Real-Time Anatomy Illustration Using Non-Photorealistic Rendering Styles

by

David Boothman

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfilment

of the requirements

for the Degree of

Master of Science in Computer Science

Supervisor: John Dingliana

Submitted to
University of Dublin, Trinity College
May 2018

Declaration

I, David Boothman, declare that the following dissertation, except where otherwise stated, is entirely my own work; that it has not previously been submitted as an exercise for a degree, either in Trinity College Dublin, or in any other University; and that the library may lend or copy it or any part thereof on request.

Signature:

Date:

Summary

The research will look at how medical illustrations can be produced using a volume rendering implementation that is capable of loading 3D medical data and displaying it on the screen for the user to interact with. The research will look at the different techniques related to non-photorealistic rendering that can be used within the application as well as different methods that can be used to increase the flexibility aspect of the implementation. The application will allow the user to choose from a list of techniques and be able to modify certain visualisation parameters of these techniques that will alter the rendering to suit their needs. The end result will be an efficient method of producing medical illustrations from volume medical data.

Acknowledgements

Thanks to my supervisor John Dingliana for all of his help during the research process and throughout the implementation of the application. Thanks to Tom Noonan for allowing me to use his volume raycaster as the base for my application. Thanks to my family for their continued support throughout. Also thanks to my classmates for their help and advice through my five years at Trinity College.

Abstract

Since the beginning of medicine and medical practice, artistic illustrations have been used as a way to educate and inform people about the human anatomy. These illustrations are known to be easier to comprehend and study than other forms of medical imaging such as scans and photographs. It is also known, however, that these artistic illustrations often take a large amount of time to create and require very skilled artists with prior medical knowledge who know what they're doing. The illustrations produced by these artists are then specific to a particular person or model. In this paper, the challenges around generating these artistic illustrations from real-world 3D data such as MRI and CT scans are investigated. The approach will allow for automated illustrative renderings of a given scan while giving the user the option of modifying certain visualisation parameters in order to render the 3D model to suit their need and preference. The different non-photorealistic techniques that can be used to aid the user in creating useful illustrations will also be investigated. These techniques would apply effects such as emphasising the more important features of the dataset to provide a focus for the rendering and suppressing extraneous details while maintaining shape and structure to provide context. This will be achieved by providing the option of choosing from a number of different artistic styles and applying them to different parts of the dataset such as to the skull or skin. The user will also have the option of modifying certain visualisations parameters related to these styles thus adding another element of flexibility to the application. A transfer function would then be used to determine the colour and opacity values within the volume allowing the user to choose which parts of the volume are visible and which parts are not.

List Of Figures

Table Of Contents

Chapter 1 Introduction

The art of illustration is one of the most significant and widely used forms of visual communication that exists today. It can tell a story, it can teach a skill or it can inspire a generation. The use of illustrations really came to the forefront with the invention of the printing press in the 15th century allowing for more books to be spread far and wide. Today, illustrations can be found in magazines, books, TV shows, films and a host of other places where someone is trying to get a point across to another person. Particular areas of interest where artistic illustrations have been used are medical imaging and the visualisation of medical data. For centuries, skilled artists with a background in medicine have been required to create detailed illustrations of the human anatomy for use in medical textbooks and journals with the aim of teaching and informing people about its structure. These illustrations have been found to be much easier to comprehend than photographs and scans hence their popularity.

However, it can take artists a long time to produce these detailed illustrations and ensure that they are as accurate and easy to comprehend as possible. The demand for visual representations of the human anatomy has also increased with the advances in technology and the increased knowledge in relation to new and old diseases, injuries, medical conditions etc. A much more accurate and less time-consuming way of creating medical illustrations is to render a 3D model of a part or all of the human anatomy from an MRI or CT scan before applying some techniques to simplify the model and make it easier to understand. These techniques would be centred around the area of non-photorealistic rendering which has long been used within computer graphics in order to reproduce artistic styles and to give a model a more simplified and colourful appearance. By combining anatomy data and non-photorealistic rendering to create a 3D model and then using this model to produce artistic illustrations, it is almost guaranteed to result in a highly accurate and easily understood technical diagram perfect for use in any of the areas that have been mentioned previously.

The aim of this research is to produce a proof of concept interactive, real-time application that takes a medical dataset made up of voxel data as input and renders it to the screen as a 3D model using a volume rendering implementation. The user can then interact with the model in real-time by rotating or zooming in/out within the rendering window. Alongside the rendering window lies a toolbar where the user can modify some visualisation parameters, also in real-time. These visualisation parameters include a number of different non-photorealistic styles that can be applied to the rendering with the ability to apply different styles to different sections of the dataset (i.e. skin, bone). Certain styles use thresholds that can also be modified by the user. A transfer function is also available to the user allowing them to change the opacity of these different sections of the volume making them fully or partly visible or completely invisible.

The scope of this research to a larger extent will be limited to two main areas with volume rendering and non-photorealistic rendering being the key aspects of the study. Within these areas, the research will focus on methods and techniques that are computationally efficient enough to be used in real-time as well as being capable of being used along with other techniques within a single rendering. This research will not directly compare against some of the competing areas of research because of its focus on being a proof of concept for a highly flexible real-time design and its use of techniques that have not been seen before within a single rendering. The report will proceed to highlight some of the related work to this research in the areas of volume rendering and non-photorealistic rendering before going on to discuss some of the key design decisions that were made during the research and application building phases. The report will go on to explain the implementation process for the application including a detailed view of the frameworks and libraries used and a description of the techniques that were chosen. This will be followed by a thorough analysis of the results that the application produces and a brief discussion of the conclusions from the research.

Chapter 2 Related Work

Volume rendering is a visualisation technique that has long been used as a way to allow users to view 3D volumetric datasets. There are two main techniques widely used for volume rendering which are isosurface rendering and direct volume rendering (DVR). Isosurface rendering involves extracting surfaces of equal value, known as isosurfaces, and rendering them to the screen. The DVR approach involves rendering the whole dataset as one single block of data. The 3D datasets used in volume rendering are usually an array of 2D slice images that have been taken by an MRI or CT scanner and now make up a volumetric grid. This grid is made up of vector values (x,y,z) known as voxels which are basically the 3D version of pixels. These vectors can be placed in a 3D array within the program and then converted into a 3D texture so that it can be loaded onto the GPU and used in the shading model. There are many different techniques used for volume rendering that will be discussed in this section along with many of the techniques used in Non-Photorealistic Rendering (NPR).

2.1 Volume Rendering Techniques

As mentioned above, there are two main techniques used for volume rendering, isosurface rendering and DVR. Numerous approaches have been developed based on both of these techniques each with different aims, ranging from simplicity and efficiency to high-quality and accuracy. The first approach that really introduced the concept of isosurface rendering was the marching cubes algorithm [LC87] published in 1987 by Lorensen and Cline. This algorithm works by using a volume contour to intersect the edges of a data volume grid in order to create a surface to display on the screen. A patent on the marching cubes algorithm along with an ambiguity in some of its cube configurations led Muller and Wehle [MW97] to release the marching tetrahedra algorithm in 1991. The marching tetrahedra algorithm is similar to marching cubes except that it computes nineteen edge intersections per cube instead of twelve leading to it being more computationally expensive but with a better sampling resolution.

Neither of the algorithms mentioned are efficient enough to provide interaction with 3D volume datasets which led to the development of more powerful methods such as the adaptive marching cubes algorithm [SCK95] published in 1995. This algorithm significantly improves the performance by reducing the number of triangles that represent a surface. Another algorithm which also saw improvements was an octree-based algorithm that was introduced in 1992 by Wilhelms and Van Gelder [WG92]. Both the number of triangles used and execution time were reduced by applying the marching cubes algorithm on larger cells in areas where the isosurface was mostly flat. More recently, much faster methods have been proposed including Dietrich et al [DSSCNS09], who proposed a modification to the grid on which the marching cubes algorithm operates and managed to not produce any degenerate triangles and can be easily integrated with existing algorithms.

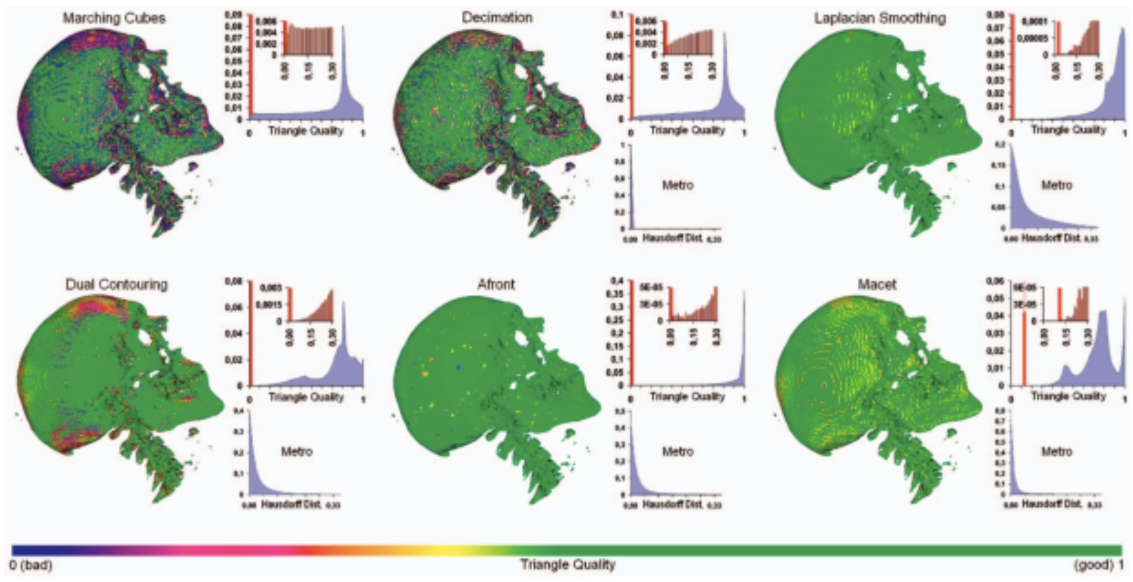


Figure 1: Dietrich et al. showing their Macet technique compared to other methods.

An alternative approach that was put forward by Westover [Wes90] in 1990, works by creating an image plane footprint for each data sample and then 'splatting' the footprint like a snowball onto the image plane. In this method, the voxel cube is traversed in object space in either back to front or front to back order. The gradient and opacity can either be computed here at the centre of the voxel cube or at the voxel position. The footprint can be calculated based on the size and orientation of the cube with each pixel in the footprint having the same colour. The footprints are then splatted on to the screen in order to render the volume to the 2D screen. This particular method sacrifices the quality of the rendering for speed and at the time was faster than a lot of the rendering algorithms around.

Another form of DVR technique is raycasting which involves sampling the volume at different intervals in order to render it to the screen. raycasting is similar to ray tracing which is used in surface-based graphics but that only primary rays are used while secondary rays are discarded. In this method, two passes are used with the first pass rendering the front and back faces of the bounding box off-screen in order to get the start and end points of each ray and the second pass using these start and end points to cast rays through each pixel on the screen. As the ray is cast through each pixel, a sample is taken at different intervals along the ray. At each sampling point, the scaler value is mapped to a colour and opacity using a pre-defined transfer function. The algorithm iterates until the ray is terminated and the volume is rendered to the screen. The concept of raycasting to render volume models on screen was first introduced by Scott Roth [Rot82] in 1982 where he demonstrated some of the possibilities of raycasting for volume rendering. This and many other algorithms, however, were ultimately let down by the main weakness of raycasting algorithms at the time which was that they were way too computationally expensive. More recent algorithms have been able to use GPU's and 3D textures to ease this computational burden while retaining the high-quality rendering of a traditional raycaster. An example of this can be seen with the GPU based method proposed by Stegmaier et al [SSKE05] in 2005 which is made up of two parts, a framework that controls the rendering process and a set of shaders that can be loaded and modified at runtime. The rendering process consists of two parts, the first involves rendering the background with a full-screen quad while the second involves rendering the volume to the screen while drawing over some parts of the background. The real advantage in this method comes from each bounding box vertex having texture coordinates identical to the vertex positions allowing each fragment to easily compute the parametric ray of sight while reducing the overall number of rays required.

2.2 Non-photorealistic Rendering

Much of the early work within computer graphics rendering was centred around photorealism which focused on replicating camera optics and producing a realistic image. Artists gradually began to apply their own styles and techniques on renderings to give them a life of their own. This became known as non-photorealistic rendering and incorporates many different artistic styles inspired by paintings, cartoons, pen and ink and many other artistic techniques. NPR is often used in medical illustration through simplification and enhancement techniques with the aim of making datasets of the human anatomy easier to comprehend and learn. There are thousands of NPR techniques that could be used and so it can be difficult to choose from such a wide abundance of options. However, according to research conducted by Redmond [RD092], most of these techniques fall into four categories: edge enhancement, shading abstraction, textural abstraction and colour transfer. Accordingly, for the purpose of this research, it was decided to focus on these areas of NPR calling them edges, stylised shading and textural abstraction as well as having the transfer function to account for the colour transfer category.

2.2.1 Edge Detection

Edge and line extraction techniques are useful for emphasising the structure of objects, providing shape cues to aid with the perception of the model as well as showing discontinuities within the rendering. One of the most well known and effective edge detection algorithms is Canny [Can86] edge detection which although computationally very expense, produces some of the best results when compared to other algorithms. Canny works by first blurring the image to smooth it and eliminate any noise. After blurring the image the strength of each edge is found by using an edge operator such as the Sobel filter which is one of many techniques that could be used to measure the gradient of the image. The Sobel filter is an edge detection technique that finds the gradient in both the horizontal and vertical directions before using a threshold to find the edges of the images. Canny then uses non-maximum suppression to trace along the direction of each edge and set the values of all of the pixels that aren't considered to be an edge to 0. The final step involves hysteresis which uses two thresholds to eliminate unwanted edges. Another widely used technique is Difference of Gaussian (DoG) [MH80] edge detection which focuses on improving edge visibility and increasing the detail within images. It works by applying two Gaussian blurs with different blurring levels to the image and then subtracting them which removes the high-frequency components e.g. noise as well as low-frequency components representing areas with little or no edges.

