image quality, a re-sampling distance that is equal to or slightly smaller than the voxel size is commonly used. This means that a 32×32×32-voxel dataset is generally rendered as a 32×32 image. Higher resolution images can be obtained by interpolating this low resolution two-dimensional image. Because splatting based methods project voxels on the screen, this is also applicable to these methods. For this reason, the rendering times reported for the voxel space volume rendering algorithms heavily depends on the number of voxels in the dataset.

The opacity and color in the voxel space volume rendering methods are calculated at the voxel locations. As in our definition of the spatial error bound these locations are considered as being the location of the estimated surface, the spatial error in voxel space volume rendering is maximal for a point of the iso-surface located in the centre of a unit cell. When for instance a single voxel is rendered opaque while the surrounding voxels are fully transparent, the surface would be visualized as a vague cube. This

visualization will not change when the shape of the iso-surface is changed, for instance by changing the iso-value.

The spatial error bound is hence equal to the distance between the centre of the voxel and the edges and hence of the order

$$E_{voxel} \approx (\sqrt{dx^2 + dy^2 + dz^2})/2 \tag{7.1}$$

In this equation, dx, dy and dz are the voxel-distances in the x, y and z directions. It is assumed that the distance between re-sampling locations is equal to or smaller than the voxel distances. In that case, the error bound is not severely influenced by the re-sampling process. If the re-sampling distances grow larger, this distance will influence the error bound and eventually even determine the error bound.

When the shear-warp algorithm is used to speed up the ray casting method, the re-sampling distance is fixed for a given viewing angle. When the rays are aligned with the voxel grid, the re-sampling distance is equal to the voxel size. When the data is rotated under the worst case angle of 45 degrees, as shown in Figure 7.14, the re-sampling distance on the rays are increased to $\sqrt{dx^2 + dy^2 + dz^2}$. When the voxel dimensions are equal in the three directions, this distance is 1.73 voxels large. This large distance will lead to clearly visible artefacts at this worst case angle.

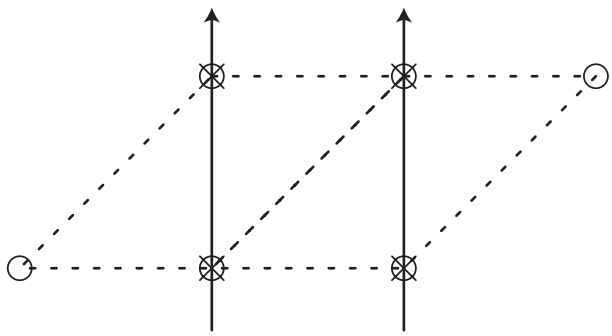


Figure 7.14 The shear-warp algorithm introduces additional artefacts due to a viewing angle dependent re-sampling distance, that can be 1.73 voxels large at an angle of 45 degrees

To reduce sampling artefacts in the ray casting algorithm and especially in the shear-warp algorithm, the corresponding authors propose using a smooth opacity function. This will result in even

larger fuzzy areas and hence in an increase of the error bound. The visibility of these re-sampling artefacts can however be effectively reduced by this blurring approach.

In accordance with earlier findings, the spatial error does not decrease when the opacity field is oversampled in the k direction. It can be seen that the spatial error can only approach zero when the voxel distances go to zero. This implies that the data field should be oversampled to improve the accuracy, which was also described by Levoy [LEV88] as a method to improve image quality. The re-sampling distance should however always be smaller than the (oversampled) voxel size, and should hence also go to zero. To improve the error bound by a factor of two, this approach will lead to increased dataset sizes by a factor of eight, and hence the memory requirements and the rendering time will also increase with a factor of eight. To reduce the error bound by a factor of ten, the dataset will increase by a factor of 1000. It is clear that while this approach can be used on very small datasets of 20x20x20 voxels, it will lead to a giant explosion of data when typical medical datasets are to be rendered with such a small error bound.

7.2.2 Super resolution volume rendering

The perceived quality of the images produced by the voxel space volume rendering algorithms just described is rather low. In the ray casting algorithm, this is mainly caused by the classification at voxel locations. In the super resolution volume rendering algorithm, this classification is done at the re-sampling locations on the ray, which significantly influences the visual appearance.

Figure 7.15 shows two renderings of the cone dataset using the super resolution volume rendering method. The left image was rendered with a re-sampling distance equal to the voxel distance, while the right image was generated using a re-sampling distance ten times smaller. Note that at this resolution, the cone rendered with the super resolution volume rendering technique using a re-sampling distance equal to the voxel distance appears as a stack of 2-D slices. This is caused by the volume gradient calculation at discrete depth locations, which leads to discontinuity of the gradient when for two neighboring rays (pixels) the surface is found at different depth locations.

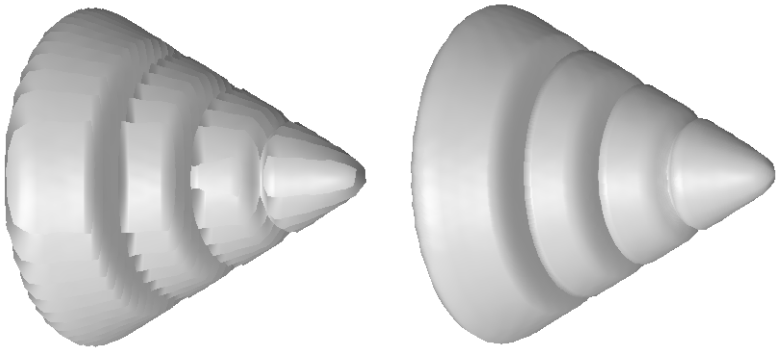


Figure 7.15 Visualizations of the cone dataset using the super resolution volume rendering method with a sampling distance equal to the voxel distance (left) and with a sampling distance ten times smaller

To reduce these re-sampling artefacts, an opacity transfer function can be used that makes the slices semi-transparent depending on the grey-value. This has the same effect as blending the edges of adjacent slices, as shown in Figure 7.16. This approach, often referred to as anti-aliasing, will blur the edges of objects as well. Consequently, the visualization of a sharp surface will not be possible. Furthermore, because more gradient calculations per ray are necessary, the gradient calculations will significantly influence the rendering time or even become the main contribution to the rendering time.

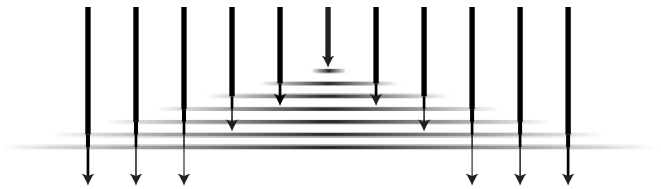


Figure 7.16 Anti-aliasing to reduce visibility of artefacts

The spatial error in super resolution volume rendering is maximal when the ray intersects the iso-surface just behind an integral k value. In this case, the surface is found at the next sample location. The spatial error bound is hence of the order:

$$E_{super} \approx \sqrt{dx_k^2 + dy_k^2 + dz_k^2} \tag{7.2}$$

In this equation, dx_k, dy_k and dz_k are the increments in the x, y and z directions corresponding with a unit step in the k direction. This spatial error bound can be clearly seen in the left image of Figure 7.15 as the distance between two adjacent slices.

The spatial error approaches zero when the re-sampling distance on the rays goes to zero. For the super resolution method, a high over-sampling rate in the k direction is hence sufficient to approach a negligible spatial error, as can be seen in the right image of Figure 7.15. In this image, the spatial error bound is decreased by a factor of ten at the cost of increasing the number of samples and therefore the rendering time by a factor of ten. The size of the dataset and hence the memory requirements are however not influenced with this approach. The spatial error bound is therefore mainly bounded by the desired rendering speed. The need to decrease the re-sampling distance to obtain a high spatial resolution can make the algorithm costly when high resolution/ quality images are required.

7.2.3 Voxel based surface rendering algorithms

Figure 7.17 shows the exact iso-surface in a dataset that consists of a single voxel using a tri-linear interpolation for a fixed iso-value. The left image in Figure 7.17 shows one cross section of this dataset. In this image, the iso-surfaces for nine different iso-values using tri-linear interpolation are visible as nine iso-contours. The right image shows how one of these iso-contours (contour 1) would be estimated by two different voxel based surface rendering algorithms. The marching cubes based approaches find the intersection of the iso-surface with the voxel grid. The surface description is made by connecting these intersection points to create polygons. This is shown in the cross section in Figure 7.17 as contour 2. An alternative approach also finds the intersection of the iso-surface on diagonals. When this calculation is just based on the value of the two points connecting the diagonal, the intersection found is not the actual intersection with the iso-surface, shown as contour 3 in Figure 7.17.

It may be clear from this figure that the approximation of the iso-surface with a polygon that interconnects the iso-values at the voxel grid gives a considerable spatial error. Furthermore, the

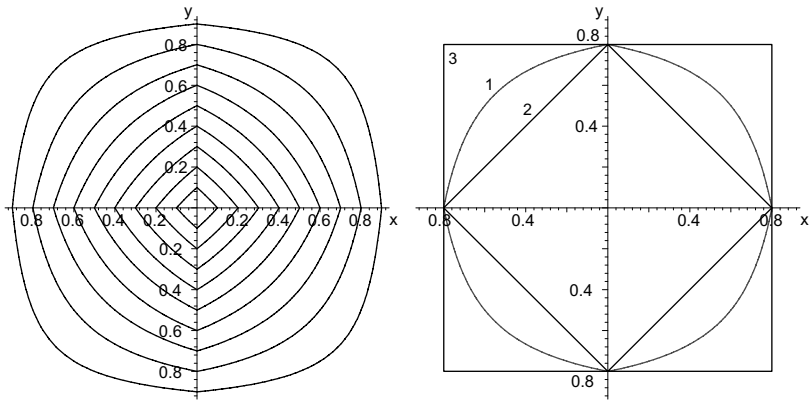


Figure 7.17 Iso-contours of the single voxel dataset (left) and the straight line approximation of one of the contours using two different iso-contour extraction methods

direction of the surface gradient is hardly useful for the estimation of the shape of small objects.

The spatial error in voxel-based surface rendering methods, that approximate the iso-surface as a set of polygons, is maximal for iso-surfaces with a high curvature, like the single voxel iso-surface. We assume a sphere with a radius of one voxel is the worst case surface. This surface will be estimated as a diamond by marching cubes based algorithms. For an isometric dataset, the distance between the sphere and the diamond will be $1-\sqrt{3}/3$ times the voxel size. This suggests that, for an almost isometric dataset, the spatial error is of the order:

$$E_{surf} \approx (\sqrt{dx^2 + dy^2 + dz^2})/2.4 \tag{7.3}$$

The spatial error of the marching cubes algorithm is hence smaller than the error made with voxel based volume rendering methods. The observed error can be significantly smaller when the congruent gradient is calculated at the vertices of the triangles and Phong normal interpolation is used.

The spatial error bound can be reduced by super-sampling the dataset. This will reduce the size and increase the number of triangles generated. As a result, surfaces with a high curvature can be better approximated. The same is true for methods that find a triangulated surface in a continuous field, which can be applied to the continuous re-sampled data field. Instead of super-sampling

the whole dataset, it is also possible to re-sample the dataset locally, using an error criterion to control the number of triangles generated. When the desired spatial error is very small, these approaches can however generate a large number of triangles for cells with a high curvature surface.

Methods that reduce the number of polygons generated with the marching cubes method may further increase the spatial error bound, depending on the reduction algorithm used.

7.2.4 Splatting

The splatting footprint, used in the splatting algorithm, should be capable of mapping a single voxel on the screen, eventually taking the local opacity into account. A possible footprint measuring 6x6 pixels was shown in Figure 4.4.