All of the above are examples of image space algorithms, however, it is also possible to extract more advanced edges from the model in object space. Object space methods work by iterating through the dataset and choosing 3D edges that satisfy certain criteria. Although computationally much more expensive than traditional image space techniques, interesting edge types can be identified such as ridges, contours and silhouettes. Silhouettes are useful for conveying the spatial relationship between different parts of the model that are rendered and by comparing surface normals of the model, silhouette edges are easily identified. Hertzmann and Zorin [HZ00] presented a method for silhouette detection that involves interpolating between the vertices of edges which have dot products of differing sign to compute a silhouette edge in a piecewise fashion. This method creates silhouettes that work really well for getting the overall outline of an object but it also shows that silhouettes alone are not enough to convey the full scene. Hadwiger et al. [HSSBG05] propose a real-time isosurface ray-casting method that could display ridge and valley edges through the use of curvature based transfer functions. It works by having each pixel that corresponds to a ridge or valley area be identified by a curvature based transfer function. These pixels are then mapped to colours so that the ridge and valley edges can be displayed. Burns et al. [BKRFD05] put forward a technique to extract a number of different linear features such as contours and suggestive contours directly from the volume. This method is computationally very expensive and so can only be performed on a smaller section of the volume but it is capable of real-time interactive performance under these conditions.

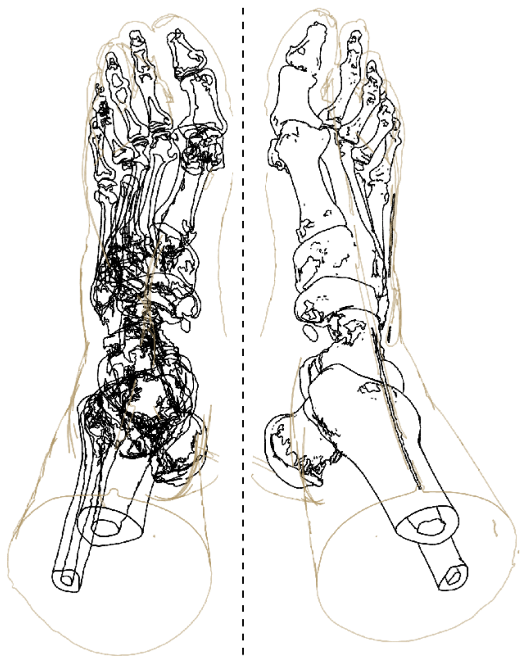


Figure 2: Burns et al. show an image of the results of their line extraction technique.

2.2.2 NPR Styles

In the past artists have worked for hours to implement their own styles on a canvas to produce drawings and paintings but now many aspects of these styles have been attempted to be recreated in a rendering environment in an attempt to streamline the process. Although these recreations may never reach the accuracy or quality of the artist's implementation, they can be produced in a much faster way. These styles range from methods applying different lighting and shading models to the scene, to methods that apply cartoon-like looks to the scene in order to make them easier to comprehend.

Shading consists of adding value to a rendering in the form of highlighting structure, space and lighting. Shading is important because it simulates the colour of objects from the current viewpoint giving the user more of a spatial perspective on the scene. One of the most popular forms of shading is the Phong illumination model [Pho75] which was introduced by Phong in 1975. It is the goto baseline shading algorithm for many volume rendering applications and it is based on the premise that light hitting a surface has three distinct components namely ambient, diffuse and specular. Combining these three components results in an efficient reflection model. Another popular shading technique known as cool to warm shading has long been used in technical illustration as a way to increase depth perception and attract user attention to an otherwise standard part of the image. One of the first cool to warm shading models was introduced by Gooch et al. [GGSC98] who managed to achieve this effect in a rendering. This effect is achieved by limiting the colours used in the shading model to a cool and a warm colour and then using the dot product of the light source vector and the surface normal to distinguish between colours. They also provide details on how to increase the depth perception of the scene by changing the position of the light source.

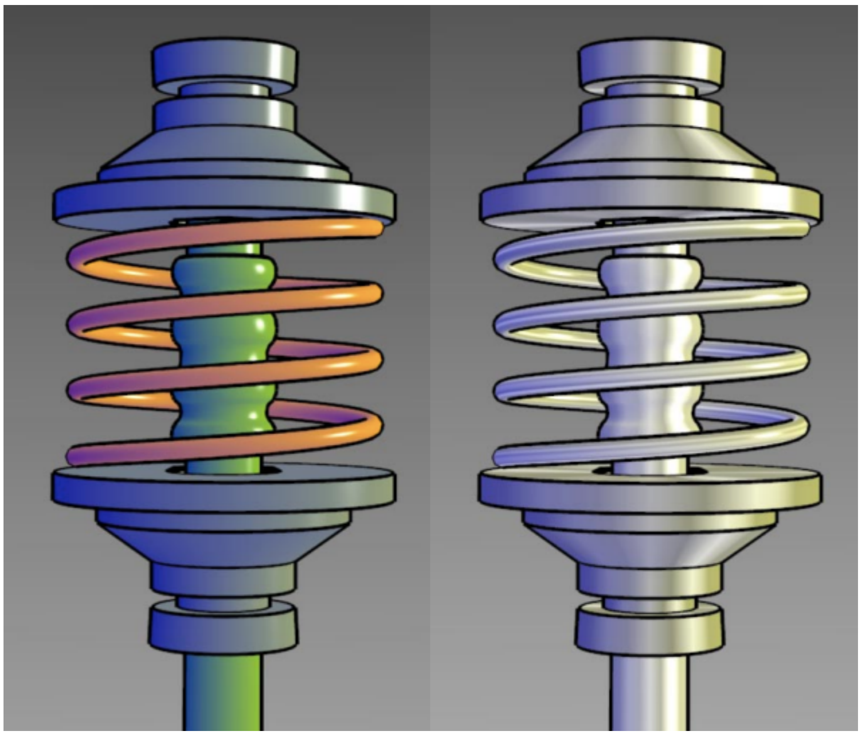


Figure 3: Gooch et al. cool-to-warm shading technique.

Another form of shading, which is a visibly distinctive style that focuses on removing extraneous detail, became known as toon shading. The concept of toon shading was first introduced by Decaudin [Dec96] who used a variation of the Phong illumination model in order to produce cartoon-like effect. This method removes the n.l term of the Phong algorithm in order to get more flat colours without gradient which are more characteristic of cartoon drawings. Lake et al. [LMHB00] presented an improved designed for real-time 3D animation and works similarly to Decaudin's technique except that when the dot product crosses the threshold of 0.5, a different colour is chosen. In addition to this, the threshold can be changed and bins can be added so that the model now has a number of uniquely coloured regions.

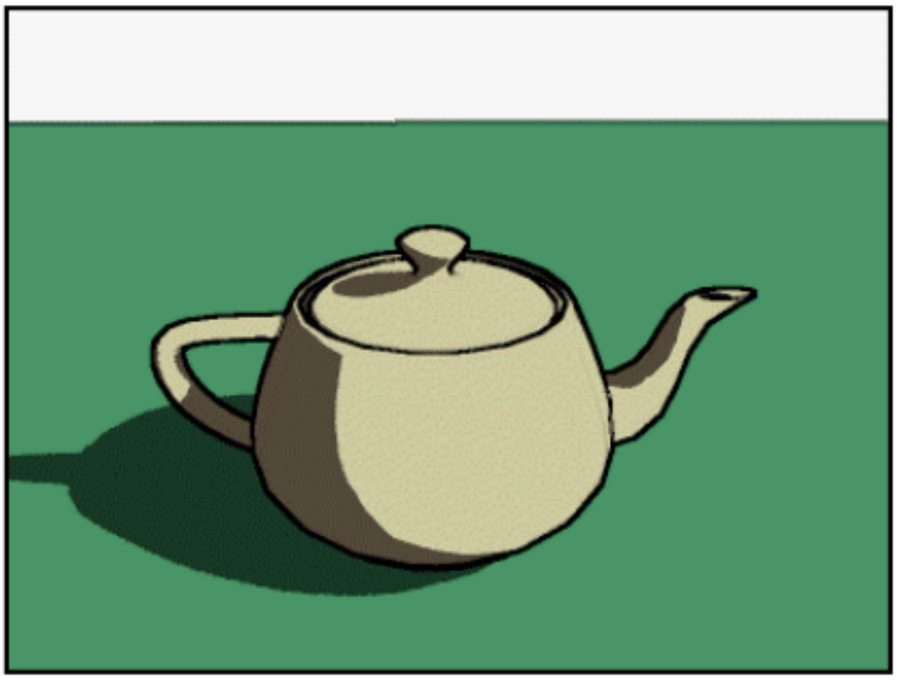


Figure 4: Decaudin's 'cartoon looking' technique.

Barla et al. [BTM06] put forward an extension to the toon shading technique which worked by replacing the 1D texture used in the existing implementation with a 2D texture. This method still uses the dot product of the light source vector and the surface normal which corresponds to the horizontal axis of the 2D texture while 'tone detail' corresponds to the vertical axis. The 2D texture is then used to map voxels in the model to a colour. Sloan et al. [SMGG01] introduced a "lit sphere" model which can map the texture of a sphere onto a model in order to provide shading. This allows an artists impression of a certain type of shading to be mapped exactly to a 3D model to produce some very impressive results.

Other techniques that have been popular in NPR implementations include 'hatching' introduced by Praun et al. [PHWF01] which consisted of drawing surfaces using groups of strokes to the convey the shape, tone and features of a model. The shading of the model is easily controlled by the local density of hatching strokes while the direction of these strokes usually corresponds to the curvature of the surface. Another technique of interest in NPR is the concept of style transfer functions put forward by Bruckner and Groller [BG07] which allows multiple different styles to be used within a single rendering. Instead of having a traditional transfer function which maps colour values to the rendering, they propose to replace the colours with styles allowing the user to apply different styles to different parts of the rendering.

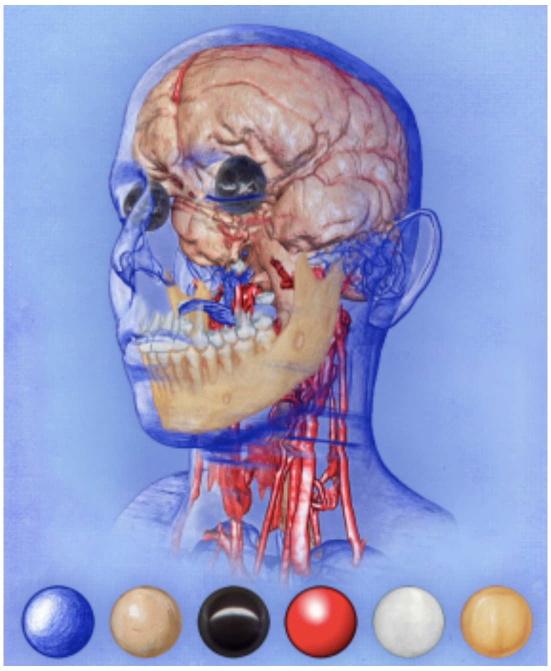


Figure 5: Bruckner and Groller's style transfer lookup technique.

2.2.3 Abstraction

One of the key concepts of NPR is removing extraneous detail and presenting an overall simplified image making it easier to understand. This had led to a number of techniques being developed with the goal of abstraction in mind. One of the most well known and used forms of blurring and abstraction is the Gaussian filter. The Gaussian filter works by convolving an image with a kernel of Gaussian values in order to reduce noise and detail within the image. Another simple form of abstraction can be seen with a box blur which simply blurs the entire image by assigning each pixel the average value of all of the surrounding pixels.

One technique that has been widely used in volume rendering but no as much in medical illustration is the Kuwahara [KHEK76] filter which is a 2D edge preserving blurring technique. This method works on a pixel by pixel basis by splitting up the image into four overlapping regions of pixels that surround the current pixel with each region containing the current pixel. The colour of the current pixel is mapped to the mean of the adjacent region which has the lowest variance resulting in the body of the rendering being blurred while the surrounding edges are maintained which has a great effect for medical data. Another technique with the goal of abstraction is proposed by Papari et al. [PPC07] and provides a generalisation of the Kuwahara filter. They state that all previous extensions of the Kuwahara algorithm suffer from the same issue; the output is not determined when the minimum variance is reached in more than one region. This method overcomes this issue by using different weighting and criteria for computing local averages. A more recent generalisation of the Kuwahara filter is the Anistropic filter proposed by Kyprianidis et al. [KKD09]. This method uses an elliptically shaped filter guided by a smoothed structure tensor in order to produce a painterly-like effect with overall sharper edges than many of the other methods that have been proposed.

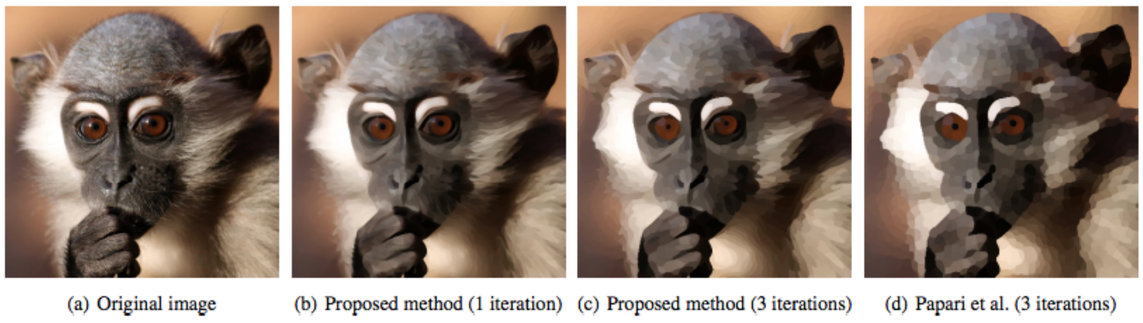

(a) Original image
(b) Proposed method (1 iteration)
(c) Proposed method (3 iterations)
(d) Papari et al. (3 iterations)

Figure 6: Anistropic Kuwahara filter compared to Papari et al. method.

Tomasi and Manduchi [TM98] introduced the concept of a bilateral filter which also smooths images while preserving the surrounding edges. This is accomplished through a process of the non-linear combination of nearby image values. Instead of using the traditional domain filters that give weight to pixel values with coefficients that fall off with distance, they propose to use range filters whose coefficients fall off with dissimilarity. Another effective abstraction technique was introduced by Comaniciu et al. [CM02] and makes use of the ‘mean shift’ pattern recognition method. It uses discontinuity preserving smoothing which, unlike traditional smoothing which averages the surrounding pixels indiscriminately, reduces the amount of smoothing near edges. The difference between the bilateral and the mean shift filter is that the bilateral filter uses a fixed window for its implementation whereas mean shift adapts itself to the image.

Chapter 3 Design

This chapter outlines the design that was chosen in order to answer the research question that was proposed of the real-time non-photorealistic volume rendering of medical data. The chapter includes the research approach, the thought process behind the implementation, the criteria used for selecting styles that were used as well as some of the limitations of the design.

The research question focused on creating a proof of concept application that was capable of rendering a medical dataset while allowing the user to interact with the model in real-time and providing them with a high level of control and flexibility over the rendering. One of the goals of the research was to give the user control of the different techniques that were being used on the dataset through the ability to change certain parameters related to these techniques. The end goal was to have a real-time application that would allow for the efficient creation of medical illustrations based on single or multiple NPR techniques. The reason for using NPR techniques to enhance the rendering of medical data is because both medical data visualisation and NPR have similar goals which are to effectively convey information and placing emphasis on important features within the data. This had led to more and more NPR techniques featuring in medical data and scientific visualisation methods in recent years such as the methods proposed by Rheingans [RE01] and Nagy [NSW02].

The research approach taken in this dissertation was one where existing research methods were examined and analysed with a select few chosen to be used in the implementation. Rather, research was done into the existing techniques that satisfied the criteria for the research question which will be discussed at a later point in this chapter. The research focused on finding existing techniques that could be used together in a single rendering and thus creating a novel combination of techniques. There is also a novelty in the way these techniques can be used together with different techniques being used for different sections of the volume. The approach involved looking for techniques that have been used before in NPR but have not explicitly been used in medical illustration. Techniques suitable for medical data would ideally render the data in a way that places emphasis on the more important features while limiting the impact of the lesser details [PB07]. Placing emphasis on the more relevant features of the dataset is important because it focuses the user's attention towards these areas. This can be done applying contours and strokes to that particular part of the dataset in order to sharpen it and bring it into focus. Removing or lessening the effect of the unimportant features of the dataset is also important because it takes the user's attention away from these areas. This can be done using abstraction through a blurring effect to minimise the features in the area and bring it out of focus. It is important, however, to not abstract too much detail away so that this section of the dataset still provides some context for the overall scene. Both of these are necessary components for effective illustrative rendering and became the central focus of the research.

The design process behind the implementation began by determining the features that the application would need in order to allow the user to efficiently produce these artistic illustrations. In keeping with the research question, the application would need to be able to render volume data in real-time and be capable of user interactivity giving them the ability to alter the rendering in a number of ways. These initial design choices meant that a volume renderer capable of dealing with data from a CT scan would be required and that any techniques that were going to be used would have to satisfy certain criteria. These criteria were set by the requirement for real-time interactivity with the dataset meaning that none of the NPR techniques could be too computationally expensive. Therefore, any methods that lacked in computational efficiency were not considered for this implementation.

A two-level volume rendering strategy was chosen during the design process of the application. Two-level volume rendering was first proposed by Hauser et al. [HMIG01] in 2001 as a means of applying different volume rendering techniques to different subsets of the data. This method was initially based on applying DVR to one section of the dataset while applying maximum-intensity projection to another. More recent techniques for two-level volume rendering [Cor10] looked at using different NPR styles for different sections of the dataset. This was the approach that was taken in this implementation. Two-level volume rendering was used to add another level of flexibility to the application by allowing the user to apply the same range of techniques to two different subsets of the data thus opening up a greater variety of illustrations that could be produced. Implementing this approach into the design of the application was a relatively simple concept that involved calculating the intensity at each voxel and using a precise threshold value of 0.94 to determine which subset of the data the voxel belonged to with anything below 0.94 belonging to the skin of a dataset and anything above belonging to the bone of the dataset.

A design choice that was made as a direct result of the decision to implement two-level volume rendering was the concept of focus and context rendering. Focus and context rendering has long been used in medical illustration as a way to enhance the important areas of a dataset while using abstraction to remove the detail from the less important areas. This approach gives the user a clear understanding of the significant sections of the dataset through the focus area and also gives the user a view of how the focus area relates to the rest of the dataset through the context area. Implementing this in the design of the application was not a difficult process as due to the two-level volume rendering approach that was used, the dataset could be split in tow having one focus and one context area of the rendering. Using this focus and context rendering approach also meant that within the NPR styles that would be used in the application, there had to be some styles that enhanced the details of the dataset and also some styles that removed extraneous detail. This provided further evidence for the need for edge detection and textural abstraction techniques that will be discussed at a later point.

Another significant design choice was choosing to use only single-pass instead of multi-pass rendering. Multi-pass rendering involves rendering a scene multiple times and then combining the results together to form the final rendering. For example, a 3D model could be rendered to a 2D texture and have image space post-processing techniques performed on it before being rendering back onto the 3D model. Therefore multi-pass rendering can involve a combination of image and object space techniques. Single-pass rendering involves rendering the scene only a single time meaning that in a 3D scene only object space techniques can be used. Multi-pass rendering has the advantage of adding much more control over the final image as the individual components of the scene can be rendered separately meaning that any adjustments made are easily done so by simply re-rendering the individual parts. The downside of multi-pass rendering is that it requires a lot of disk space to store all of the individual components of the rendering as textures, while another disadvantage is that a strong knowledge of the compositing process is required to ensure that the individual components are combined in the right way. Compositing is basically the combinations of different sources into a single image. Single-pass rendering has the advantage of providing significant savings on CPU overhead by reducing the number of draw calls, which are a way of sending data to the GPU to be drawn, as well as requiring less texture swapping than multi-pass rendering. The downside of course of single-pass rendering is that only object space techniques may be used resulting in much less control over the final image.

Implementing a multi-pass rendering approach would involve using a popular technique in modern computer graphics known as Render To Texture (RTT). To understand how RTT works it is important to understand how images are rendered to the screen. OpenGL renders objects to a collection of buffers known as a framebuffer. By default, OpenGL renders to the screen also known as the default framebuffer but can also render to user-defined framebuffers known as framebuffer objects (FBOs). FBOs allow the application to render to different locations thus not affecting the main screen, perfect for multi-pass rendering. In order to implement RTT, the first step is to create an FBO and then bind it to the pipeline so that it is used instead of the default framebuffer. An empty texture is then created and bound to the FBO. The next step is to create a quad where the texture can be rendered to and vertex and fragment shaders so that post-processing techniques can be applied to the texture. RTT then works by binding an FBO, applying post-processing to the scene texture and then unbinding the FBO. This process is repeated until finally the default framebuffer is bound and the final scene is rendered.

Unfortunately, implementing multi-pass rendering was made difficult by the configuration of the original volume rendering implementation that was used. Due to an issue with binding and unbinding each FBO, the RTT approach proved difficult to implement. Ultimately it had to be concluded that the time that would have to be spent working out the issues with implementing the multi-pass renderer would not be worth it when compared to what the research would gain. This design choice was taken after significant time was spent attempting to convert an originally single-pass renderer into a multi-pass renderer. This was significant from both a research and an implementation standpoint as it limited the range of NPR techniques that could be used to strictly object space methods. This meant that all image space techniques were eliminated due to the inability to render the texture of the volume to a 2D texture and apply the screen space methods on the 2D texture in separate passes. However, the lack of multi-pass rendering simply led to the use of more object space techniques and to the exploration of the adaption of screen space techniques such as blurring and Kuwahara to be used in object space.

One of the goals of the research was to give the user the ability to pick single or multiple NPR styles and apply them to the rendering. To satisfy this goal it was decided to present the user with a list of the styles they could choose from and allowed them to adjust a parameter of the style. The styles were split up into three categories namely edges, stylised shading and textural abstraction. The user would be able to apply a style from each category and they would also be able to do this for different parts of the volume i.e. apply one set of styles to the bone and another set of styles to the skin. This was important as the user needed to have the ability to focus in and alter different parts of the volume separately adding flexibility to the application. Another key aspect in the implementation was to give the user the ability to adjust the transfer function so that they could choose to view different parts of the volume i.e. only the bone or a small bit of skin and bone. Having the ability to adjust the transfer function at runtime gives the user much more control over the rendering and can be seen used in some of the more popular medical renderers such as Voreen [MRMH09]. In terms of UI for the application, the user needed to be able to choose and alter the style of the rendering while also being able to see the effect of this rendering and manipulate the model in real-time. This led to the decision to choose a large rendering window where the model was located with a toolbar along the side of the window where the user could adjust different parameters.

Chapter 4 Implementation

Many applications exist today with the aim of performing volume data visualisation so one of the key goals of the implementation was to differentiate the application from the rest. To accomplish this, a number of different NPR techniques would have to be available for the user to choose from with at least some of these techniques never having been used together in a single implementation. Having the application run in real-time and maintain a high level of user interactivity was also an important goal and meant that any techniques used would have to be lightweight in computation. The user interface had to be simple in its design while providing flexibility to the application by allowing the user to alter multiple parts of the rendering and observing these alterations in real-time. This chapter discusses the implementation of the above goals as well as providing in-depth descriptions of each NPR technique used and the proposed adaption of certain image space techniques to object space.

4.1 Volume Renderer

In order to render volume data, an efficient volume rendering implementation is needed. The volume rendering implementation used in this application was a volume raycaster created by Tom Noonan. The volume raycaster was implemented using Microsoft Visual Studio C++ and used the OpenGL graphics library as part of the rendering process. The renderer worked by first initialising the application by using the standard OpenGL techniques glViewport() and glLoadIdentity() which are used to create the rendering window and the view matrix respectively. The camera model is then initialised and contains various functions to zoom, rotate and translate the rendering scene. The shading manager is initialised which is tasked with loading and compiling and producing a shader ID that is then used at a later stage by the raycaster. The voxel reader is also initialised which is in charge of reading data from .mhd and .raw files where the CT scans are stored. The data is then loaded into a buffer before being stored in a VolumeDataset variable.

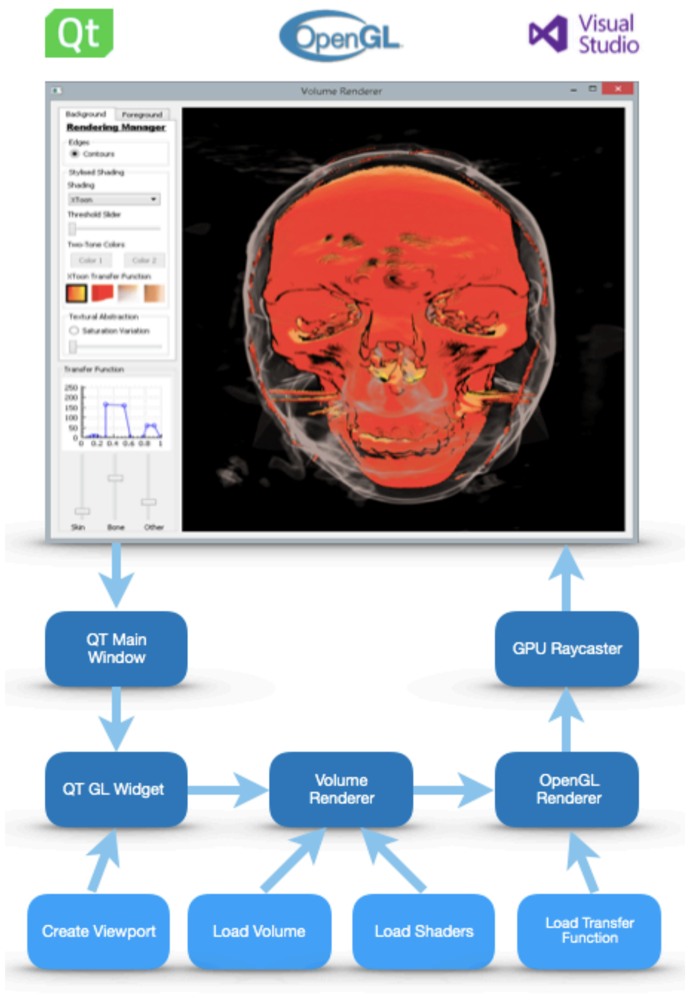


Figure 7: The layout of the UI with a flow of the application design.

After all of this initialisation is completed a 3D texture is created from the VolumeDataset variable that will store the texture of the CT scan dataset. At this point, the original transfer function is also initialised by loading in the .tfi file containing the intensity, colour and opacity values for the transfer function and creating a 1D texture that will be used in the shaders. The raycaster then passes a number of different parameters to the shaders using the glUniform command which modifies the value at a given location which in this case is the shaders. The raycaster then creates vertices for the bounding box that will be used to cast rays through in the fragment shader to render the data. The fragment shader is where the majority of the work is done and where the volume raycaster is actually implemented. Raycasting is achieved by first casting a ray through the volume taking two sample points along the ray where the normal between these two points is then calculated. The colour value is then found by performing a texture lookup using the transfer function and an index based on the current voxel location in the volume. The normal, position and colour can all then be used to create and apply NPR styles to the current voxel. After this, the position of the ray is then moved based on a pre-assigned ray step size and the process is repeated.