The opacity and the gradient, that determine the opacity and the color to be used for the splatting footprint, are computed from the opacity and the gradient at the center of the elementary grid-cell surrounding the voxel or at the voxel location. The spatial error equals hence the spatial error of the direct volume rendering method:

$$E_{splat} \approx (\sqrt{dx^2 + dy^2 + dz^2})/2 \tag{7.4}$$

7.2.5 Iso-surface volume rendering

The goal of iso-surface volume rendering is to visualize an iso-surface with negligible error bound, without significant loss in rendering speed. Figure 7.18 shows a rendering of an iso-surface in the cone dataset generated using the iso-surface volume rendering method with the quadratic re-sampling function.

The spatial error made in iso-surface volume rendering fully depends on the selection of proper initial (sample-) locations, the iteration algorithm chosen and the convergence criterion used. Figure 7.19 shows in a cross section how one of the iso-surfaces of Figure 7.17 is estimated using the iso-surface volume rendering method at a viewing angle of 45 degrees.

In this figure, some of the viewing rays are depicted as dotted lines. The implementation shown uses n bi-section steps followed by one

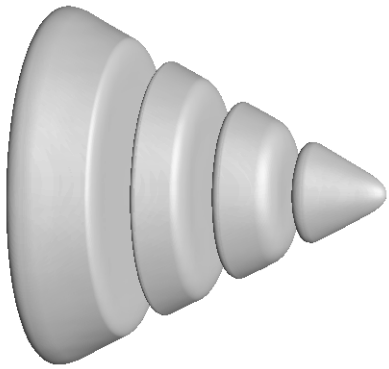


Figure 7.18 Rendering of the cone dataset generated with the iso-surface volume rendering algorithm using quadratic interpolation

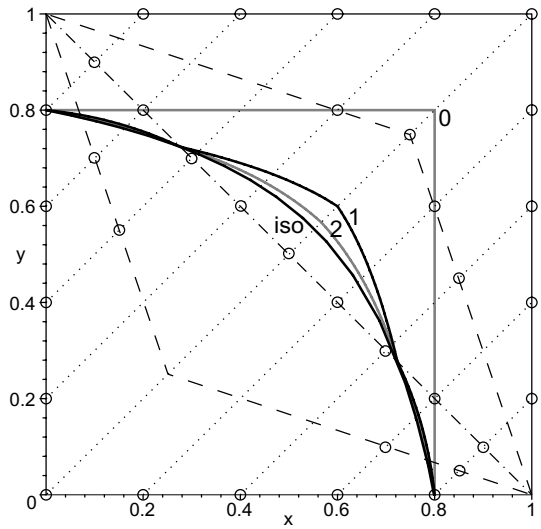


Figure 7.19 Using bi-section, a very accurate estimation of the iso-surface can be made using only a few iteration steps

regula-falsi step. Without bi-section *(n=0)*, the exact iso-surface, shown as the curve labelled 'iso', is estimated using the regula-falsi step on the samples at the intersection points of the rays with the voxel grid. This results in the curve labelled '0'. At the first bi-section step, the grey-values are also calculated on the dash-dotted bi-section line. When only one bi-section step is used, the

estimation error is zero at the intersection points of this bi-section line with the actual iso-surface. On both sides of these intersection points, the regula-falsi step in the surface location estimation is based on different re-sampling points. On one side, the estimation is based on the re-sampled values at the intersection points of a ray with the bi-section line and the voxel grid in front of the surface, while at the other side the estimation is based on the re-sampled values at the intersection points of a ray with the bi-section line and the voxel grid behind the surface. The surface estimated when only one bi-section step is used is shown in Figure 7.19 as the contour labelled '1'. At a second bi-section step the grey-values are also calculated on either one of the dashed lines, depending on the grey-value on the first bi-section line. The iso-surface location can now be estimated from four different combination of intersection points. The estimated iso-surface after two bi-section steps and one regula-falsi step is shown in Figure 7.19 as the contour labelled '2'.

It is clear that a very small spatial error can be reached using just a few iteration steps within one cell of the data field. In every step of the bi-section process, the spatial error is decreased by a factor of two. The regula-falsi step further decreases the spatial error bound with a factor that is generally much larger than two.

The spatial error bound is hence of the order:

$$E_{iso} \approx (\sqrt{dx^2 + dy^2 + dz^2})/2^{n+1} = \varepsilon_{residual} \tag{7.5}$$

In this equation, n is the number of bi-sections. It is possible to supply a spatial error bound ε, after which the iso-surface volume rendering algorithm stops the bi-section process when the distance between two samples is smaller than this error bound. The actual error will however be much smaller due to the regula-falsi iteration step.

Figure 7.20 shows the iso-surface of a single voxel after one, two, three, and six bi-sections followed by the regula-falsi step. To be able to see the errors in the estimated surface location, the shading is based entirely on this estimated surface and not on volume gradients. In this figure, the intersection of the bi-section planes with the iso-surface are clearly visible as contours on the surface. At these contours, the spatial error is zero. While the generated surface is continuous, the direction of the surface normal is not. This is caused by switching the pair of base points at the

intersection contours. It can also be seen in this figure that every bi-section step adds a contour with zero error between two zero error contours.

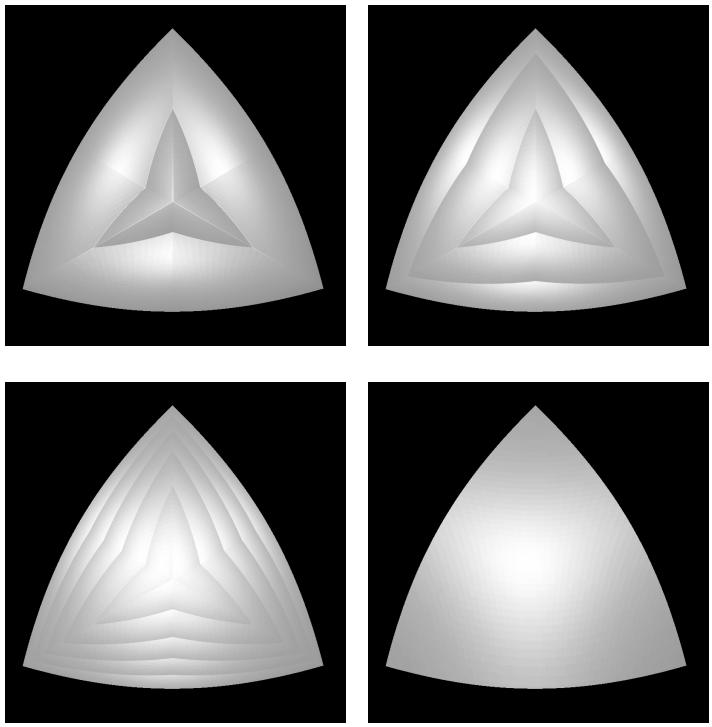


Figure 7.20 Iso-surfaces of the single voxel dataset after 1, 2, 3, and 6 bi-sections

Although the spatial error is very small after four bi-sections, the shading of the estimated surface can give undesirable artefacts due to the discontinuity of the surface normal. However, measurements have shown that the rendered image does not significantly change when the number of bi-sections is increased from two to eight when the shading is based on the volume gradient at the estimated iso-surface. Hence, only two to three bi-sections are necessary to avoid viewpoint dependent artefacts.

Furthermore, the intermediate difference volume gradient is also continuous at the boundaries of a unit cell. Because the surface gradient, and hence the congruent volume gradient, is discontinuous at these boundaries, intermediate difference volume

gradients can be used to generate a smoother surface when tri-linear interpolation is used.

Shading true surface gradients or congruent volume gradients can however be very helpful to analyze the true shape of the estimated iso-surface as shown in Figure 7.21. While the intermediate difference volume gradient generally can be used to improve the perceived image quality of an iso-surface in a tri-linearly interpolated data field, it is obvious that a non-congruent volume gradient is not an accurate estimate of the surface normal of the iso-surface in this figure.

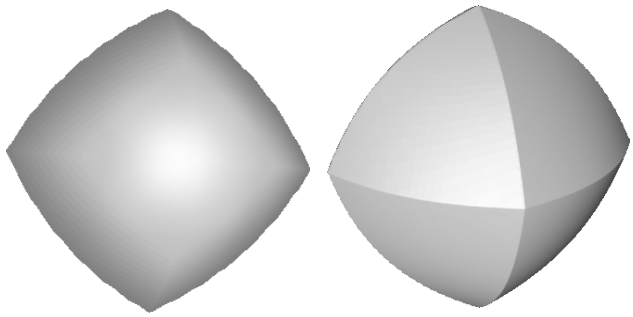


Figure 7.21 Rendering of a single voxel dataset shows that the intermediate difference volume gradient (left) is not an accurate approximation of the congruent gradient of tri-linear interpolation

7.2.6 Survey

The inherent spatial error of several volume rendering algorithms has been computed. Table 7.1 includes an extra column for the isometric case with $\Delta x = \Delta y = \Delta z$. Designers of volume rendering software and hardware can use these results to find algorithms

with improved spatial accuracy and/or to apply the theory to existing implementations to evaluate their spatial accuracy.

Table 7.1 Survey of methods with their associated spatial error and the action to be taken to improve the method

Algorithm	Error	Isometric	Super-sample
Voxel space VR	½ Δv	0.87 Δx	Dataset
Shear warp	Δv	1.73 Δx	Dataset
Splatting	½ Δv	0.87 Δx	Dataset
Super res. VR	Δs	Δs	Rays
Marching cubes	0.42Δv	0.73 Δx	Dataset
Iso-surface VR	$\varepsilon_{\text{residual}}$	$\varepsilon_{\text{residual}}$	

In this table, Δv is the voxel size, which is $\sqrt{dx^2 + dy^2 + dz^2}$ and Δs is the sample distance, which equals $\sqrt{dx_k^2 + dy_k^2 + dz_k^2}$. The $\varepsilon_{\text{residual}}$ for the iso-surface volume rendering method is an error bound that can be chosen freely. With all methods, an error bound that approaches zero can be achieved. By using a super-sampling factor N in the first five methods, the error bound is reduced to $1/N$. In the super resolution algorithm, this is obtained by super-sampling the rays, which will increase the complexity of the algorithm with a factor N. In the four other methods, the dataset should be super-sampled, which increases both the amount of data as well as the complexity with a factor N^3. When the desired accuracy is for instance 1/10-th of the voxel distance, the complexity of the shear warp algorithm will increase with a factor 1000, the complexity of direct volume rendering and splatting will increase with a factor 125 and the complexity of marching cubes will increase with a factor 74 compared to straightforward implementation without super-sampling the data. To reduce the error bound from one voxel distance to 1/10-th of the voxel distance, the complexity of the super resolution algorithm will increase with a factor 10. With the iso-surface volume rendering algorithm, the same result can be obtained without significantly increasing the complexity.

The iso-surface volume rendering algorithm is hence the most accurate algorithm to visualize iso-surfaces in 3D datasets.

Especially when high accuracy is needed, for instance for medical applications, this algorithm can be much more efficient than the other algorithms that can only achieve the same high accuracy at an extremely high cost.

Dedicated implementations of the algorithms analyzed may produce better results, when additional (error) criteria are taken into account. One important feature of spatial error criteria and spatial error bounds is that the conditions under which these errors can be diminished are now better understood.

The medical applications described in Chapter 9 all use the iso-surface volume-rendering algorithm on raw output data of a CT or MRI scanner. Neither pre-filtering [SAK95] nor upsampling of the data was needed to obtain images free of sampling artifacts.

It is good practice to switch from one re-sampling function to another to get a better understanding of the scene at hand. The linear interpolation function, with its discontinuities in the gradient, can be used to visualize the boundaries of the voxel-grid. This gives an understanding of the resolution of the data. The cubic spline interpolation function on the other hand, can be used to verify that the high frequency content in the data is sufficiently preserved. Note that this results in different iso-surfaces of choice and that it has not been the subject of this chapter to find criteria, which of those is preferred. The identification and subsequent elimination of spatial errors, made in various volume rendering applications, has been the main objective.

CHAPTER

8

IMPLEMENTATION AND OPTIMIZATIONS

The previous chapters gave a description of various volume rendering algorithms. Most of these algorithms have been implemented and much of the current research is focussed on optimizing the algorithms to obtain higher rendering speeds. Many of these optimizations will however result in visible artefacts, which is undesirable especially in medical applications. Iso-surface volume rendering is a new algorithm with high accuracy and free of re-sampling artefacts. To be able to use this algorithm in medical applications, the rendering speed should be high, while the visual appearance should not be affected by optimizations. This chapter will describe some techniques to improve the speed of various parts of the algorithm without affecting the image quality.

8.1 Binary shell

One of the most important optimizations that does not influence the final image is minimization of the number of calculations. When rays are cast through the whole volume, many sub-volumes will not contribute to the rendering. When an iso-surface is visualized, only the basic volumes that are intersected by the iso-surface have to be processed, while all other volumes can be skipped.

A cell is intersected by an iso-surface when it contains both voxels with a value higher than the iso-value and voxels with a value lower than the iso-value. When only these cells are processes, not only the empty (transparent) space around an object to be visualized can be skipped, but also the opaque volume within this object. Consequently, only the shell of an object has to be processed to find the iso-surface.

A fast and easy way to find and store the information about the presence of an iso-surface is based on binary volumes. Using one bit it is possible to distinguish between empty and non-empty cells. By simply thresholding the data, a binary volume indicating what voxels have a value higher than the iso-value can be created. Simple logic operations on the resulting binary volume makes it possible to calculate the presence of an iso-surface for many sub-volumes simultaneously. Since only the sub-volumes lying in the shell of an object are selected to be processed, the resulting binary volume will be referred to as the binary shell. Figure 8.1 shows a two-dimensional equivalent example of the computation of the binary shell.

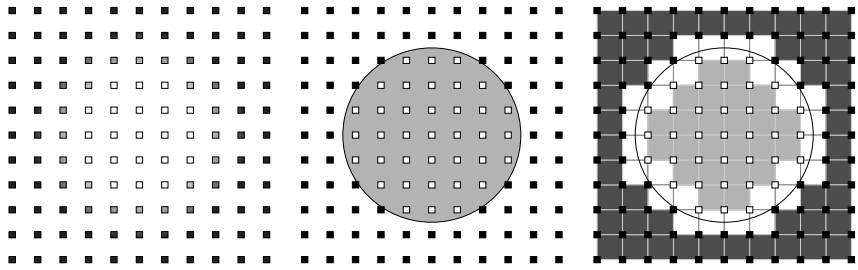


Figure 8.1 Two-dimensional equivalent example showing the grey-value data (left), the binary volume (middle) and the binary shell

First, the grey-value data shown in the left image is thresholded. The voxels with a value below this threshold are depicted as black squares in the middle image of Figure 8.1, while voxels with a value above the threshold are depicted as white squares in this image. The object to be visualized is shown as a grey disk. The corresponding iso-surface is also shown as a black circle. It is obvious that voxels with a value higher than the iso-value lie within this circle while the other voxels lie outside the circle. The dark-grey squares in the right image of Figure 8.1 indicate the cells that lie completely outside the object and can hence be skipped. The other cells all contain a part of the desired object. To find the iso-surface, only the cells that are intersected by this iso-surface have to be processed. The light-grey squares in the right image, that lie completely inside the object, can hence also be skipped. The remaining cells that contain a part of the iso-surface are depicted as (larger) white squares. It can be seen in this image that the iso-surface indeed intersects all selected cells and that each cell selected for processing contains both a voxel with a grey-value higher as well as a voxel with a grey-value lower than the iso-value. The resulting binary shell is the smallest set of cells that have to be processed in the iso-surface volume rendering algorithm.

For visualization of the iso-surface, only the binary shell has to be processed. A similar approach can also be used to minimize the number of computations when for instance a maximum intensity projection is calculated and a volume of interest is selected using a lower threshold or some other kind of segmentation. In this case however, all cells that contain a part of the object should be processed. In the two-dimensional example in Figure 8.1, this means that also the light-grey cells in the right image have to be processed. For objects with a large volume, this will increase the number of computations drastically.

8.2 Rendering speed

Using the iso-surface volume rendering algorithm in combination with the binary-shell, high rendering speeds can be obtained without any other pre-computation. To be able to compare this algorithm with other visualization methods, the speed of iso-

surface rendering on some freely available medical and non-medical datasets will be discussed.

The datasets used for the speed measurements are available from the University of North Carolina at Chapel Hill.

This set of datasets contains an MR dataset of a human head consisting of 256×256×109 voxels, a CT dataset of a human head containing 256×256×113 voxels and a CT dataset of an engine block containing 256×256×110 voxels.

The measurements are carried out on a 500 MHz Intel Pentium III computer with 128 MB RAM memory. The resolution of the images has a higher impact on the rendering speed than the resolution of the dataset. Therefore, measurements are done at two different image resolutions: 256×256 and 512×512. As the resolution of the datasets is 256×256 for each slice, many visualization algorithms will use a corresponding 256×256 pixel output image because oversampling does not or hardly increase the image quality. With the iso-surface volume rendering algorithm, it is possible to generate very sharp surface edges. A higher image resolution is hence also desirable. It is possible to use a lower image resolution of 256×256 pixels during interaction and a higher resolution of 512×512 pixels images when interaction is stopped. This approach ensures a faster response during interaction, while the image resolution is still very useful for orientation.

Figure 8.2, Figure 8.3 and Figure 8.4 show renderings of the different tissues that are visualized to measure the speed of the iso-surface volume rendering algorithm.

Table 8.1 shows the results of rendering these tissues in full color using tri-linear interpolation and the intermediate difference volume gradient. The rendering times were measured at the angles shown in Figure 8.2, Figure 8.3 and Figure 8.4. For the objects

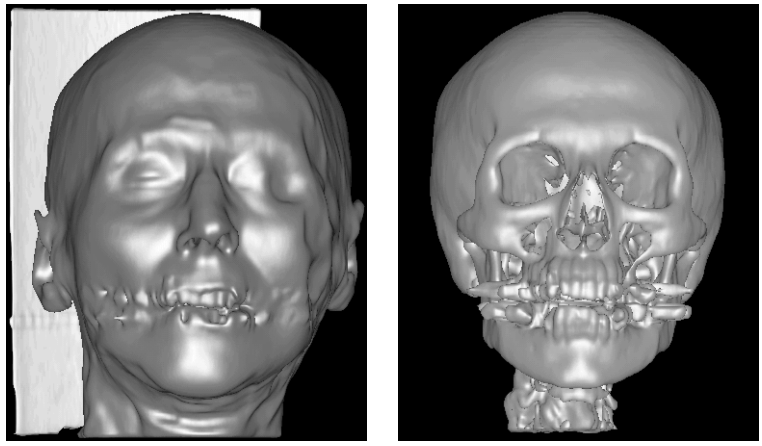


Figure 8.2 Skin and bone tissues in the CT head dataset

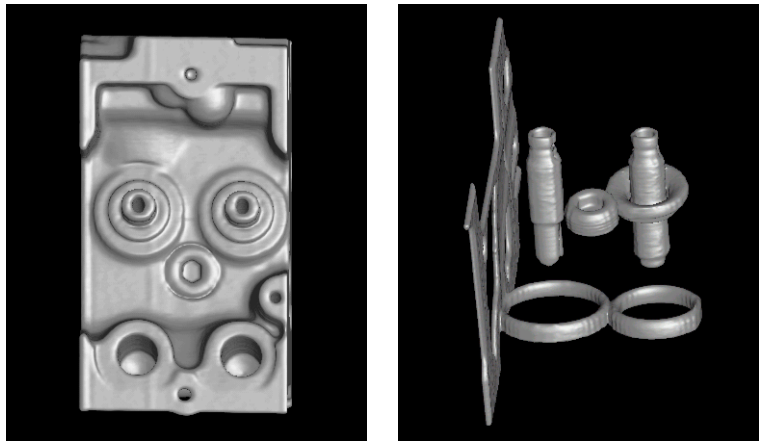


Figure 8.3 Engine block and steel parts from CT engine dataset

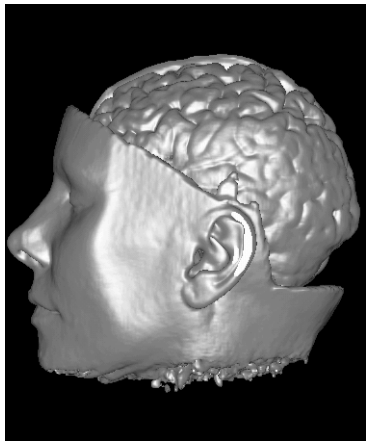


Figure 8.4 MR brain dataset

shown, the rendering speed depends only marginally on the viewing direction.

Table 8.1 Timing of rendering speed

Dataset	256×256	512×512
CThead skin	0.20s - 5.0 fps	0.60s - 1.7 fps
CThead bone	0.18s - 5.5 fps	0.49s - 2.0 fps
MRbrain	0.14s - 7.1 fps	0.38s - 2.6 fps
Engine	0.17s - 5.9 fps	0.39s - 2.6 fps
Engine steel	0.08 s - 12.5 fps	0.16s - 6.1 fps

It can be seen that all tissues can be rendered at multiple frames per second even at 512×512 image resolution. At a lower 256×256 image resolution all tissues can be rendered at more than 5 frames per second, while tissues with a small surface, such as the steel parts of the engine block, can be rendered at well over 10 frames per second.

The only pre-computation that has to be done is the determination of the binary shell. This computation has to be done every time the iso-value changes. The time-critical part of this pre-computation is the thresholding of the grey-value data. Changing the iso-value in

the CThead dataset for instance takes less than 0.2 seconds. Changing the iso-value and visualization of the skin or bone can hence be done at 2.5 frames per second at 256×256 image resolution. This makes it possible to interactively adapt the iso-value. Voxel based surface rendering algorithms have to calculate a new triangulated iso-surface when the iso-value is changed. This requires much more computation time, especially when the number of generated triangles is that high that an algorithm has to be used to reduce the number of triangles to be able to visualize the surface fast enough.

In the following chapter about medical applications, also higher resolution datasets are used. It will be shown there that the speed of iso-surface volume rendering remains high when the resolution of the dataset grows significantly.

8.3 Multiple iso-surfaces

In the previous chapters, only the visualization of a single iso-surface, that is defined by a single iso-value, is described. This iso-surface splits the volume in two parts: one in which all (reconstructed) grey-values are lower than the iso-value, and one in which all grey-values are higher than the iso-value. It is also possible to define multiple iso-values. When for instance two iso-values are given, the volume is split in three parts: one in which all grey-values are lower than the lowest iso-value, one in which all grey-values are higher than the highest iso-value, and one in which all grey-values lie between the lowest iso-value and the highest iso-value. In the extreme case that both iso-values are equal, only the iso-surface itself is visualized.

To be able to efficiently visualize such a multiple iso-surface, the calculation of the binary shell has to be adapted. Only the cells that are intersected by one (or more) of the iso-surfaces have to be processed. This means that the sub-volumes in which all grey-values are lower than the lowest iso-value and the sub-volumes in which all grey-values are higher than the highest iso-value can be skipped. Furthermore, all sub-volumes in which all grey-values lie between two consecutive iso-values can be skipped. The remaining set of sub-volumes will be intersected by one or more iso-surface and should hence be processed.