4.2 User Interface

The user interface of any application is hugely important as it is the part of the application that the user will initially see and interact with. This means that the UI has to be relatively simple in its design while still providing all of the functionality necessary to alter the rendering. Using a complex design would put first time users off of using the application and would probably require a detailed user manual so a straightforward design was chosen. The functionality required to make a highly flexible volume renderer consists of giving the user the ability to change what parts of the volume they are viewing via the transfer function, the ability to select single or multiple NPR styles to apply to different parts of the volume and the ability to change certain parameters for these styles.

4.2.1 Qt GUI

Today there exists a vast amount of graphical user interface (GUI) frameworks ranging from the easy to use with a low amount of functionality to the much more difficult to use with a high amount of functionality. As a result, choosing the right GUI framework for an implementation can be difficult but one of the more popular frameworks that was capable of providing all of the functionality needed with a relatively small learning curve was the Qt framework. Qt is a cross-platform framework that is mostly used as a graphics toolkit for applications and because of its extension tool for Visual Studio, it was easily integrated with the existing implementation. The Qt tool automatically adds several features that are key components in any Qt application namely, the meta-object compiler (moc), the user interface compiler (uic) and the resource compiler (rcc). The moc reads C++ files looking for classes with the Q_OBJECT macro declaration so that it can build a source file for those classes. The uic reads the .ui file created by the Qt Designer, the Qt tool used for designing creating the layout of the GUI, and builds a corresponding C++ header file. The rcc is used during build time to embed resources into the application by reading data contained in the .qrc file. Using this extension tool allows the layout of the UI to easily be changed with the Qt Designer providing a preview of what the UI will look like inside the application.

4.2.2 The Layout

The layout of the UI was decided by two key aspects of the functionality which were viewing and manipulating the 3D model that was rendered and changing the rendering itself and these led to the UI consisting of an OpenGL window with a toolbar along the side. The OpenGL window was where the data would be rendered to and allowed the user to click and drag on the screen to rotate around or use the mouse wheel to zoom in and out of the model. This is an important aspect as the user can have full control of the viewing angle of the rendering and can choose exactly what part of the volume they want to look at. The toolbar or sidebar of the UI was implemented using several widgets that provide different functionality to the application.

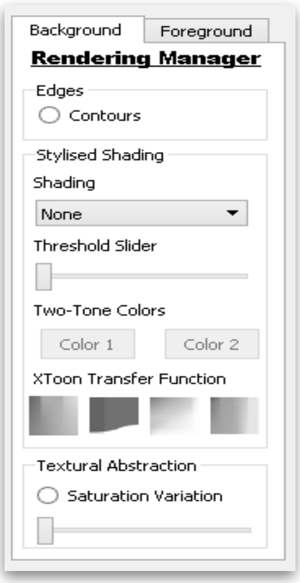


Figure 8: The tab that is shown to the user for each volume level.

The toolbar itself consisted of a general transfer function for the whole rendering, implemented using the QCustomPlot library, and two tabs, implemented using the QTabWidget class, with each tab offering the same functionality as the other but both dealing with different parts of the volume. The first tab would control the rendering styles and style parameters for the bone segments of the data whereas the second tab would control the rendering styles and style parameters for the skin segments of the data. The value used to separate the skin and bone segments was intensity as bone had a much higher intensity than skin in each dataset. Within each tab, there are three categories of style that the user can alter the rendering from which are edges, stylised shading and textural abstraction. The edge detection category lists the different types of edge detection that can be applied to the volume. The shading category lists the advanced NPR styles that can be applied to the volume and also contains parameters for these styles that can be adjusted by the user. The abstraction category lists the techniques that can be used to remove a certain amount of detail or colour and bring a section of the dataset out of focus.

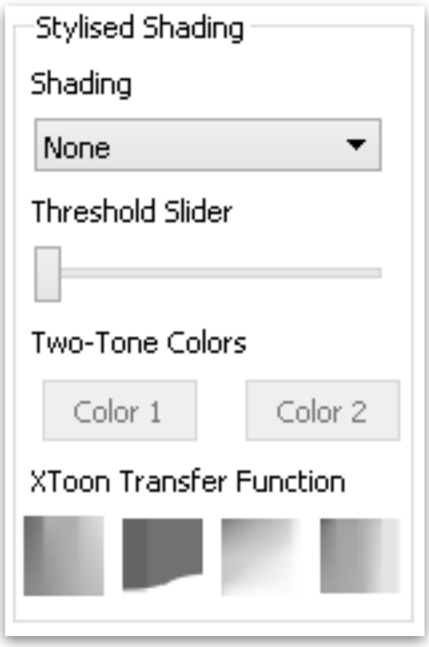


Figure 9: The stylised shading category and it's visualisation parameters.

In the edges and textural abstraction categories, the user can use radio buttons to choose if they wish to enable contours or saturation variation with the added flexibility of using a slider to modify the saturation level. In the stylised shading category, the user can select a style from a drop-down list, implemented using the QComboBox class, that they wish to apply to the rendering. The shading category also contains additional parameters that the user can change. Some styles contain a threshold that can be adjusted using a slider, implemented using the Slider class, once that particular style has been selected. The colours used for the two-tone style can be chosen by the user from a colour pallet allowing any colour to be used. The 2D texture used in the X-Toon style can also be selected from four different buttons containing image previews of the textures. The final part of the toolbar is the transfer function that consists of a QCustomPlot graph with three sliders underneath it. The graph plots the transfer function with opacity on the Y-axis and colour on the X-axis. The sliders control the opacity levels of the three peaks of the transfer function that correspond to bone, skin and teeth in the case of a skull dataset. Adjusting these sliders allows the user to control the visible and invisible parts of the rendering.

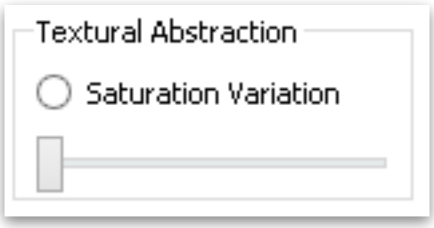


Figure 10: The textural abstraction category and the saturation variation style with slider.

4.3 NPR Techniques

The choice of NPR techniques or styles that the user is given in the implementation has a significant bearing on the capability and diversity of the application. If all of the styles available had the same or similar effects on the rendering then the application becomes very one dimensional and in cases where these types of styles have a poor effect on the dataset, there is no alternative option for the user. To combat this, the implementation used three categories of styles, edge detection, shading and abstraction, with the user being able to select one style from each category and combine them to render the data. On top of this, the types of styles in the shading category varied greatly ranging from stroke and line based techniques such as hatching to texture-based techniques such as X-Toon providing diversity and flexibility to the application. The techniques chosen also had to be object space-based and be computationally efficient for the single-pass volume renderer to work in real-time.

4.3.1 Edges

Edge detection and line extraction techniques have long been used in NPR as a way to add detail and convey information about shape such as the work produced by DeCarlo et al. [DFRS03]. Without edges and lines, it can be difficult to differentiate sections of the dataset from one another or to identify which section of the dataset is the focus of the rendering. For the skull dataset for example, without lines or edges, it is difficult to make out the eye sockets or the overall bone structure which are key parts in the study of the human skull. Another important aspect of lines and edges is apparent when small breaks or cracks in bone are the focus of the rendering which can be almost impossible to identify with edges. Due to the techniques being restricted to object space, none of the popular 2D edge detection methods could be used. This meant that any methods that were implemented would be in object space and would be relatively simple.

Contours

Contours or contour lines are often used to convey information about the shape of an object or structure through the drawing of lines on the surface of the object. Contours are drawn in areas where the surface of the model turns away from the view of the user and becomes invisible or in other words in areas where the surface is perpendicular to the view direction. The method for computing contours consists of locating areas of the model where the dot product of the surface normal *n* and the view direction vector *v* is close to zero. If the dot product is close to zero, this means that the two vectors n and v form a 90° angle and that the current sample point is within the boundary of a contour and should be darkened. The contour lines produced using this technique help to convey the shape and structure of the dataset. The algorithm used for this method is shown below while an example of the edges produced by this technique can be seen in figure [11] below:

If (Dot Product (V, N) <= 0.3)) { Voxel Colour = Black }

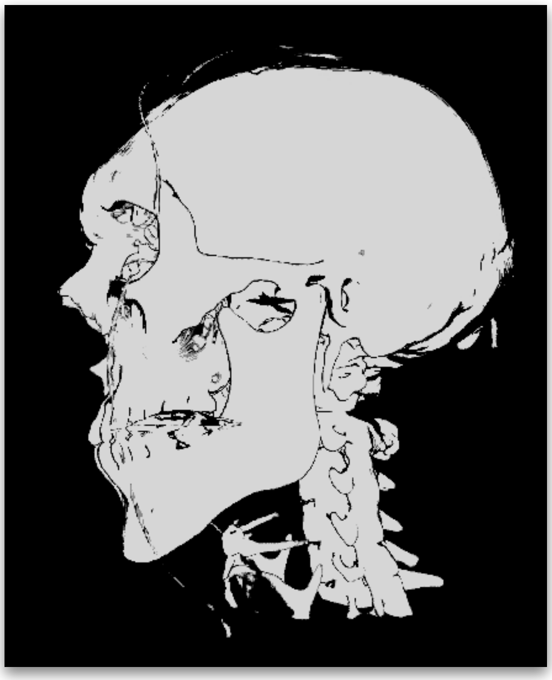


Figure 11: The skull dataset with contours applied to the grayscale rendering.

4.3.2 Stylised Shading

For years artists use shading as a way to give the impression of light and depth within their illustrations and now this same process is used within rendering to achieve the same goals. Langer et al. [LB00] put forward a perceptual hypothesis that "dark-means-deep" in relation to shading 3D surfaces stating that humans perceive brightness just as they perceive depth. Caniard et al. [CF07] conducted several experiments on the effects of lighting direction on shape perception and found that lighting conditions can have a strong effect on the perception of 3D objects. Both of these sources alone show that shading is a key part of conveying shape and depth information. The techniques chosen in this category are a mixture of old and new methods, some of which have been used together for the purposes of the volume rendering of medical data and some of which have not.

Phong Shading

The Phong illumination model [Pho75] was first published by Bui Tuong Phong in 1975 and refers to a local illumination technique that describes how a surface reflects light. Phong states that light hitting a surface has three distinct components namely ambient, diffuse and specular. The ambient component refers to the light that is scattered throughout the scene and is usually given a constant value of around 0.3. The diffuse component refers to the approximation of light reflecting from a surface that is non-glossy and is calculated by computing the dot product of the light source vector *L* and the surface normal *n*. The specular component refers to an area on the surface of the object, known as a highlight, that reflects light in a shiny way and is calculated by computing the dot product of the view vector *V* and the reflection vector *R* raised to the power of the shininess factor. Combining these three components results in an efficient reflection model that satisfies the criteria for working in a real-time rendering environment. The equation for combining these components can be seen below along with an example of the effect that this style produces shown in figure [12].

*Voxel Colour = (Original Colour * Diffuse) + Specular + (Original Colour * Ambient)*

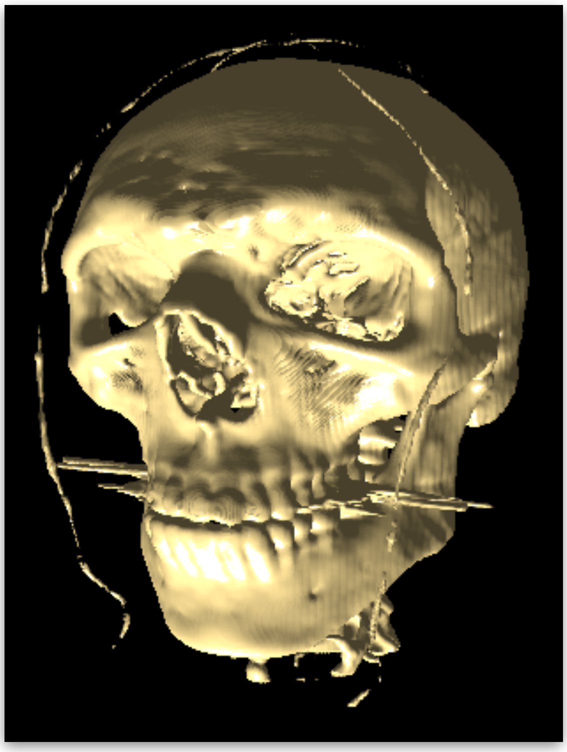


Figure 12: The The Phong illumination model applied to the skull dataset.

Toon Shading

Toon shading was first introduced by Decaudin [Dec96] in 1996 and refers to the production of cartoon-like images in a 3D scene. The algorithm was based on the Phong illumination model and described the goal of a stylised cartoon-like look with constant-sized outlines and uniformly coloured areas. Lake et al. [LMHB00] brought this a step further in 2000 and proposed a 'hard shading' technique where transition boundaries were found on the surface of the volume and each side of the boundaries were shaded using a colour. The method uses in this implementation was similar to this 'hard shading' technique. Firstly, the intensity of the light at the current sample point is calculated by computing the dot product of the light source vector L and the surface normal n. The diffuse lighting is then added to the surface colour to compute what would be the final colour. The intensity value is then compared to certain threshold values to see where it lies within four different boundaries. Each boundary changes the colour value of the current sample to a different extent with the higher intensities seeing less of a change than the lower intensities. The result is a very cartoon-like effect on the rendering where different parts of the data have different shades of colour depending on their position from the light source. The algorithm used to alter the tone at each voxel can be seen below while an example of the cartoon-like effect that is produced can be seen in figure [13]:

*Voxel Colour = Original Colour * Threshold*

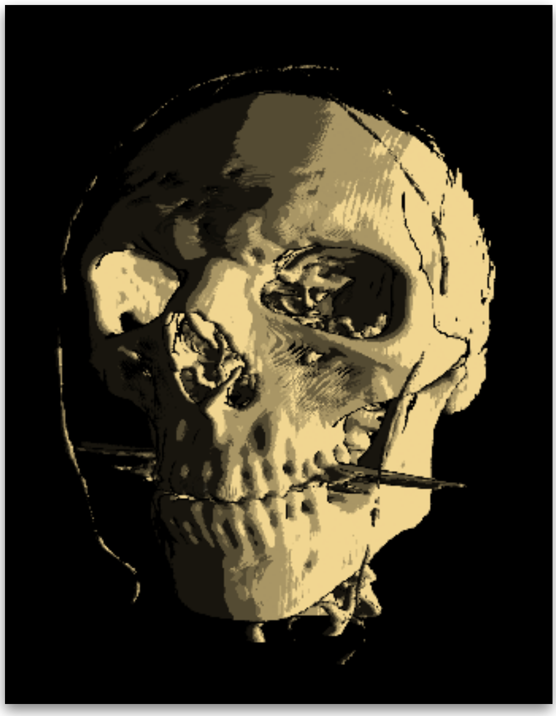


Figure 13: The toon shading effect applied to the skull dataset.