Figure 8.5 shows an example of two iso-surfaces within a single sub-volume. This figure is only used to demonstrate how accurate and free of sampling artefacts two iso-surfaces can be rendered simultaneously. In this figure, the part of the volume with grey-values between the two iso-values is rendered opaque while the remaining part of the volume is transparent. This figure also shows what kind of complex surfaces can be rendered using the iso-surface volume rendering method. The visualized cube contains voxels with the values (and locations): 3 (0,0,0), 6 (0,0,1), 4 (0,1,0), 10 (0,1,1), 5 (1,0,0), 7 (1,0,1), 7 (1,1,0), and 2 (1,1,1). With an iso-value between 3.0 and about 5.35 the volume will contain two separate surfaces. With values of 5.4 to 5.55 a double iso-surface similar to the surface shown in Figure 8.5 is visualized. An iso-value above 5.7, again yields two separate surfaces. The cases at which two separate surfaces are visualized can also be approximated using the marching cubes algorithm. The main difference is that the estimated surfaces are flat. Using the marching cubes algorithm however, also the closed surfaces shown in Figure 8.5 would be rendered as two opaque planes.

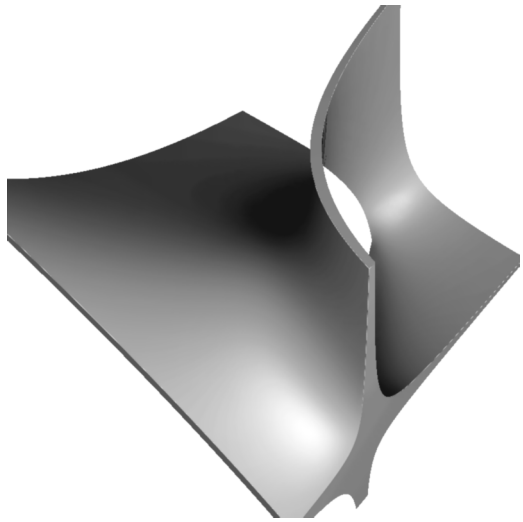


Figure 8.5 Multiple iso-surfaces within a single voxel cube

The described method for rendering multiple iso-surfaces is however only useful when it is possible to see both the outside as well as the inside of an object. This is possible by segmentation of

the volume or by cutting the object in pieces. In Figure 8.5, the volume is segmented so that only one voxel cube is visible.

8.4 Shading

By using the iso-surface volume rendering method to visualize iso-surfaces in volumetric data, it has become possible to perform manipulations of the rendered image with interactive to real-time speed. Some of these manipulations are: combining several tissues, changing the position of the light source, changing tissue colors, depicting cut-planes etc.

To increase the speed of the shading, several table look-up based approaches have been proposed [TER95]. Given the light position, it is possible to pre-compute the intensity for all possible surface normal vector orientations, and store this intensity in a lookup table. When the shading table calculation and the lookup table based shading can be done fast enough, the position of the light source can be changed in real-time within a rendered scene.

The complete rendering pipeline based on this approach is depicted in Figure 8.6.

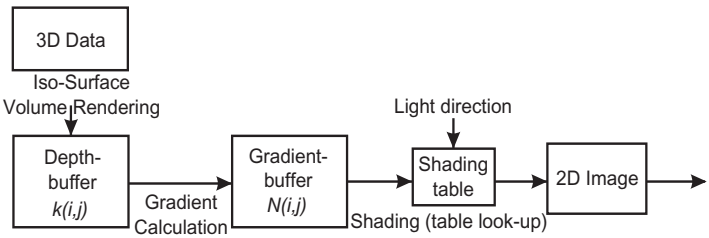


Figure 8.6 Rendering pipeline

From the 3D dataset, a 2D floating-point depth-buffer can be created using the iso-surface volume rendering algorithm. Using this depth-buffer, a gradient buffer that contains the surface normal or the volume gradient for each pixel can be calculated. This surface normal is used as an index to find the correct shading in a pre-computed 2D shading-table. This shading will result in a 2D grey-value image. Besides calculating the gradient buffer from the depth buffer, it can also be generated in the iso-surface volume rendering algorithm directly.

The calculation of the shading-table can be simplified by considering the fixed direction of the observer. When the gradient vectors are calculated in viewing coordinates, the observer vector always equals [0,0,-1]. Consequently, the inner product of the observer vector $[O_i,O_j,O_k]$ and the reflected light vector $[R_i, R_j, R_k]$ is always equal to $-R_k$. Hence, only the k component of the reflected light vector has to be calculated. Furthermore, the shading table can be kept small because the k component of the gradient should always be negative.

Because of this approach, a shading table with 128×128 entries can be computed in less than 0.01 seconds on a SPARCstation 5. It is clear that the goal of real-time shading is not prohibited by the calculation of the shading table. By using a gradient buffer, the shading of medium sized images (256^2 - 512^2) can be done interactively (>10 frames per second). On UltraSparc workstations and modern Pentium III pc's the shading can be performed in real-time (>25 fps), even for large images. In this case, also the size of the shading table can be enlarged, for instance to 256×256.

8.5 Coloring

The shading approach described in the previous section results in a 2D image that consists of grey-values. To give a better interpretation, especially when multiple tissues will be rendered in the same image, the shading has to be extended to support colors. This extension can be done in several ways. An obvious approach is to extent the shading model to support full color. This approach will however significantly increase the calculation time. A simple and fast approach is to use a color model to translate the grey-values into (RGB) colors. In this chapter, two color models will be discussed: the HSV and the HLS model [FOL90], [WAT93].

8.5.1 The HSV color model

A frequently used approach to convert grey-values to colors is based on the hue, saturation and value (HSV) color model. This model can be used in our algorithm by choosing fixed hue and saturation values for a tissue. When the value is chosen to be equal to the grey-value, each grey-value will correspond to one color. When the saturation is zero, the output image will be equal to the

grey-value image. The colors will be fully saturated when the saturation is chosen to be one. Choosing a hue value of zero (red) and a saturation of one will lead to an image which has zero G and B values, while the R value is equal to the grey-value. Because the specular highlights, which are white in the grey-value image, will become red, the HSV color model is unsuitable for coloring. Because in nature the specular highlights of a white light source are white, the highlights will seem to disappear. The color white can be made in the HSV model by making the saturation zero when the value equals one. A possible solution to this problem is to let the saturation depend on the grey-value. Another solution is to use another light model.

8.5.2 The HLS color model

The hue, lightness and saturation model (HLS) is a slightly moderated version of the HSV model. In this model, the color white can be made by setting the lightness parameter to 1.0, regardless of the saturation and hue. Completely saturated colors are made by setting the saturation to 1.0 and the lightness to 0.5. Therefore, the HLS model can be used to directly map a grey-value to a certain color. The hue and saturation can again be chosen freely, while the lightness is set equal to the grey-value. In this way, the grey-value white will always be mapped to the color white. This approach gives a more natural color mapping than when the HSV model was used in a similar way.

However, this approach results in piece-wise linear R, G and B values, which is clearly visible, especially when an object with a smooth surface, like a sphere, is rendered. Figure 8.7 shows the resulting R, G and B values when for instance the hue is set to zero and the saturation to one.

As can be seen in this figure, the red component is fully saturated at a grey-value of 0.5, while the green and blue components are zero. As mentioned, this gives unacceptable results.

This problem can be solved by making the saturation smaller at a grey-level of 0.5. When the saturation is chosen to be 0.5 at the grey-level 0.5 and one at the grey-levels zero and one, the color mapping will look most natural. At this setting, the functions that map a grey-value to the three color components will have a

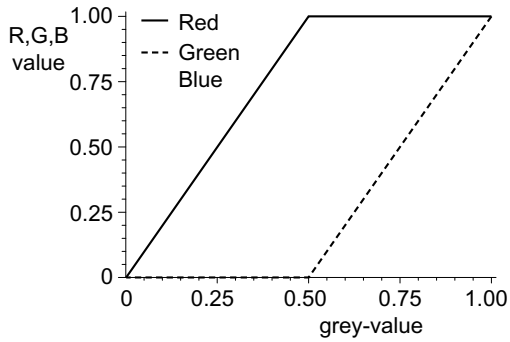


Figure 8.7 Direct grey-level to color mapping using the HLS model

quadratic form. In this case, both these functions and their derivatives are continuous. Consequently, the color images will have smooth continuous color transitions. The saturation value can still be chosen freely as a multiplication factor. For grey-values lower than 0.5, the resulting saturation will be (1-g)*s, while for higher values the resulting saturation will be (g)*s. When for instance the saturation is chosen to be 0.5, the real saturation is 0.5 at the grey-levels zero and one and 0.25 at the grey-level 0.5.

The result of this adaptation of the saturation can be seen in Figure 8.8. In this figure, two spheres are rendered, one with a constant saturation of one, and one with an adapted saturation. It can be seen clearly that the constant saturation value leads to an undesirable saturated red color band in the output image. Adjusting the saturation value based on the grey-value using the above mentioned method leads in the contrary to a smooth transition from black to white. Although this approach limits the maximum obtainable saturation, the mapping of grey-values to full color looks far more natural.

The transformation of the grey-valued images to color images can be done very quickly using a lookup table. This table contains the R, G and B values corresponding to a certain grey-value given the hue and saturation. This means that the grey-value to color calculation has to be done once for each grey-value, resulting in a 256 entry lookup table of three bytes per entry.

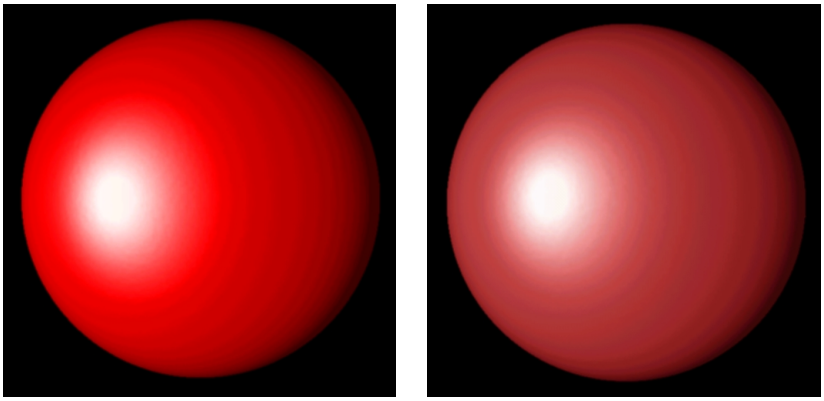


Figure 8.8 Result of grey-value to color mapping using the HLS model without (left) and with adapting saturation

8.6 Combining multiple tissues

Based on the floating-point depth buffer that can be generated with the iso-surface volume rendering method, it is possible to combine the renderings of different tissues very accurately. For each ray, the location of the intersection of the ray with the iso-surface of each tissue is known. When the iso-surfaces are rendered fully opaque, it is hence easy to select the iso-surface and tissue that is first intersected by the ray. A fast and easy way to implement this combination of tissues is by pre-computing a new shading buffer that contains for each pixel an index in the shading table and a tissue number. This can be easily done by storing the depth of the first intersection point in a new depth buffer and use the shading index and tissue number of the corresponding iso-surface in a new combined shading buffer. Adding another (already rendered) tissue only requires to compare the value of all pixels in the corresponding depth buffer with the pixel values of the combined depth buffer. The corresponding pixel value in the combined shading buffer has to be replaced only when the depth of the new tissue is smaller. Because this is a two-dimensional operation, combining several tissues in this way can be done very fast, even for high resolution images.

When each tissue, and hence each pixel, is allowed to have its own color and shading parameters, the approach of the shading table

and coloring has to be changed a little. It is not possible to have one shading/coloring table for all tissues. It is however not practical to calculate a separate shading/coloring table for each tissue. Especially when the number of tissues is large, this would require a lot of memory and calculation time. A possible solution is to store only parts of the shading in the two-dimensional shading table, while the final shading and coloring can be done by a simple calculation and a one-dimensional table lookup, as shown in Figure 8.9.

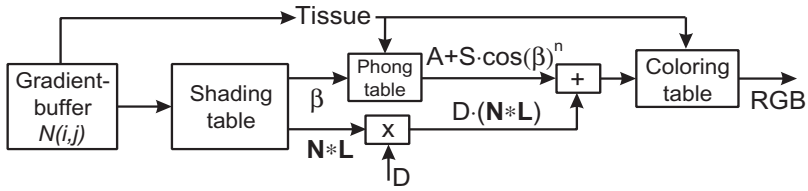


Figure 8.9 Full-color shading pipeline for multiple tissues

In the shading table, only the parts of the Phong shading model are stored that are independent of the shading and coloring parameters. In the diffuse part of the shading model, the calculation of the inner product of the light and the normal vector **N*L** is independent of the tissue parameters. This inner product will hence be stored in the shading table. In the specular part of the shading model, the inner product of the reflected light vector and the observer vector **R*O** is independent of the tissue parameters. This inner product will be used as an index in a second table to calculate the n-th power of this inner product. This approach will however give unsatisfactory results when the parameter n has a high value. For instance, when n has a value of 100, and eight bits are used for the index, all points where the inner product has a value of 255 will give an output value of 255, which corresponds to one. The neighboring points, where the inner product has a value of 254, will have an output value of $255*(254/255)^n = 172$. This huge step in the output value will become clearly visible in the output image, where a highlight with a high exponent will consequently be visualized as areas with only a few possible grey values, as shown in Figure 8.10.

Much better results can be obtained when not the inner product, but the angle β between the reflected light vector and the observer vector is used as an index in the second table. When again an eight

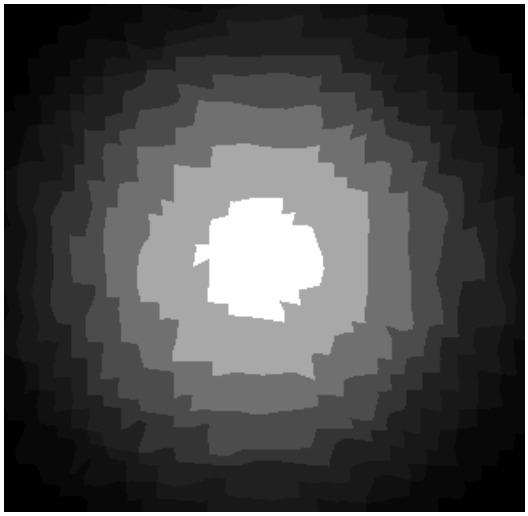


Figure 8.10 Discretization of the specular shading component

bit index is used, the value of $\cos(\beta)^n$ can be determined for angles that are a multiple of 90/255 = 0.35 degrees. This approach will hence give a much more continuous highlight than is obtained with the previous method, as shown in Figure 8.11.

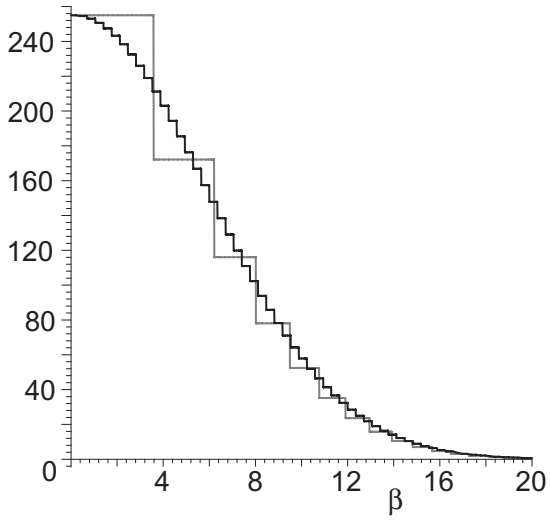


Figure 8.11 Improvement of the angular resolution of the specular shading component

In this way, only one shading table has to be calculated that is tissue independent. Using a second, one dimensional table, the ambient and specular component can be calculated by a single table look-up. This table has to be calculated for each tissue. This table is very small and only has to be re-calculated when one of the parameters A, S or n changes. The calculation of the diffuse part only requires a multiplication with the diffuse parameter D. Finally, the ambient, specular and diffuse components have to be added, which requires only one addition. When the tissues are rendered in color, a color look-up table can be calculated for each tissue.

When eight bit numbers are used for all indices in the tables, a 256×256 shading table containing 16 bit numbers, a 256 entry Phong table containing eight bit numbers and a 256 entry coloring table containing 24 or 32 bit numbers have to be used. This results in a 128 KB shading table, a 256 byte Phong table and a 768 byte to 1024 byte coloring table. Although the last two tables have to be calculated for each tissue separately, the total size of these tables per tissue is more than 100 times smaller than the shading table. As a result of this approach, many tissues can be rendered simultaneously with all independent shading and coloring parameters, while the memory requirements are hardly influenced.

Figure 8.12 shows a rendering of different materials inside an engine block. The aluminum outer surface of the engine block is depicted as the blue iso-surface. Using the iso-surface volume rendering method, it is possible to render iso-surfaces with an arbitrary small thickness. It is therefore possible to render the inside and the outside of a very thin shell simultaneously. In Figure 8.12, the inner side of the aluminum surface is rendered with a green color. The aluminum itself is invisible. The steel parts inside the engine block are rendered using a beige color. The aluminum to air surface is only partly rendered, so it is possible to look inside the engine block.

Figure 8.13 shows two material transitions inside the engine block with more detail. The left image shows the aluminum to air transition, while the right image shows the aluminum to steel transition.

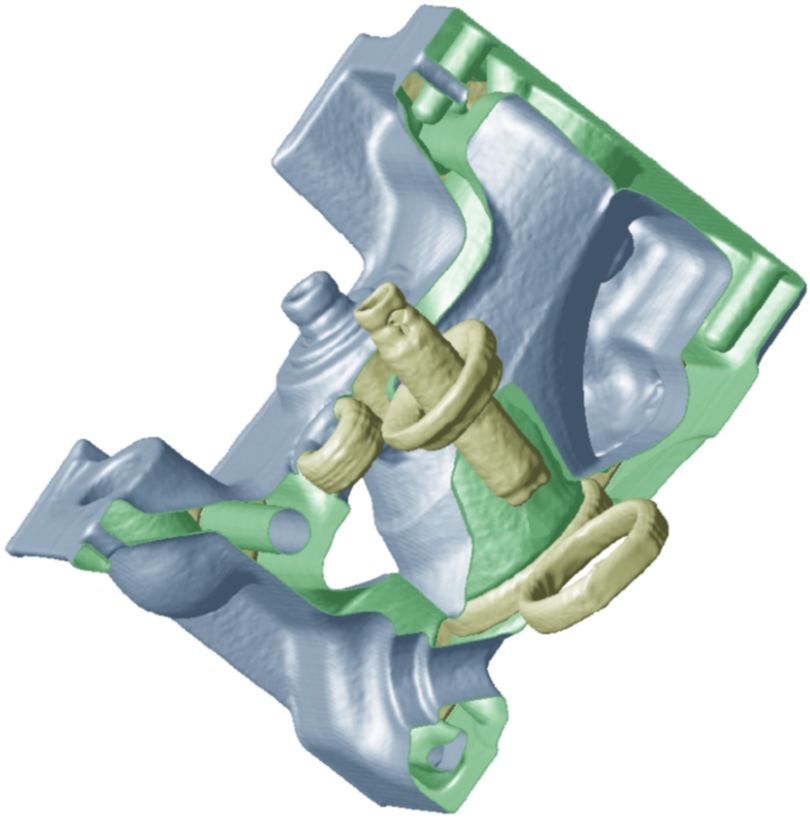


Figure 8.12 Rendering of different materials inside an engine block

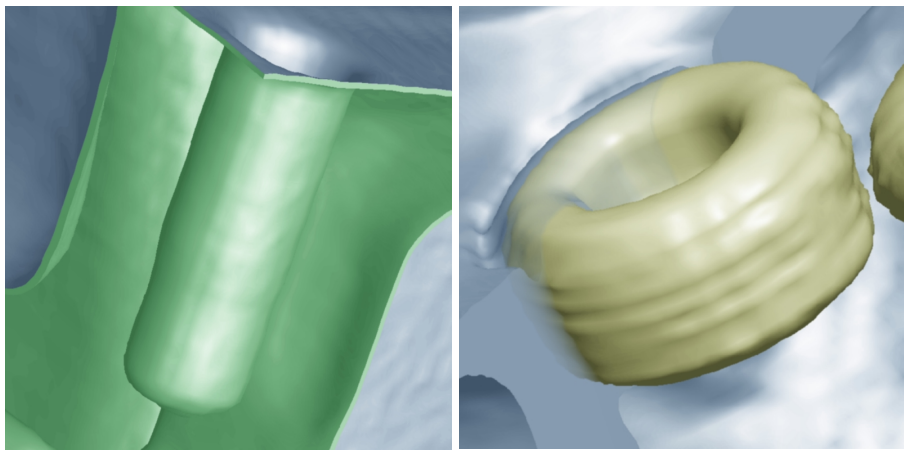


Figure 8.13 Details of material transitions inside the engine block

In the left image, the aluminum to air surface has been given a small thickness. The outer surface is rendered blue, while the inner surface and the volume between the two surfaces are rendered green. Because it is possible to look at the inner side of the aluminum to air transition, it is possible to depict a screw hole in this way.

In the right image, the aluminum is not only rendered partly, but also semi transparent. This makes it possible to see the steel screw through the 'aluminum' surface. It should be noticed that this image shows in fact also the transition from steel to air. Because the aluminum has a lower grey-value than the steel parts, the iso value of an iso surface between aluminum and air also lies between steel and air. As a result, a steel to air transition will be depicted as both a steel to air and a aluminum to air transition, with the 'aluminum' to air surface positioned at the air side. This problem can be partly solved using segmentation. In this way, steel parts inside aluminum or fully surrounded by air can be segmented out and rendered separately. The example in the right image of Figure 8.13 is however more difficult, because this area contains the aluminum to air, the aluminum to steel and the steel to air transitions. To remove the 'aluminum' layer on the steel screw, not only the screw needs to be segmented out, but also the transition area from steel to air. By making the aluminum slightly transparent, it is possible to see through the 'alimunum' layer between air and steel. In Figure 8.12, where the aluminum was rendered fully opaque, this was not possible.

The renderings in Figure 8.13 also show how much detail can be obtained using the iso-surface volume rendering method. In the left image, the depth of the thread tapped inside the screw hole can be observed. While the thread itself is much smaller than the point-spread function and therefore too small to be observed, the hole looks wider at the part where the thread is tapped. In the right image the thread of the screw is much larger and is clearly visible.

8.7 Cut planes

Figure 8.12 and Figure 8.13 show renderings in which only a part of the aluminum was rendered. Although it is possible to select the part to render before the actual rendering, it may be advantageous

to render the hole volume and select the visible part in a post-processing step. In this way, the volume does not have to be rendered again when the selection changes.

By using the floating-point depth buffer it is easy to select points that lie inside the selected area. The cut plane between the selected an not selected parts of a tissue can be shaded as shown in Figure 8.13. In this way, parts that are hidden by an iso-surface can be made visible. Because this is again a two-dimensional operation, it can be done very fast even for high resolution images.