Two-Tone Shading

Two-tone shading is based on the tone shading or Gooch shading algorithm proposed by Gooch et al. [GGSC98] in 1998 with the only difference being the colour range is not limited to just cool and warm colours. Gooch et al. proposed a shading model that made use of cool-to-warm tones and was based on some of the characteristics of traditional technical illustration. This method also makes use of the Phong illumination model to control the lighting of the scene. The algorithm works by first taking two colours, one with a slightly cooler temperature (i.e. light blue) and one with a slightly warmer temperature (i.e orange), and then applying the diffuse lighting to each colour. The diffuse and specular components of the Phong model are then calculated from the surface of the volume. The final colour is then computed by using the diffuse component to decided how cool or warm the current sample should be. This is a way of varying the luminance of the volume based on the dot product of the surface normal and the light source vector. The reason that Gooch et al. decided to use cool and warm colours was because these shades play to the human senses and provide depth cues such as cool colours appearing to recede while warm colours appear to advance. This is why surfaces that are facing away from the light source are given a cooler tone while surfaces facing towards the light are given a warmer tone. This implementation uses two-tone shading instead of just Gooch to give the user the option of using colours with tones other than cool and warm. The algorithm used for the cool to warm tones can be seen below along with an image of the results produced by this method in figure [14] :

Voxel Colour = Minimum(Mix(Cool Colour, Warm Colour, Diffuse) + Specular, 1.0)

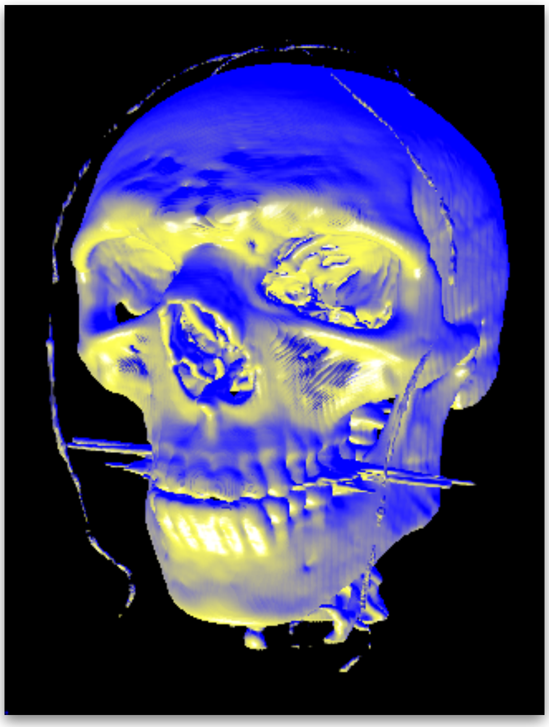


Figure 14: The Two-Tone shading effect applied to the skull dataset.

Hatching

Hatching is a technique where groups of strokes and lines are used to convey shape and structure in a rendering. These groups of strokes essentially take the place of colours and convey the lighting in the scene with the density of these groups determining the tone of the lighting. Praun et al. [PHWF01] proposed a real-time hatching method in 2001 that consisted of hatch strokes being pre-rendered into a series of mip-mapped images that correspond to different tones. The method showed that real-time rendering was possible using stroke-based textures. In this implementation, a much more basic and computational approach to hatching was taken known as cross-hatching. Cross-hatching is very similar to hatching only that lines are drawn at an angle to each other. Initially, the Phong illumination model is used to calculate the intensity of the light at the current sample. This light intensity is then used in a similar way to how it was used in the toon shading method, where it is compared to four different thresholds to see where it falls. Once the threshold level has been found, the gl_FragCoord variable is used to get the 2D location of the pixel that the current fragment is rendered to. Using these coordinates each diagonal direction is checked to see if the current fragment lies on a hatch that should be drawn. This approach does not convey shape and lighting as well as other approaches to hatching but it is very efficient and relatively simple to implement. One part of the algorithm is shown below that assigns a dark colour to the voxel if a hatch is present in the top left corner of the fragment. An example of the results produced by this method is also shown below in figure [15]:

If (mod (Fragment.x + Fragment.y, 10) == 0) { Voxel Colour = Black }

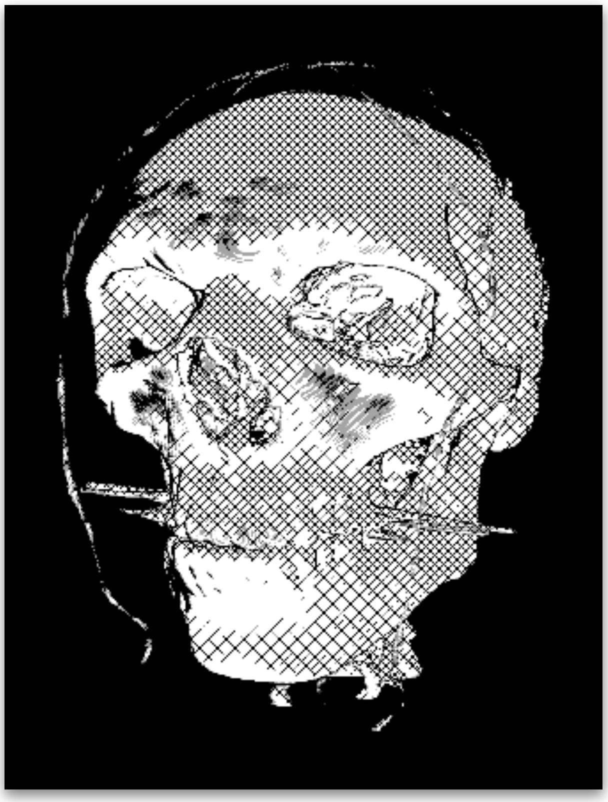


Figure 15: The Hatching technique applied to the skull dataset.

X-Toon

The extended toon shader or X-Toon was introduced by Barla et al. [BTM06] in 2011 as an extension to the traditional method for toon shading that used a 1D texture to map how tone varies with the surface orientation of the model with respect to the location of the light source. The method works by using a 2D texture instead of the 1D texture with the X-axis of the 2D texture corresponding to 'tone' with the coordinates computed from the dot product of the light source vector and the surface normal and the Y-axis corresponding to 'tone detail' (D) with the coordinates computed by either depth or orientation. For this implementation, firstly, a .tga file reader was needed to load in Truevision Graphics Adapter files containing the texture to be used for the 2D texture lookup. In the fragment shader, depth of field was used to calculate the coordinate for the Y-axis. This worked by first getting the *z* value for the position of the current sample. Two values zmin and zmax are also predetermined with zmin corresponding to the minimum z distance at which the level of detail starts to fall off and zmax corresponding to the point of greatest abstraction. The Y coordinate is then computed using the following formula:

$$\text{If } (z < z_c)\ \{\ D = 1 - \log(z/zmin)/\log(zmax/zmin)\ \}$$
$$\text{If } (z > z_c)\ \{\ D = \log(z/zmax)/\log(zmin/zmax)\ \}$$

The result of this formula and the dot product of the light source vector and the surface normal are combined to form an index to retrieve the corresponding colour value from the 2D texture. This implementation provides the user with a choice of four different 2D textures each with their own level of detail and colour values. An example of the results produced by this method can be seen below in figure [16] which also contains the 2D texture that was used:

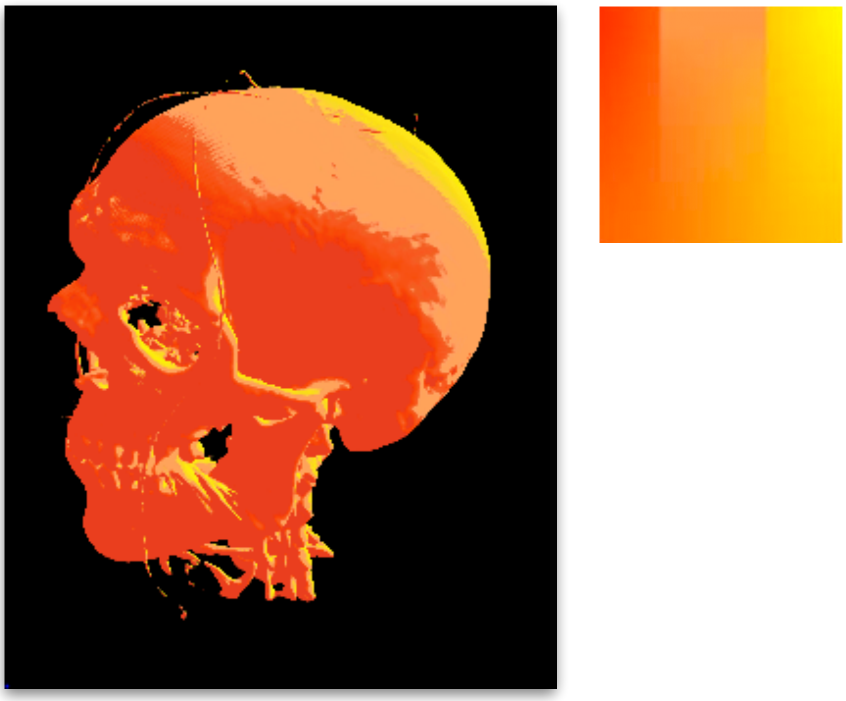


Figure 16: The X-Toon technique applied to the skull dataset using the texture on the right.

4.3.3 Textural Abstraction

Some of the goals of both non-photorealistic rendering and medical illustration are to remove extraneous details and to reduce the overall complexity of the volume. The process of abstraction shares these goals and many abstraction methods are already used within NPR as a way of emphasising shape and form over detail. Previous research conducted by Redmond et al. [RD091] further backs up the use of abstraction in NPR as their experiments found that stylised abstraction can improve the visual search time of users or in other words improve the user's perception of the rendering. Other uses for abstraction range from taking user focus away from less important sections of the volume in order to focus on other more important sections as well as a way of clarifying the visual structure of the volume.

Saturation Variation

Varying the saturation of the rendering is an efficient technique for adjusting the purity of its colours. A highly saturated rendering would have vivid and pure colours whereas a less saturated rendering would have mundane greyish colours. It is this less saturated, mundane look that fades the colour of the rendering and is ideal for abstraction. The method is relatively simple in theory, the RGB colours have to be converted into HSV space where the saturation component is extracted and lowered before being placed back and converted to RGB space again. In this implementation, two functions were used to convert between colour spaces, the *rgb2hsv* method and the *hsv2rgb* method. The rgb2hsv method takes in an RGB colour as a parameter and returns the converted HSV colour. The saturation value is contained in the Y value of the colour and is extracted and divided by a certain factor before being used as a parameter for the hsv2rgb method. This method takes in an HSV colour as a parameter and returns the converted RGB colour. This RGB colour is now the final colour for the current sample and can be returned to complete the process. The algorithm used can be seen below along with an image of the results produced by this method in figure [17]:

```
HSV = RGB2HSV( RGB );
HSV.y /= User Defined Variance Value
RGB = HSV2RGB( HSV );
```

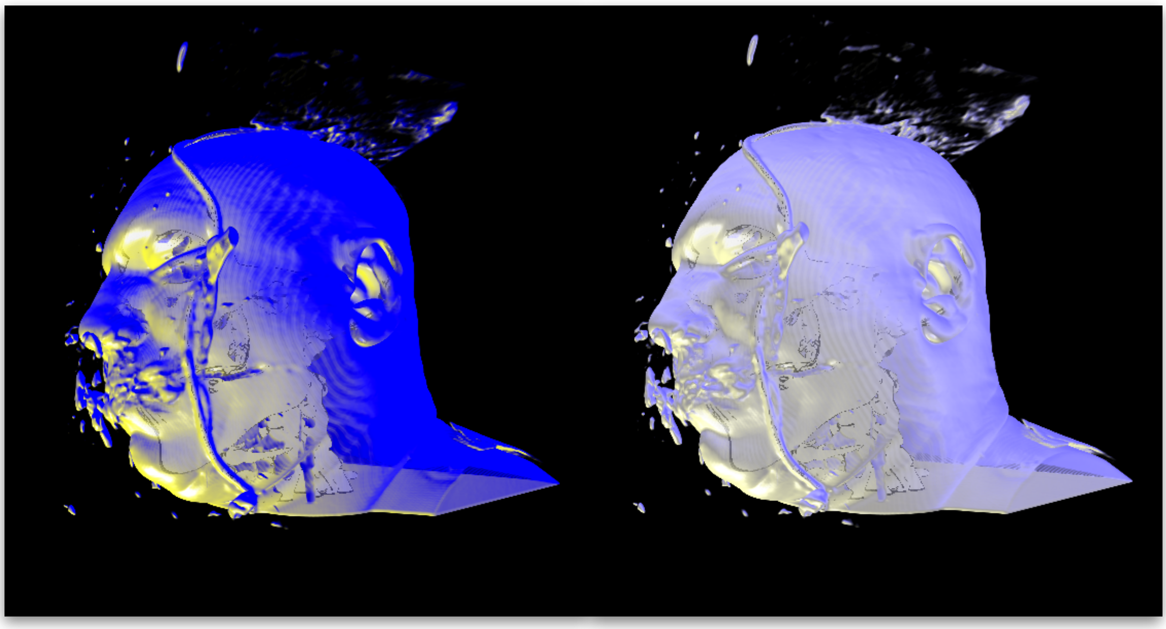


Figure 17: [Left] Two-Tone Shading. [Right] Two-Tone shading with Saturation Variation

4.4 2D to 3D Method Adaption

One major limitation of using a single-pass volume renderer is the loss of useful screen space (2D) techniques that have been effective in the rendering of medical data and medical illustration. This led the research to look into the possibility of implementing certain techniques, which were originally designed for screen space, in object space. The plan was to see what the results would be like if a 2D method that required the use of the surrounding pixels in both the X and Y directions of the image was adapted to become a 3D method that required the use of the surrounding voxels in the X, Y and Z directions of the rendering. This section of the implementation was very experimental and it was not expected to get any results that could be deemed computationally efficient and useful enough to be used in the actual implementation.