Besides revealing hidden parts, cut planes can also be used to give information about the inside of a tissue. This can be achieved by mapping for instance grey-value information on the cut plane. The iso-surface shows a surface with a constant grey value. The grey value inside the tissue is not necessarily constant and may contain much information. In an MRI dataset of a head for instance, the iso-surface of the cortex gives information about the shape of the brain. Inside the brain, the grey-value is however not constant. To be able to see this information in a 3D rendering, the grey-values may be mapped onto a cut plane directly. To be able to distinguish the grey-values from different tissues, it is also possible to translate the grey-values to colors using the same approach as described in section 8.5. Using the same hue for a tissue and a cut plane may however be confusing because it is difficult to see whether a certain color is caused by shading the surface or by mapping a grey value. It is therefore more sensible to choose a hue for each cut plane separately. In this way it is also possible to distinguish between two or more cut planes inside a single tissue.

CHAPTER

9

APPLICATION TO MEDICAL DATA

As mentioned previously, the medical application of volume visualization algorithms requires both high quality as well as high speed. It has been shown that rendering iso-surfaces using the iso-surface volume rendering method can be fast without sacrificing the image quality. This chapter will show some of the possible applications in which the iso-surface volume rendering method can be used.

9.1 Parallel projection

Figure 9.1 shows one slice of the raw data of a CTA dataset in which the aorta and a stent at the wall of the aorta are clearly visible. This CTA dataset contains 512×512×123 12-bits voxels, stored in 16-bits per voxel, which corresponds to a total amount of 61.5 MB of data. This is currently a very common size for a CT dataset.

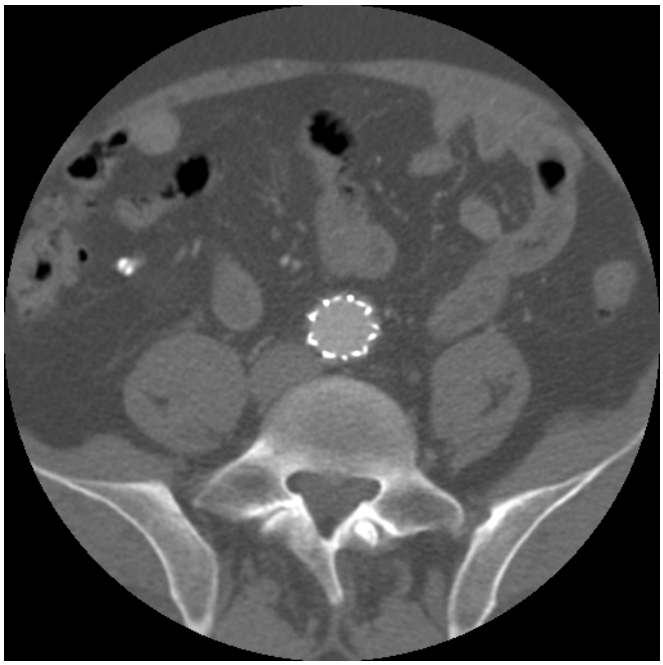


Figure 9.1 Slice of a 512×512×123 voxel CTA dataset

Figure 9.2 shows a detail of this slice. In the left image the resolution of the dataset is clearly visible. The radius of the aorta is about 25 voxels. The smallest parts of the stent have a radius of less than two voxels. The right image shows the effect of the iso-surface visualization on this data. Because the stent has a higher grey-value than the aorta (actually the contrast agent in the blood), the iso-value to visualize the stent has to be between the grey-values of these two tissues. To visualize the blood in the aorta, an iso-value should be used that is lower than the grey-value of the blood. Using these two iso-values, two iso-surfaces are generated: one for the aorta and one for the stent. In the right image of Figure

9.2, the iso-surface of the stent is visible as the edges between the grey and the white areas, while the iso-surface of the aorta is visible as the edge between the grey and the black area.

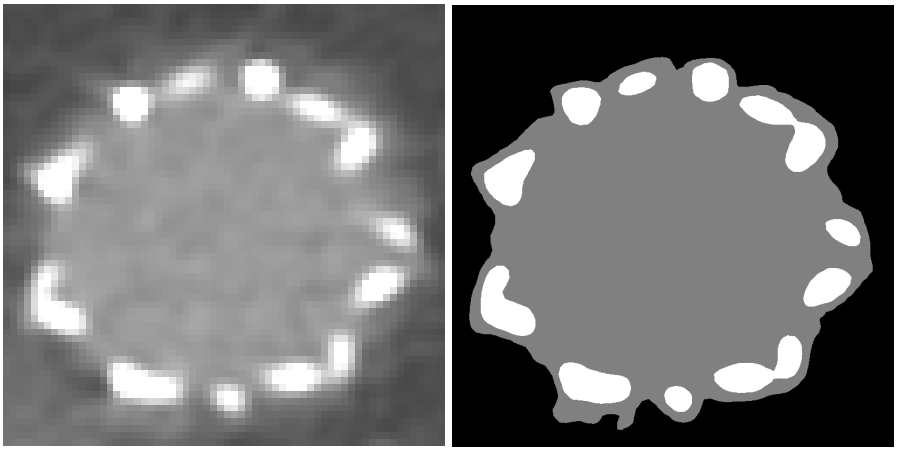


Figure 9.2 Detail of the slice in Figure 9.1 showing the aorta with the stent inside (left), and the surface that is estimated using the iso-surface volume rendering algorithm

As can be seen in this image, the iso-surface of the stent lies within the iso-surface of the aorta, which means that the stent will be invisible when both tissues are observed from outside. By using alpha blending based on the two generated depth-buffers, the aorta can be made slightly transparent to be able to see the stent just below the surface. Using this approach the stent was made visible in Figure 9.3.

The good quality of iso-surface rendered volumetric data can be seen from the rendering of a raw MRA dataset of a cortex with very thin blood vessels.

This MRA dataset consists of two 256×256×151 12-bits per voxel datasets, stored in 16-bits per voxel: one for the cortex and one for the blood vessels, resulting in a total amount of 37.75 MB of data. One slice out of each of these datasets was shown in Figure 3.2. The tissues were rendered separately and combined after rendering based on the two floating-point depth buffers generated. The cut-plane shown in Figure 9.4 shows the original grey-values in one of the slices. This cut-plane was also generated after rendering based on information from the floating-point depth buffer using the approach described in the previous chapter. It can

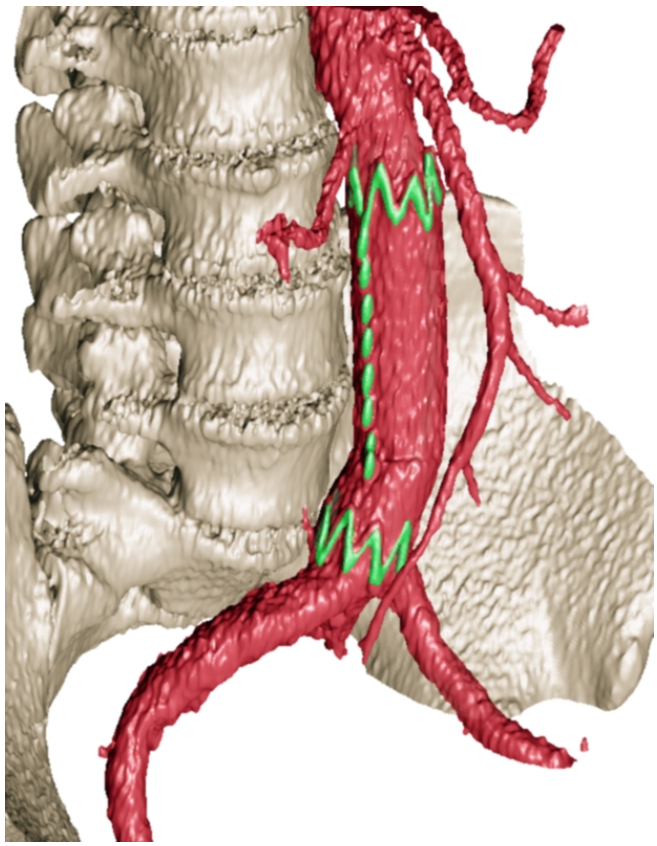


Figure 9.3 Iso-surface volume rendering of the spine, the aorta and the stent in the CTA dataset

be seen in Figure 9.4 that the iso-surface volume rendering method is capable of visualizing very small details in a medium resolution dataset.

Figure 9.4 also shows the advantages of full color images.

9.2 Virtual endoscopy

Figure 9.3 and Figure 9.4 show renderings of a CT and an MRI dataset using parallel projection. These images can also be generated with a perspective component. In these examples, perspective projection would however hardly add any information.

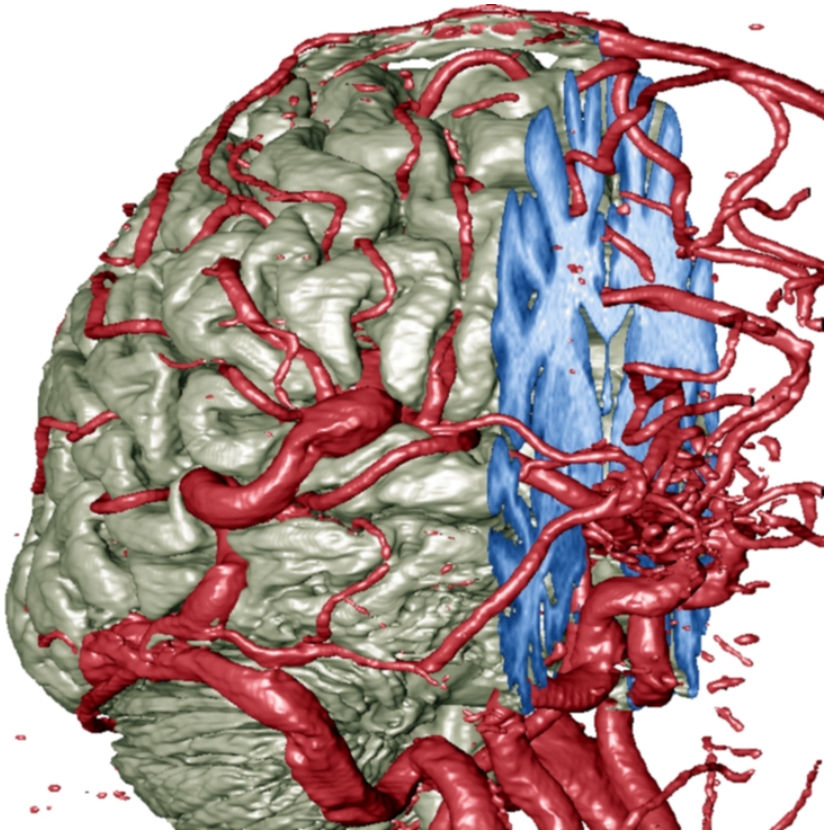


Figure 9.4 Rendering of a raw 256×256×151 voxel MRA dataset of a brain with blood vessels

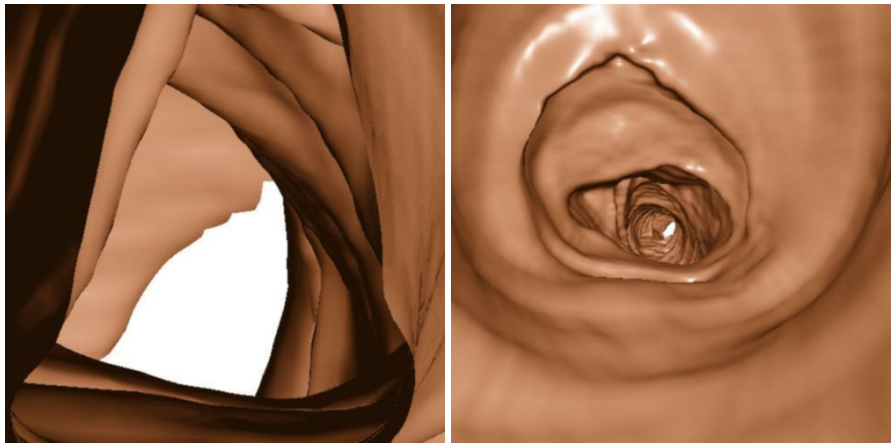


Figure 9.5 Parallel (left) and virtual endoscopic view inside the trachea

There are however cases in which the use of perspective projection is inevitable. The parallel projection of an iso-surface of the trachea in the left image of Figure 9.5 shows that it is difficult to observe the inside without perspective projection. Furthermore, a parallel light source, that was used to shade the iso-surfaces in previous examples, can only light one side.

In medical examinations, a special camera with attached light source called an endoscope is often used to observe the inner side of the human body. This is a wide angle camera with extreme perspective. The light source can be considered to be a point light-source that is positioned near the lens.

In (iso-surface) volume rendering, similar projections can be made. By using extreme perspective projection with the viewpoint inside the object under study and a light source positioned at the viewpoint, so called virtual endoscopic images can be created. The right image in Figure 9.5 shows such an virtual endoscopic view inside the same part of the trachea as was shown in the parallel projection in the left image of Figure 9.5. It can be seen in this image that the extreme perspective projection makes it possible to observe the wall of trachea in all directions simultaneously. Furthermore, the shading based on a point light-source at the viewpoint results in an equally shaded surface for all viewing directions.

As mentioned in section 7.2, voxels near the viewpoint are enlarged extremely when perspective projection is used. Conventional volume rendering methods would produce a severely distorted image due to re-sampling artefacts. Even when the spatial error would be about half a voxel size, these artefacts would become visible. Super resolution techniques can be used to improve the image quality at the expense of a considerable increase in the time needed to generate images with ‘acceptable’ re-sampling errors. Due to the angular dependency of the sampling process in the conventional volume rendering methods, a very small step size is needed when diverging rays are used, especially in a virtual endoscopic view. The iso-surface volume rendering approach is however able to make endoscopic views without re-sampling artefacts and without the need to decrease the sampling distance. This makes the iso-surface volume rendering algorithm very suitable for virtual endoscopic applications.

Virtual endoscopy can be used to generate images that are comparable to ordinary endoscopic images. The main advantage is that virtual endoscopy is less invasive. Another disadvantage of the traditional endoscopy is the restriction of the viewing direction. With an endoscope it is only possible to look forward in the direction in which the endoscope is inserted. Using virtual endoscopy it is possible to look in all directions. This is illustrated in the rendering of the inner wall of a colon shown in Figure 9.6.

Endoscopic examination of the colon is commonly referred to as colonoscopy, so the virtual rendering of the colon can be referred to as virtual colonoscopy. In Figure 9.6, the viewpoint is positioned at a bend in the colon, looking both forward and backward simultaneously. It is clear that such a view is impossible with traditional colonoscopy. Furthermore, it is obvious from this view that manoeuvring with an endoscope through this part of the colon without damaging the wall can be very difficult.

A further complication with traditional endoscopy is the opacity of the material. Figure 9.7 shows a virtual endoscopic view inside the aorta of the CTA dataset previously shown in Figure 9.3. With normal endoscopy, the opacity of the blood inside the aorta makes it difficult or even impossible to see the wall and the stent. The clear view of both the stent as well as the wall of the aorta is impossible with traditional endoscopy.

As mentioned before, the radius of some parts of the stent is less than two voxels. As shown with the renderings of the single voxel dataset, even for these small objects the iso-surface volume rendering method is capable of rendering a well defined iso-surface without visible sampling artefacts.

Some medical applications of volume rendering extract shape information from the iso-surfaces in the data field. For instance, the typical round nature of a polyp can be color coded on the iso-surface found. This has been done by [SUM98] using a polygonal approximation extracted from a data field using the marching cubes algorithm. It may be clear that the extraction of shape parameters is quite simple when the depth information is present for each ray within a bundle, reaching a smoothly curved surface. The accurate calculation of the local curvature is, due to the discontinuity in the direction of the surface normal, by far less trivial when a polygonal approximation is given.

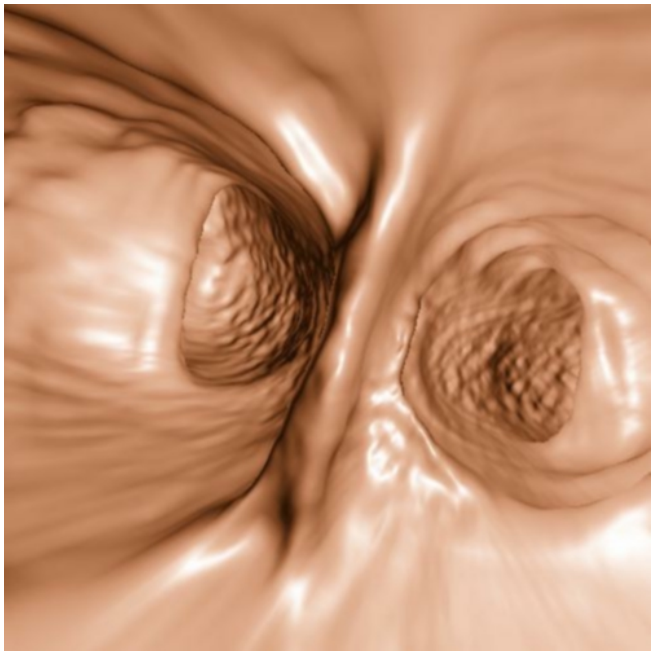


Figure 9.6 Virtual colonoscopy

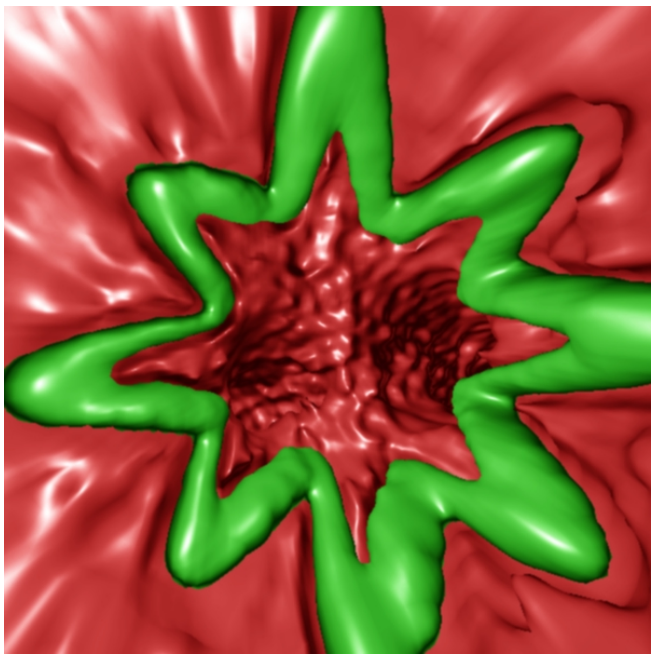


Figure 9.7 Endoscopic view inside the aorta

The rendering time of the iso-surface volume rendering method mainly depends on the projected size of the tissue on the screen. Complex tissues with a large surface, such as the cortex and the spine in Figure 9.4 and Figure 9.3, can be rendered well below one second on a Pentium III pc running at 500 MHz using high resolution output images of 512×512 pixels. The spine for instance can be rendered in 0.77 s.

Simpler tissues, such as the aorta, can be rendered in less than 0.3 seconds at the same high resolution, while the stent can be rendered with more than 10 frames per second. At a low resolution of 256×256 pixels, interactive rendering speed (multiple frames per second) can be obtained for all tissues. At this resolution, the spine can be rendered at more than 3 frames per second, the aorta can be rendered at over 10 frames per second, while the stent can be rendered at about 25 frames per second. It is clear that the size of the dataset is not the main factor in the rendering time. The size of the CTA dataset is about 4.5 times the resolution of the test datasets that were used for benchmarking in the previous chapter.

CHAPTER

10

CONCLUSIONS

In this thesis, a method to evaluate the spatial error bounds for the rendering of an iso-surface in a volumetric dataset is presented. This spatial error bound is estimated for four different categories of visualization algorithms. Using the derived equations for the spatial error bound, it is possible to find a way to reduce the spatial error for each visualization method.

In the voxel space volume rendering algorithms, the spatial error bound grows linearly with the voxel size. A valid way to reduce the spatial error is hence the reduction of the voxel size, for instance by oversampling the dataset. Unfortunately, the computation time of these algorithms grows linearly with the number of voxels, which means that this approach will have a much larger impact on the rendering time than on the spatial error.

The marching cubes algorithm has a spatial error bound that grows linearly with the voxel size as well. In this case, the spatial error can also be reduced by reducing the voxel size. However, this will dramatically increase the time needed to extract the iso-surface as well as the number of polygons generated. Alternative surface extraction methods are able to generate a triangle mesh

with a smaller spatial error bound. These methods will however suffer from the same performance problems.

In the super resolution algorithm, the error bound grows linearly with the re-sampling distance on a ray. Because the rendering time grows linearly with the number of samples on a ray, the rendering time will increase linearly with the spatial accuracy. When a small detail in a dataset is enlarged, well over ten samples per unit cell are necessary to avoid re-sampling artefacts.

In the iso-surface volume rendering method, the spatial error bound can be decreased by increasing the number of iteration steps in the surface finding algorithm. Each additional re-sampling operation per ray decreases the spatial error bound by a factor of two. In this way, the spatial error can be made very small without a noticeable degradation of the rendering speed. Therefore, the iso-surface volume rendering technique makes it possible to visualize the iso-surface of a discrete data field, re-sampled with an interpolation filter of choice, without the introduction of a noticeable spatial error.

The search for an efficient method with a negligible spatial error bound has made it possible to make endoscopic views of arbitrary shaped iso-surfaces from raw (medical) data without re-sampling artefacts. The rendering speed is just marginally influenced by the size of the dataset. Hence, interactive rendering speeds can be obtained for very high resolution datasets.

When compared to other methods, it can hence be concluded that the iso-surface volume rendering method has an excellent performance both in terms of the spatial error bound as well as rendering speed, even when virtual endoscopic applications are considered. For these reasons, iso-surface volume rendering is a very good alternative to surface and volume rendering algorithms for the visualization of iso-surfaces in volumetric data.

It has been shown that the iso-surface volume rendering method performs also very well for high resolution datasets. Due to improvements in the scanners, the size of medical datasets will even increase. While this has only a minor impact on the rendering speed in the iso-surface volume rendering method, the rendering speed of many other visualization algorithms will be decreased drastically.

To further increase the speed of iso-surface volume rendering, current research is targeted toward implementation on multi-processor engines. Our goal is to achieve real-time (~25 fps), high resolution rendering with low-cost general purpose processors.

In section 7.1, the accuracy of the determination of a step edge was investigated. It was shown that reconstruction filters that perform worse in typical 2D image re-sampling applications may be better suitable for the detection of the location of the step edge. This conclusion is only applicable to a 1D step edge. In 3D data it is possible to have large surfaces that act like a 1D step edge. Further research is however necessary for other 3D objects.

Using the knowledge of the systematic error, it is possible to correct the estimated location. In 1D this would be possible when the values on both sides of the step edge and the point-spread function are known. Experiments have shown that the systematic error can be reduced to almost zero. This correction is however extremely noise sensitive. When the point-spread function in 3D is point symmetrical, and the values on both sides of a surface are known, a similar method can be used that translates the grey-values into a distance to the surface. It has been shown however that in current medical scanners, the point-spread function is usually not point symmetrical. When future scanners have a fully point symmetrical point-spread function, it may be worthwhile to further explore the possibilities.