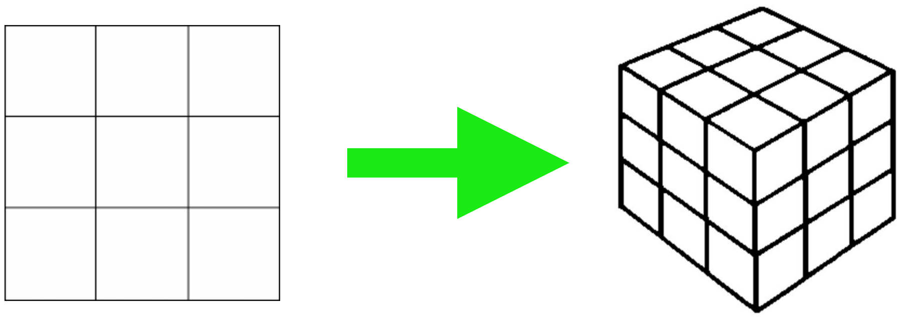


Figure 18: An example of a 2D grid vs a 3D grid.

3D Smooth

A simple box blur based filter was attempted as it is one of the most simple smoothing/blurring techniques possible within image space. A 2D box blur consists of adding up the values of all of the surrounding pixels and dividing by the number of pixels to get the average within the neighbourhood. The current sample is then assigned this average value. A 3D implementation of this would simply expand the neighbouring voxels along the z-axis in both the positive and negative directions resulting in a uniform grid. The implementation worked by iterating through the surrounding voxels while adding the colour value at each voxel to a variable value. After all of the iteration is complete this variable value is then divided by the number of voxels in the grid to get the average value that will be applied to the centre voxel. The basic formula for this implementation can be seen below:

Centre Voxel = Total Value of Voxels / Number of Voxels

3D Edge Preserving Smoothing Filter (Kuwahara Based)

The Kuwahara filter is a widely used 2D edge preserving blurring filter within NPR and medical illustration which was proposed by Kuwahara et al. [KHEK76] in 1976. The goal of the filter was to smooth out the majority of the image while maintaining only the important edges. This was achieved by splitting up a grid of surrounding pixels within a certain radius into four adjacent overlapping grids with the central pixel present in each one. Each overlapping region is analysed before the value of the centre pixel is set to the mean of the overlapping region with the lowest variance. In theory, extending the Kuwahara filter from a 2D image space technique to a 3D object space technique would only take a few adjustments to the original design meaning a complete overhaul was not needed. It worked by expanding the overlapping grids along the z-axis in both the positive and negative direction leaving a perfectly uniform cube of voxels making up the current sample point and its surrounding voxels. The four overlapping regions in the original 2D Kuwahara method were now replaced with eight overlapping regions with the grid now split up into four sections in the x, y and z directions. The method worked by first iterating through the surrounding voxels and calculating the mean and the standard deviation for each region. The standard deviation values of each region were then analysed to find the lowest value. The mean of the region with the lowest standard deviation was then applied to the centre voxel.

3D Edge Preserving Smoothing Filter (Bilateral Based)

Another widely used image space edge preserving blurring filter is the Bilateral filter. In image space, the Bilateral filter works by replacing each pixel with the weighted average of its neighbours. This weight is normally based on some type of distribution and depend on both the Euclidean distance and the colour difference between pixels. Tomasi and Manduchi [TM98] were the first to coin this non-linear process as Bilateral filtering. In this implementation, the method was based on an existing shader created by [shadertoy.com, 2013] which contained a 2D Bilateral filter. This filter was modified so that it could be used as an object space smoothing technique. The method worked by first calculating a kernel that would be used determine the weighted average of the nearby voxels based on distance. Two functions were used to calculate the probability density function (pdf) of the standard normal distribution with one function being used for difference in distance while the other was used for the difference in colour between voxels. Separate sigma values are used to specify the standard deviation of each distribution. The pdf is a function used to determine the probability of a distance or colour difference value falling within a certain range so that values that are closer to zero, meaning that there is little or no difference between voxels will have a higher weighting. The formula for these pdf functions can be seen below where the value *V* corresponds to the colour or distance value:

*0.39894 * exp(-0.5*dot(V,V)/(SIGMA*SIGMA))/SIGMA*

The method then iterates through the surrounding voxels in the dataset calculating the colour and distance factors at each voxel. The distance factor is calculated based on the kernel values for the location that the current voxel is from the centre voxel. The colour factor is calculated based on a combination of the pdf value of the difference in colour between the current voxel and the centre voxel and a normalised pdf using the same sigma value. The two factors are then combined and used to find the overall weighted average that will be used for the centre voxel.

Chapter 5 Results

This chapter will look at the results of the implementation is capable of producing. This will include a detailed look at the illustrations that can be produced due to the application having a high level of user flexibility and interactivity, how the rendering looks when styles are applied to both levels of the volume and what the results are like when different datasets are used. This chapter will also look at how the application can implement a focus and context like rendering approach and will also showcase the results that the individual style can produce.

5.1 Flexibility and Interactivity

Two of the key aspects of the application that had been focused on and identified at an early stage in the research process were the need for flexibility and the need for the user to be able to interact with the rendering. Having flexibility means that the user does not always have to look at the same NPR styles or the same version of NPR styles. This can be achieved by giving them the ability switch between multiple styles while also being able to modify some visualisation parameters of these styles. Having interactivity means that the user is not confined to just a single view of the rendering and can themselves interact with the model and change their view as they wish.

Flexibility was provided by allowing the user to choose from different styles that have been explained and shown previously but further flexibility was provided by allowing the user to modify visualisation parameters for these styles. For example, for the two-tone shading style the user had the option of choosing which two colours they wanted the style to mix between, for the hatching style the user was able to modify the threshold value using the slider to determine at what intensity of light a hatch would be drawn and for the X-Toon style the user could select which 2D texture would be used. Another example of this flexibility can be seen with the toon style where, very much like the hatching style, the user can use the slider to modify the threshold value that will determine at which intensity of light the different levels of tone will be shaded. This can be seen in figure [19] below showing three images of the skull dataset with three different levels of the threshold value used. It can be seen that by having this extra flexibility, a variety of illustrations can be produced from just a single style.

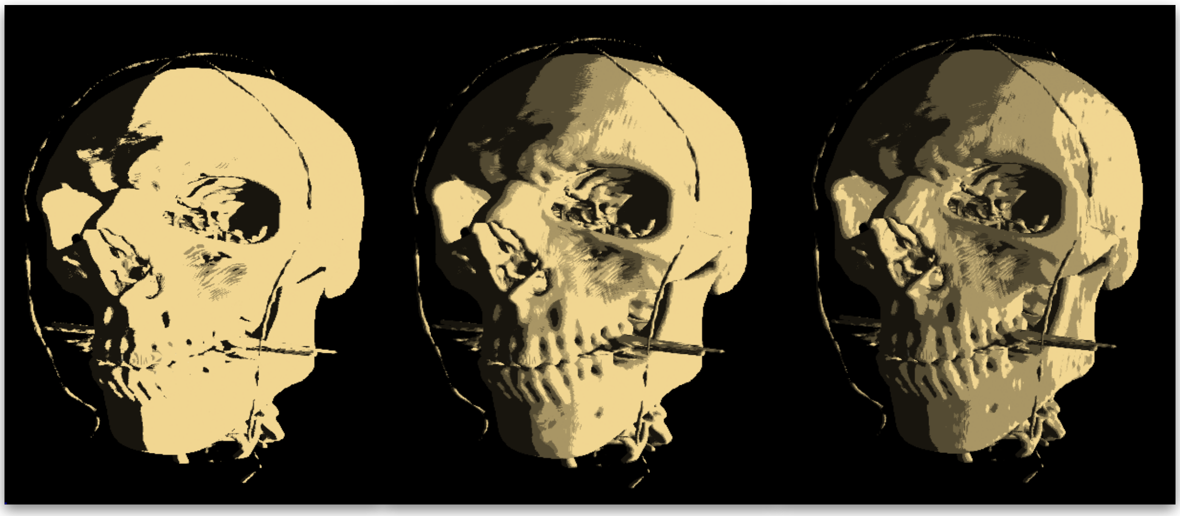


Figure 19: Toon Shading using three different threshold values.

The transfer function also adds a completely different level of flexibility to the application. By allowing the user to change the opacity of different levels of the volume, they can essentially control what they can see and what they can't see on the screen. The user is able to adjust opacity sliders for the skin, bone and other sections of the data where the other section refers to teeth in the skull dataset or bone marrow in other datasets. This results in the ability to produce a wider variety of illustrations from each dataset as the user is able to view the dataset from a number of perspectives that weren't available previously. An example of these new perspectives can be seen in figure [20] below where the opacity values have been modified to show different views of the same dataset within a single rendering.

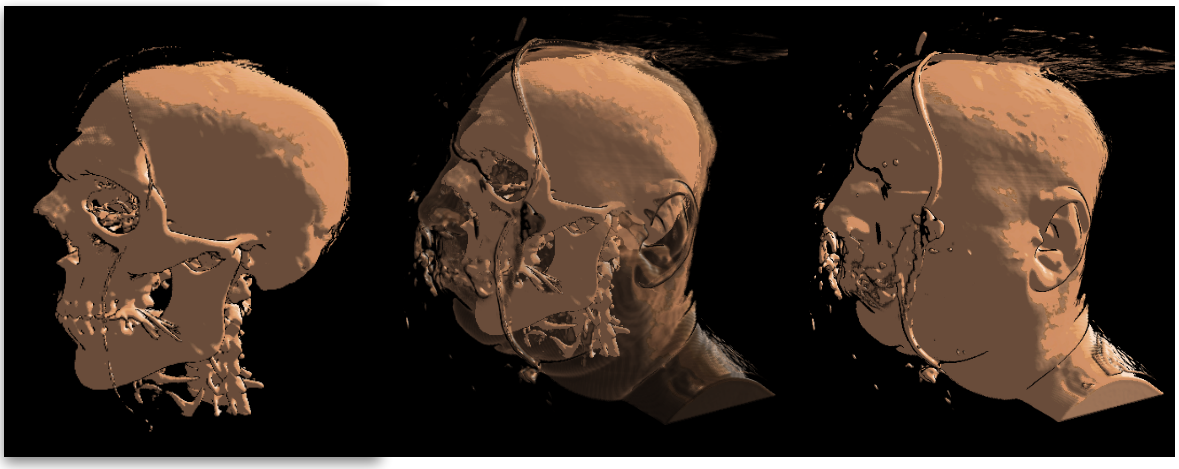


Figure 20: The X-Toon shading effect shown using 3 different skin opacities.

Interactivity was provided by giving the user the ability to alter their view of the model in any way that they wished. For an application to be interactive it needs to be able to run in real-time meaning that the user needs to be able to not only change the view of the rendering but that the styles and visualisation parameters can be changed in real-time and any of the styles that are used are computationally efficient enough to run in real-time. All of the styles used in this application were efficient enough to be compatible with a real-time environment and all of the style parameters were capable of being modified and have their changes shown in real-time. This allowed the user to rotate and zoom in and out of the rendering as they wished. Some of the possible views that can be achieved are shown below in figure [21] that again shows three images of the skull dataset from three different viewing angles that were all achieved within the same rendering. This shows the interactive nature of the application and that allowing the user to alter their view of the rendering can allow them to produce illustrations from whatever angle they wish.

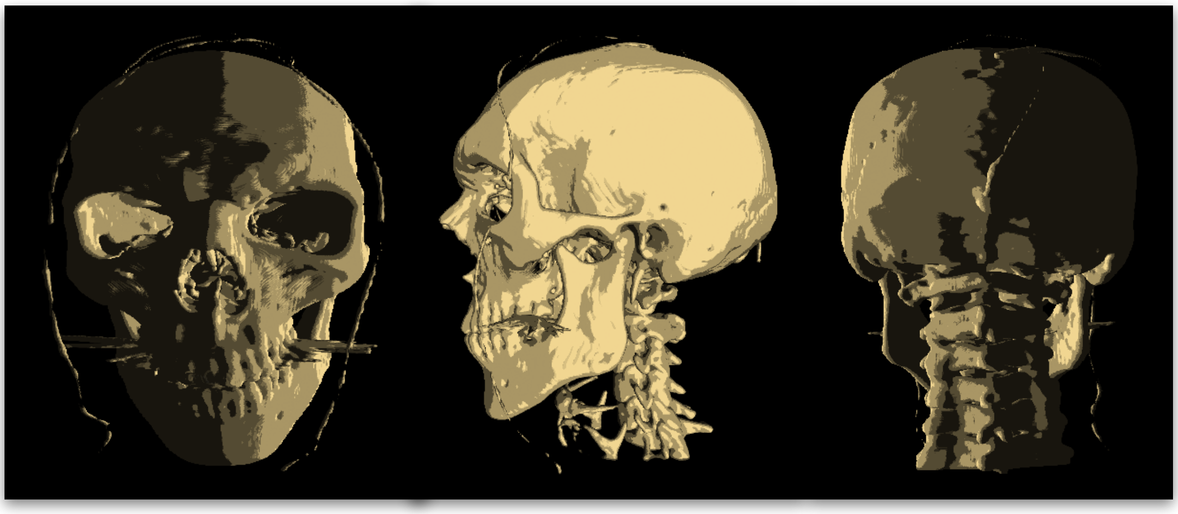


Figure 21: The Toon shading effect from three different perspectives

Another element of flexibility that was achieved in the application was the ability to load and render different CT scan datasets. This was an important aspect as it meant that the user would not be confined to just rendering the same model over and over again. This solved one of the main issues with artistic illustrations whereby the model or person that the illustration was based on could not be changed unless a completely new illustration was created, this application only required a new dataset to be loaded for this to work. Unfortunately, this ability was not granted in real-time but simply required the user to place the new dataset in the enclosing folder for the volume renderer to load and render it. This resulted in a much greater variety of illustrations that could be produced by the application as well as providing the ability for datasets that were specific to a certain person or a certain medical condition to be made into easy to understand illustrations. Examples of some of the illustrations that can be produced using different datasets can be seen in figure [22] below.