CHAPTER

11

BIBLIOGRAPHY

[BEN95] M.J. Bentum (1995) *Interactive Visualization of Volume Data*. Ph.D. thesis, University of Twente, ISBN 90-9008788-5.

[BEN96] M.J. Bentum, B.B.A. Lichtenbelt, and T. Malzbender, (1996) Frequency Analysis of Gradient Estimators in Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 2(3), pp. 242-254.

[BLO88] J. Bloomenthal (1988) Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4), pp. 341-355.

[BOS95a] M.K. Bosma and J. Terwisscha van Scheltinga, (1995) *Efficient Super Resolution Volume Rendering*. Masters thesis, University of Twente, http://utelnt.el.utwente.nl.

[BOS95b] M.K. Bosma, J. Smit, and J. Terwisscha van Scheltinga, (1995) Super Resolution Volume Rendering Hardware. *Proc. of the 10th Eurographics Workshop on Graphics Hardware*, volume EG95HW, pp. 117-122.

[BOS98] M.K. Bosma, J. Smit, and S. Lobregt, (1998) Iso-surface volume rendering. *Proc. of SPIE Medical Imaging '98*, 3335, pp. 10-19.

[CHU95] J.H. Chuang and W.C. Lee, (1995) Efficient generation of iso-surfaces in volume rendering. *Computers & Graphics*. vol.19, no.6; Nov.-Dec. 1995; pp. 805-813.

[COH98] J. Cohen, M. Olano, and D. Manocha (1998) Appearance-Preserving Simplification. *Siggraph Conference Proceedings*, pp. 115-122.

[CRA93] R.A. Crawfis and N. Max, (1993) Texture splats for 3D scalar and vector field visualization. *Proceedings Visualization '93* (Cat. No. 93CH3354-8). IEEE Comput. Soc. Press, Los Alamitos, CA, USA; 1993; xv+423.

[CRA96] R.A. Crawfis (1996) Real-time slicing of data space. *Proc. Visualization '96* (IEEE Cat. No.96CB36006). ACM, New York, NY, USA; 1996; 516 pp. 271-277.

[DOD97] N.A. Dodgson (1997) Quadratic Interpolation for Image Resampling. *IEEE Transactions on Image Processing* 6(9), pp. 1322-1326.

[FOL90] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes (1990) *Computer Graphics: Principles and Practice*, 2nd. ed., Addison Wesley, ISBN 0-201-12110-7.

[FRI89] B. Friedman, J.P. Jones, G. Chavez-Munoz, A.P. Salmon, and C.R.B Merritt (1989) *Principles of MRI*. McGraw Hill.

[FUJ96] I. Fujishiro, Y. Maeda, H. Sato, and Y. Takeshima (1996) Volumetric data exploration using interval volume. *IEEE Trans on Visualization and Computer Graphics*, vol. 2, no. 2; June 1996; pp. 144-155

[GOU71] H. Gouraud (1971) Continuous Shading of Curved Surfaces. *IEEE Transactions on Computers*, 20(6), pp. 623-629.

[GRE97] G.J. Grevera and J.K. Udupa (1997) An Objective Comparison of Interpolation Methods. *Proc. of SPIE Medical Imaging '97*, 3031, pp. 2-10.

[GRO95] M.H. Gross, L. Lippert, A. Dreger, and R. Koch (1995) A new method to approximate the volume rendering equation using wavelet bases and piecewise polynomials. *Computers & Graphics*. vol.19, no.1; Jan.-Feb. 1995; pp. 47-62.

[GUO95] B. Guo (1995) A multiscale model for structure-based volume rendering. *IEEE Trans on Visualization and Computer Graphics*. vol.1, no.4; Dec. 1995; pp. 291-301.

[HEL99] S. Helgason (1999) *The radon transform*. (Progress in mathematics 5), 2nd. ed., Boston: Birkhäuser, ISBN 3-7643-3006-6.

[KAK88] A.C. Kak and M. Slaney (1988) *Principles of Computerized Tomographic Imaging*. New York: IEEE Press, ISBN 0-87942-198-3.

[KLE96] R. Klein, G. Liebich, and W. Strasser (1996) Mesh reduction with error control. *Proc. Visualization '96* (IEEE Cat. No.96CB36006). ACM, New York, NY, USA; 1996; 516 pp. 311-318.

[LAC94] P. Lacroute and M.S. Levoy (1994) Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation. *Computer Graphics*, 28(4) , pp. 451-458.

[LEV88] M.S. Levoy (1988) Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3), pp. 28-37.

[LIP95] L. Lippert and M.H. Gross (1995) Fast wavelet based volume rendering by accumulation of transparent texture maps. *Computer Graphics Forum*. vol.14, no.3; 1995; pp. C/431-443.

[LOR87] W.E. Lorensen and H.E. Cline (1987) Marching cubes: A High Resolution 3D Surface Construction Algorithm. *Computer Graphics*, 21(4), pp. 163-169.

[LUR98] C. Lurig, L. Kobbelt, and T. Ertl (1998) Deformable surfaces for feature based indirect volume rendering. *Proc. Computer Graphics International* (Cat. No.98EX149). IEEE Comput. Soc., Los Alamitos, CA, USA; 1998; xxi+800 pp. 752-760.

[MAC96] R. Machiraju and R. Yagel (1996) Reconstruction Error Characterization and Control: A Sampling Theory Approach. *IEEE Transactions on Visualization and Computer Graphics*, 2(4), pp. 364-377.

[MAL93] T. Malzbender (1993) Fourier volume rendering, *ACM Transactions on Graphics*, vol. 12, no. 3, pp. 233-250.

[MIT88] D.P. Mitchell and A.N. Netravali (1988) Reconstruction Filters in Computer Graphics. *Computer Graphics*, 22(4), pp. 221-228.

[MUE96] K. Mueller and R. Yagel (1996) Fast Perspective Volume Rendering with Splatting by Utilizing a Ray-Driven Approach. *Proceedings of Visualization'96*, pp. 65-72.

[PHO75] B. T. Phong (1975) Illumination for computer generated pictures. *Communications of the ACM*, 18(6), pp. 311-317.

[OPP75] A.V. Oppenheim and R.W. Schafer (1975) *Digital Signal Processing*. Prentice Hall, ISBN 0-13-214635-5.

[RAM71] G.N. Ramachandran and A.V. Lakshminarayana (1971) Three-dimensional reconstruction from radiographs and electron micrographs: Application of convolutions instead of Fourier transforms. *Proc. Nat. Acad. Sci.*, vol. 68, no. 9, pp. 2236-2240.

[RAT92] S. Rathee and Z.J. Koles (1992) Image Restoration in Computed Tomography: Estimation of the Spatially Variant Point Spread Function. *IEEE Transactions on Medical Imaging*, December 1992, pp. 539-545.

[SAK95] G. Sakas and S. Walter (1995) Extracting surfaces from fuzzy 3D-ultrasound data. *Computer Graphics Proceedings, SIGGRAPH 95*. ACM, New York, NY, USA; 1995; 518 pp. 465-474.

[SCH46] I.J. Schoenberg (1946) Contributions to the Problem of Approximation of Equidistance Data by Analytic functions. *Quant. Appl. Math.,* 4:45-99, pp. 112-141.

[SHE96] R. Shekhar, E. Fayyad, R. Yagel, and J.F. Cornhill (1996) Octree-Based Decimation of Marching Cubes Surfaces. *Proceedings of Visualization'96*, pp. 335-342.

[SHE74] L.A. Shepp and B.F. Logan (1974) The Fourier reconstruction of a head section. *IEEE Trans. Nucl. Sci.*, vol. NS-21, pp. 546-553.

[SHU95] R. Shu, Z. Chen, and M.S. Kankanhalli (1995) Adaptive marching cubes. *The Visual Computer*, 11(4), pp. 202-217.

[SMI96] J. Smit, M.K. Bosma, and J. Terwisscha van Scheltinga (1996) Metric Volume Rendering. *Virtual Environments and Scientific Visualization '96*, Springer Verlag, ISBN 3-211-82886-9, pp. 211-222.

[SUM98] R.M. Summers, L.M. Pusanik, and J.D. Malley (1998) Automatic Detection of Endobronchial Lesions Using Virtual Bronchoscopy. *Proc. of SPIE Medical Imaging '98*, 3338, pp. 327-335.

[TER95] J. Terwisscha van Scheltinga, J. Smit, and M.K. Bosma (1995) Design of an On-Chip Reflectance Map. *Proc. of the 10th Eurographics Workshop on Graphics Hardware*, volume EG95HW, pp. 51-55.

[TER96] J. Terwisscha van Scheltinga, M.K. Bosma, J. Smit, and S. Lobregt (1996) Image Quality Improvements in Volume Rendering. *Lecture notes in computer science, VBC '96*, ISSN 0302-9743, 1131, pp. 87-92.

[TIE98] U. Tiede, T. Schiemann and K.H. Hohne (1998) High quality rendering of attributed volume data. *Proc. IEEE Visualization '98*, Los Alamitos, CA, IEEE Computer Society Press, pp. 255 - 262.

[VAN93] A. Van Gelder and J. Wilhelms (1993) Rapid exploration of curvilinear grids using direct volume rendering. *Proceedings Visualization '93.* (Cat. No.93CH3354-8). IEEE Comput. Soc. Press, Los Alamitos, CA, USA; 1993; xv+423 pp. 70-77.

[VLA99] M.T. Vlaardingerbroek and J.A. den Boer (1996) *Magnetic Resonance Imaging: theory and practice*. 2nd ed., Berlin: Springer, ISBN 3-540-64877-1.

[WAT93] A. Watt (1993) 3D Computer Graphics, 2nd ed. Addison Wesley, ISBN 0-201-63186-5.

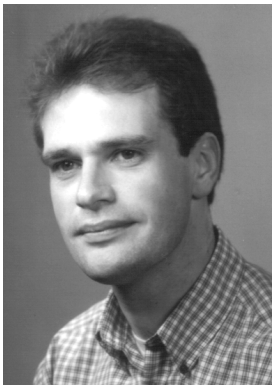
[WES89] L. Westover (1989) Interactive Volume Rendering. *1998 Chapel Hill Volume Visulization Workshop*, 9-16.

[WIL92] J. Wilhelms, and A. Van Gelder (1992) Octrees for Faster Isosurface Generation. *ACM Transactions on Graphics*, 11(3), pp. 201-227.

[WIL95] J. Wilhelms and A. Van Gelder (1995) Multi-dimensional trees for controlled volume rendering and compression. *Proc. 1994 Symposium on Volume Visualization*. IEEE, New York, NY, USA, pp. 27-34.

[ZUI95] K.J. Zuiderveld (1995) *Visualization of Multimodality Medical Volume Data using Object-Oriented Methods*. Ph.D. thesis, University of Utrecht, ISBN 90-393-0687-7.

CURRICULUM VITAE



Marco Bosma was born on October 28, 1970 in Losser, The Netherlands. In August 1995, he received his M.Sc. degree in Electrical Engineering from the University of Twente, Enschede, The Netherlands. His masters project was performed at the Laboratory for Network Theory and VLSI Design of the Control Systems and Computer Engineering group and dealt with 'Super Resolution Volume Rendering'.

In November 1995 he joined the Laboratory for Network Theory and VLSI Design as a Ph.D. student to work on improvements of volume rendering algorithms. This work was supported by Philips Medical Systems.

Since May 2000, he is working as a research scientist at the Laboratory for Network Theory and VLSI Design.

INDEX

N

O

P

R

S

T

V