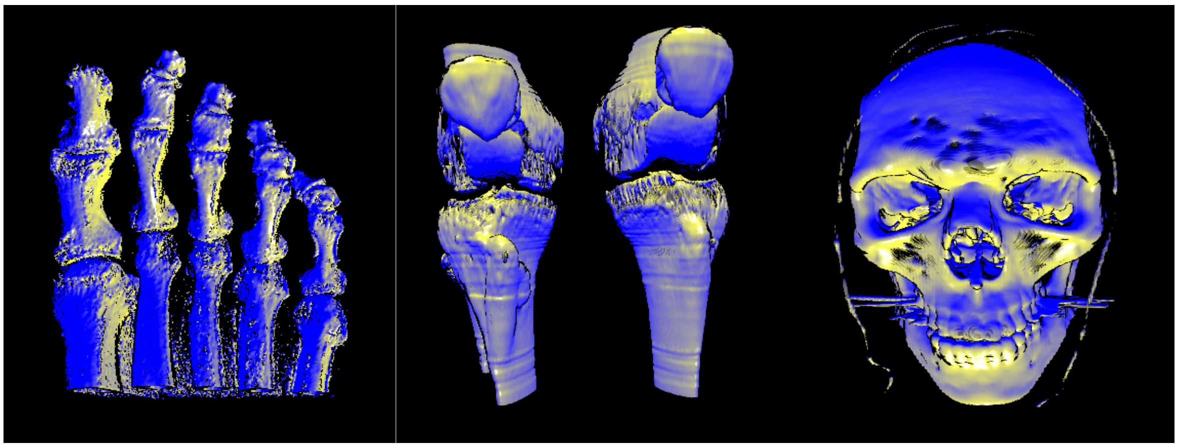


Figure 22: The Two-Tone shading effect applied to [left] foot, [middle] knee and [right] head datasets.

5.2 Two Levels of Volume

Through the use of a two-level volume rendering strategy, the application is better able to aid the user in their perception of the data. As mentioned previously, Corcoran [Cor10] showed that having two levels of the volume allows the separation of the most relevant part of the dataset from the peripheral extraneous details which helps the user to comprehend the dataset. This was achieved in this application by using the intensity at each voxel in the rendering to determine if the voxel was part of the skin or part of the bone. A style was then applied to each voxel depending on which section of the volume the voxel belonged to. As user perception and understanding of the image is an important aspect of medical illustration, this approach is very well suited to this application. This also adds another element of flexibility to the implementation allowing the user to apply multiple styles to each section of the dataset as they need. This can be seen in figure [23] below which shows how different styles can be applied to different levels of the volume:

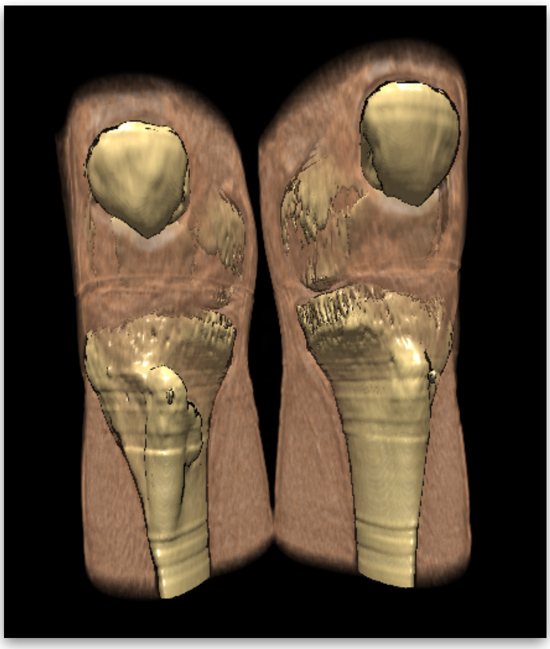


Figure 23: The Phong shading effect applied to the bone while the X-Toon effect is applied to the skin.

5.3 Focus and Context

One approach that was looked at during the research was the concept of a focus and context rendering. This concept is based on the two-level volume rendering approach mentioned earlier but with an emphasis on the effects that are applied to each level of the dataset. Focus and context rendering is popular within traditional illustrations but perhaps more so in medical illustration due to its ability to point out features of interests within a dataset. Focus and context rendering was made possible in this implementation by giving the user the ability to use textural abstraction techniques such as saturation variation on the rendering to suppress extraneous detail while also being able to apply edge detection and stylised shading techniques to enhance certain parts of the dataset. Using the two-level volume approach, one section of the dataset had edge detection applied to it to make it the focus of the rendering while the other section of the dataset had saturation variation applied to it and its opacity value lowered via the transfer function so that it was only partly visible to provide context to the scene. In figure [24] below an example of this focus and context approach produced by the application can be seen.

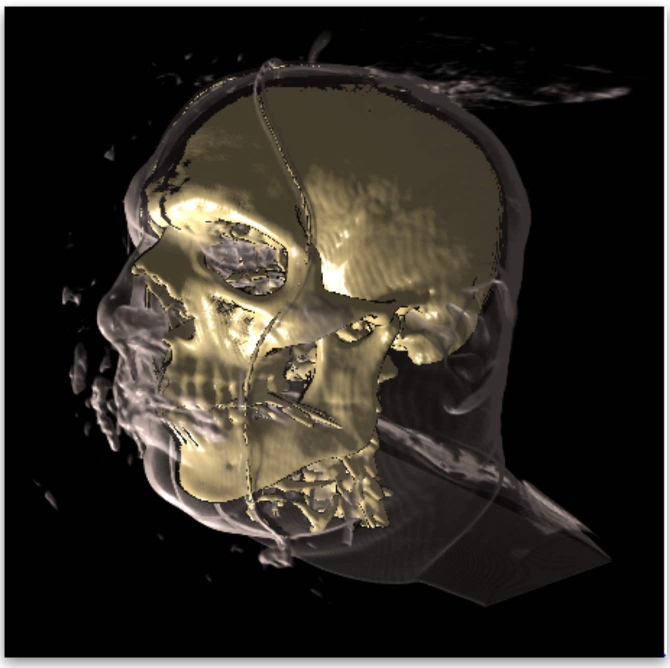


Figure 24: The focus [skull] with a Phong Shading effect while the context [skin] has saturation variation.

5.4 Different NPR Styles

The user has the ability to select NPR styles from three different categories namely edges, stylised shading and textural abstraction. Edges are often used in illustrations to add detail and convey information about shape. In this implementation, contours were the edges that were extracted from the surface of the rendering. The contours algorithm was very simple but still was effective in adding detail and shape to the scene. Due to its ability to add detail and information to the rendering, contours also proved effective when applied to the focus section of the data making it stand out from the rest of the dataset. Stylised shading has been proven to be an effective approach for drawing user attention to the important sections of the dataset.

In this implementation, a number of stylised shading techniques were available for the user to choose from. The Phong illumination model was used to add lighting to the rendering and resulted in an effective option for the user to add details about the shape and curvature of the model. The Toon shading style was implemented because of its ability to provide a simplified version of the dataset and proved to be effective and simulating lighting within the scene and adding detail to the rendering. The Two-Tone style was used because of its ability to allow the user to implement a Gooch shading model and it proved to be very effective at providing depth cues as well as conveying shape in the dataset. The Hatching style was chosen because of its ability to convey shape and lighting within the scene through the use of stokes and lines and in this implementation it was effective at both of these aspects once the user chose the correct threshold level. A dataset using the Hatching style also appeared visually more comprehensible when contours were applied in the same rendering. The X-Toon style was used because its ability to provide depth cues using a number of different textures and provided a visually appealing method which had a variety of tones within each texture.

Textural abstraction is widely used in medical illustration and is an important aspect in removing detail from areas of lesser importance in the rendering while also holding onto shape and structure to provide context. Saturation variation was a form of textural abstraction used within this implementation and provided the ability to vary the saturation of the model in order to provide abstraction in the rendering. Saturation variation was effective in fading the colour in the dataset to apply a more mundane look to a section of the dataset and take the focus away from this area. Numerous examples of all of these styles can be seen in the appendix below.

5.5 3D Filters

The decision to use single-pass over multi-pass rendering led to the investigation of the adaptation of 2D image space processing techniques to 3D object space implementations. This adaptation was achieved by using techniques that previously used 2D filters which had an X and a Y dimension and transforming them into 3D filters by adding a third Z dimension. The techniques that were attempted to be adapted were the Kuwahara Filter, Bilateral Filter and a simple box blur. The results of these adaptations were somewhat expected with these new methods being computationally very expensive for a real-time rendering approach. The visual results were interesting as can be seen in figure [25] below showing a comparison between the results of each method and the original style of the model. The Kuwahara based 3D filter resulted in a very yellowish colour being applied to the rendering, this was because of the skin, which had its opacity lowered and as a result was not visible was pure yellow in colour whereas the bone was purposely given a variety of colours so that a smoothing effect could be seen. As Kuwahara works by setting each pixel to the mean of the surrounding area with the lowest variance which in most cases around the outer parts of the skull was yellow. Both the box blur and Bilateral based filters resulted in a much more smoothing like effect with each resulting in an image with a slight yellow tint from the surrounding skin mixed with the varied colours of the skull. Of all of the techniques, the box blur based filter produced the best results together with the most efficient implementation. Unfortunately, these techniques were too computationally expensive to be used within a single real-time rendering with any other techniques.

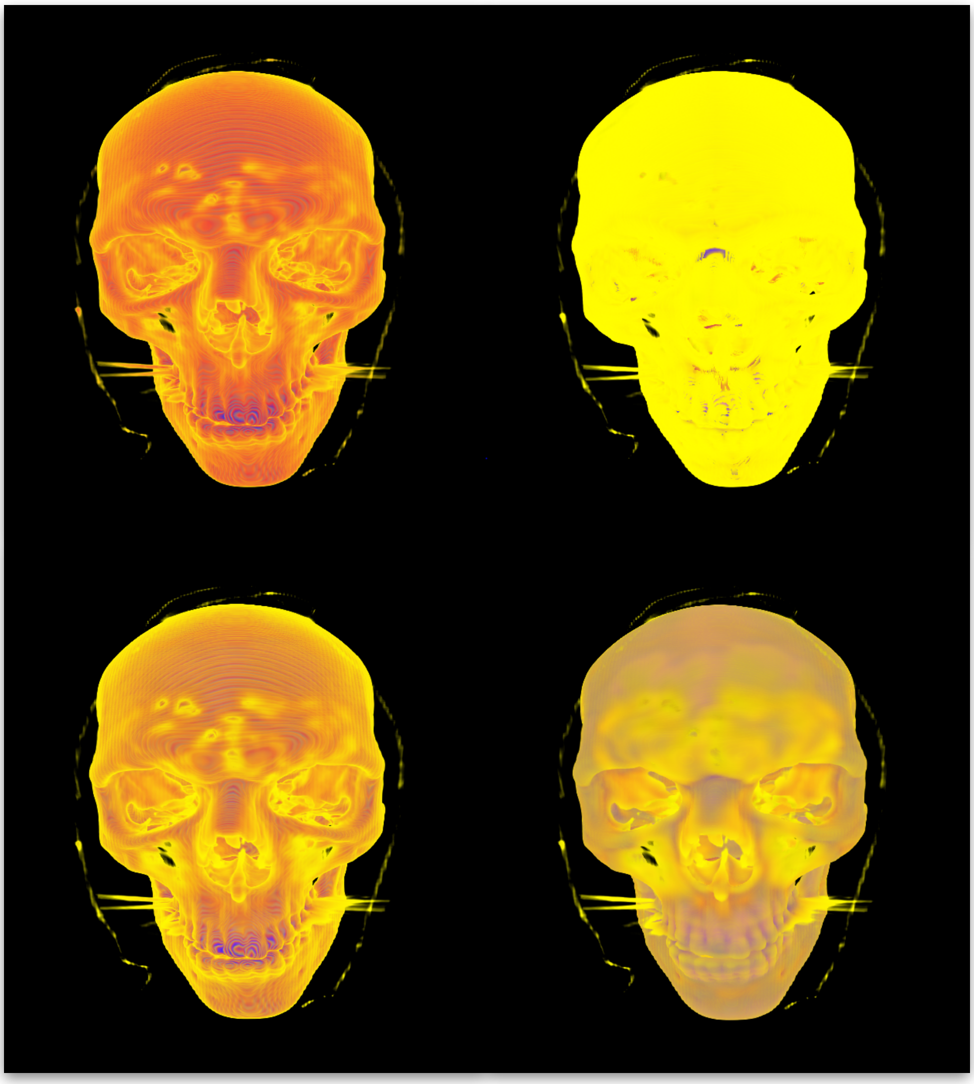


Figure 25: Original Image[Top Left], Kuwahara Based[Top Right], Bilateral Based[Bottom Right] and box blur based[Bottom Right].

Chapter 6 Conclusion

This final chapter will contain a reflection on a number of different aspects of both the research and the application implementation. Certain limitations that remain present in the implementation will be discussed along with some suggestions as to how these limitations can possibly be solved. Some possible future work that could be done as an extension of this research will also be proposed along with some final thoughts on the overall project itself.

Like any volume rendering design, this implementation does have its limitations. As the decision was made to implement a single-pass rendering of the volume, in doing so many more possible NPR techniques were eliminated that could have been used. This meant that no screen space techniques could be used that required a render to texture approach where the current texture of the volume was mapped to a 2D texture in one pass before the screen space methods were applied to the 2D texture. As a result, all of the techniques considered during both the research and the implementation were object space or 3D approaches. Of course, all of this could be solved by implementing a multi-pass rendering system which would open up a much larger variety of NPR techniques that could be used here.

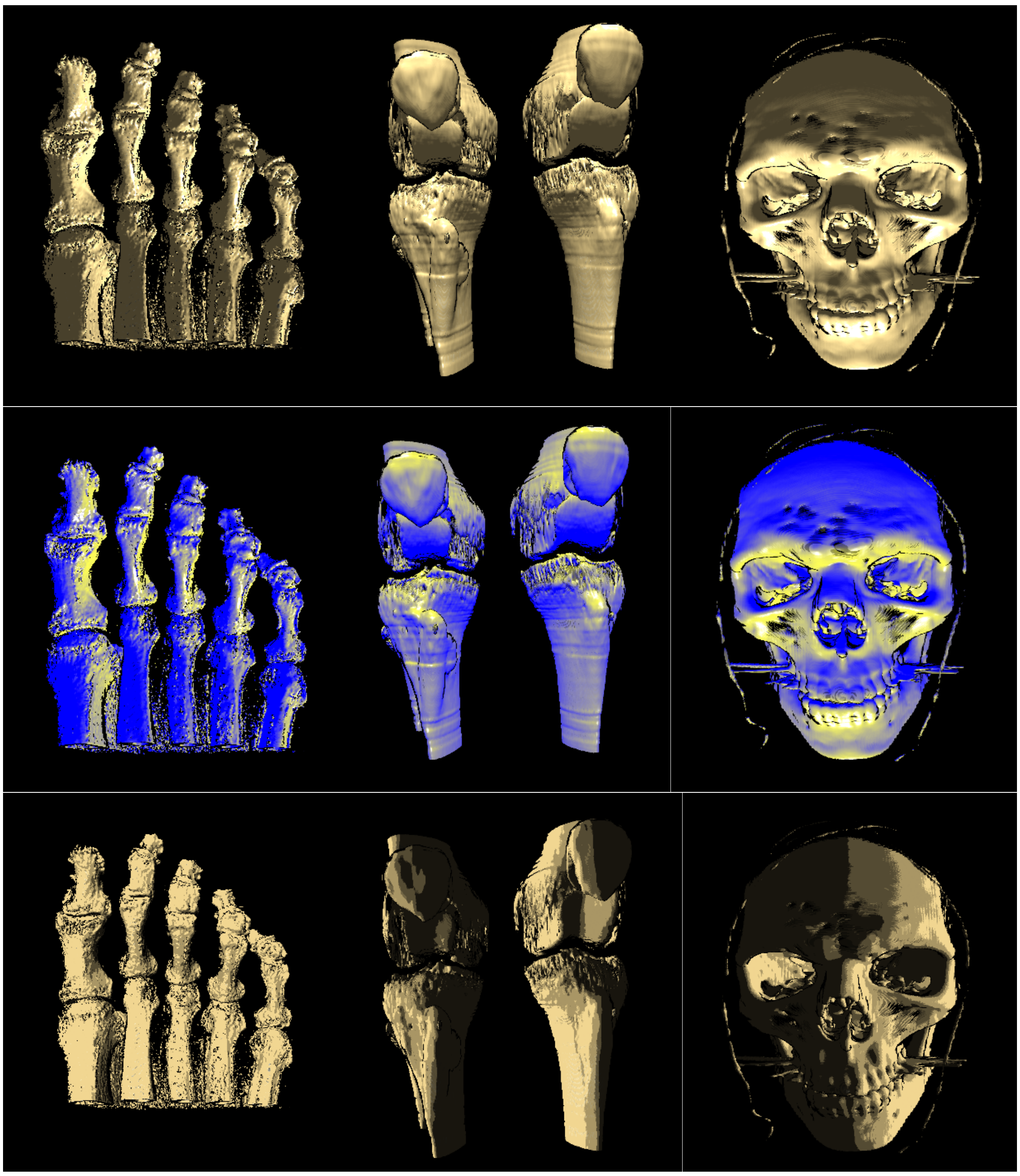
Other limitations arose through the decision to use raycasting as the approach to rendering the volume. raycasting is a very computationally expensive approach and as a result, may not render the volume as fast as other direct volume techniques would. There are a number of optimisation techniques that can be used with raycasting such as Hierarchical Spatial Estimation (HSE) and Adaptive termination. HSE is a technique used to make the raycasting algorithm more efficient in how it samples points along the ray. Adaptive termination works by identifying the last sample along a ray that significantly changes the colour of the ray so that the ray can terminate early if it will not change colour significantly along the rest of the ray. Another issue that is often seen with raycasting algorithms is the generation of artefacts due to a low sampling rate. These artefacts are often very subtle and difficult to spot but in the area of illustration, every detail matters. There are some techniques that can help with artefacts produced by raycasting such as stochastic jittering and virtual sampling. Stochastic jittering uses random sampling while virtual sampling interpolates between sampling points in order to combat artefacts that are caused by low sampling rates.

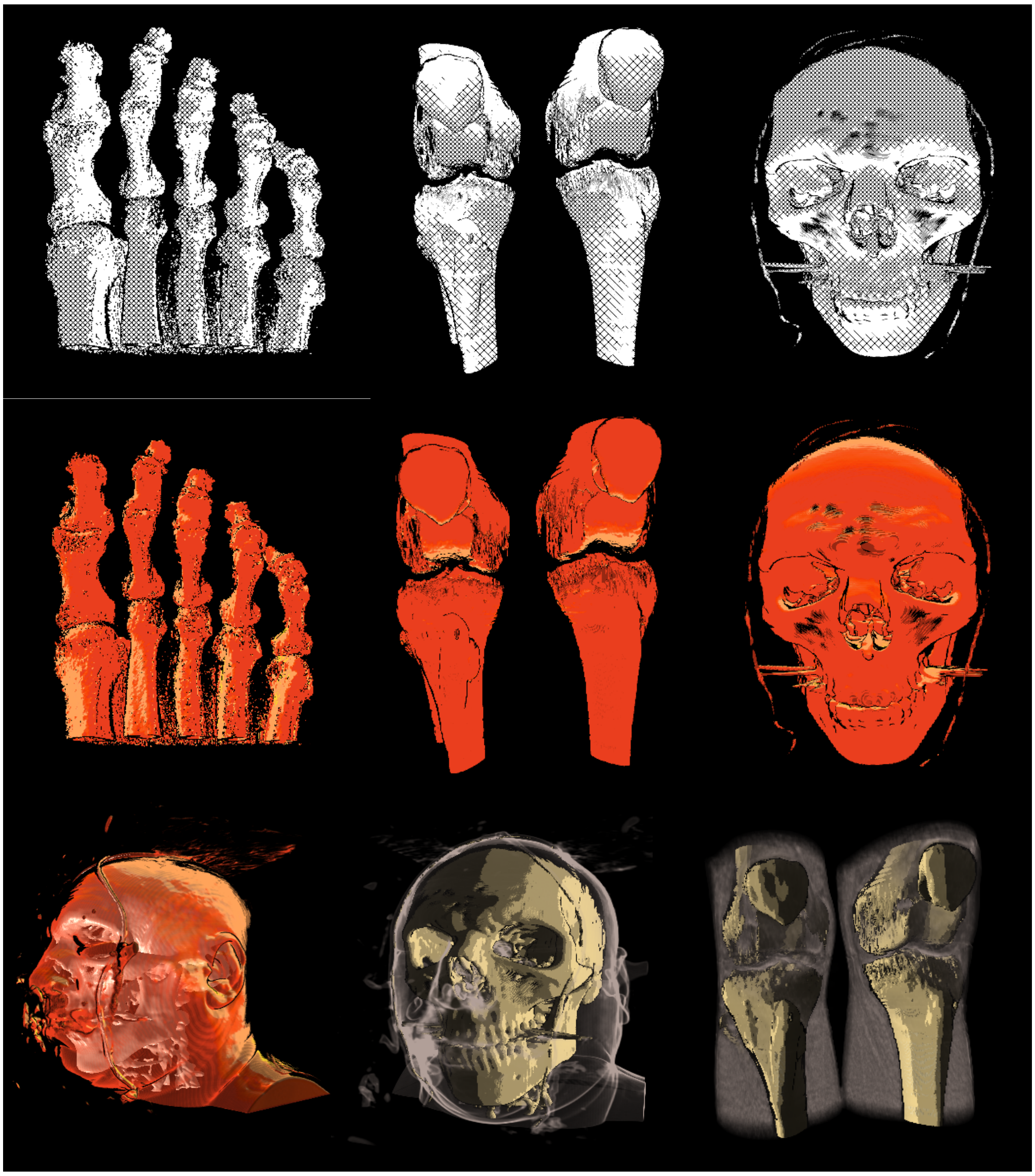
Some possible future work that could be explored to extend this research would be the implementation of a multi-pass rendering system so that 2D image space techniques could be added to the pool of NPR styles from which the user can choose from. This would allow especially for much more effective edge detection techniques to be used that would help to greater enhance the detail shown within each illustration. Work could also be done to increase the range of stylised shading styles that the user has to choose from. As there are thousands of different NPR styles there are sure to be many more that are efficient enough to be used within a real-time rendering scenario combined with multiple other techniques. Other forms of future work could look at implementing user tests to study different aspects of a user's interaction with the application. These tests could include timing how long it takes for a user to produce an illustration which they are satisfied with, monitoring which styles are used most of all by a user or even to look at what threshold range is the most popular amongst a set of users for each style.

The initial goal of the research was to identify the techniques that were required and that could be used to create medical illustrations and then use these techniques to create a proof of concept highly flexible automated anatomy illustration application. Throughout the research, these techniques were identified as non-photorealistic rendering styles that made up three categories namely edges, stylised shading and textural abstraction. Through the use of these techniques, the user was able to implement a focus and context rendering approach that has been widely used within medical illustration. The application itself was driven by its flexible design and interactivity that gave the user a high level of control over the rendering. Through the use of sliders, drop-down menus and radio buttons the user could modify various visualisation parameters within the rendering all of which added to the high level of flexibility. The main goals of the research were achieved even with a few setbacks and limitations that were encountered along the way and ultimately an efficient means of producing anatomy illustrations was established.

Appendices

Appendix A - Images





Appendix 1: The first five rows contain examples of the different styles applied to each dataset in the following order: Phong, Two-Tone, Toon, Hatching, X-Toon. The last row contains, from left-to-right, Phong shading on the bone with X-Toon on the skin [skull], Toon shading on the bone with Saturation Variation on the skin [skull], Toon on the bone with Saturation Variation on the skin [knee].

Bibliography

[LC87] Lorensen, W. and Cline, H. (1987). Marching cubes: A high resolution 3D surface construction algorithm. SIGGRAPH Comput. Graph., 21(4):163–169.

[MW97] Müller, H. and Wehle, M. (1997). Visualization of implicit surfaces using adaptive tetrahedrizations. In Scientic Visualization Conference, page 243, Los Alamitos, CA, USA. IEEE Computer Society.

[SCK95] Shu, R., Chen, Z. and Kankanhalli, M.S. (1995). Adaptive marching cubes. The Visual Computer, 11(4), pp. 202-217.

[WG92] Wilhelms, J. and Van Gelder, A. (1992) Octrees for Faster Isosurface Generation. ACM Transactions on Graphics, 11(3), pp. 201-227.

[DSSCNS09] Dietrich, C. A., Scheidegger, C. E., Schreiner, J., Comba, J.L.D., Nedel, L. P. and Silva, C. T. (2009). Edge transformations for improving mesh quality of mar- 171 Bibliography ching cubes. IEEE Transactions on Visualization and Computer Graphics, 15(1):150–159.

[Wes90] Westover, L. (1990). Footprint evaluation for volume rendering. SIGGRAPH Comput. Graph., 24(4):367–376. ACM ID: 97919.

[Rot82] Roth, S.D. (1982). raycasting for Modeling Solids. Graphical Models /graphical Models and Image Processing /computer Vision, Graphics, and Image Processing - CVGIP. 18. 109-144. 10.1016/0146-664X(82)90169-1.

[SSKE05] Stegmaier, S., Strengert, M., Klein, T. and Ertl, T. (2005). "A simple and flexible volume rendering framework for graphics-hardware-based raycasting" *Fourth International Workshop on Volume Graphics*, pp. 187-241.

[RD091] Redmond, N. and Dingliana. J. (2009). Investigating the effect of real-time stylisation techniques on user task performance. In Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization (APGV '09). ACM, New York, NY, USA, 121-124.

[Can86] Canny, J. (1986). "A Computational Approach to Edge Detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679-698.

[MH80] Marr, D., Hildreth, E. (1980). *Theory of Edge Detection* in Proceedings of the Royal Society of London. Series B, Biological Sciences, Vol. 207, No. 1167. (Feb. 29, 1980), pp. 187-217.

[HZ00] Hertzmann, A. and Zorin, D. (2000). *Illustrating smooth surfaces*. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH '00). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 517-526.

[HSSBG05] Hadwiger, M. , Sigg, C. , Scharsach, H. , Bühler, K. and Gross, M. (2005), Real−Time Ray−Casting and Advanced Shading of Discrete Isosurfaces. Computer Graphics Forum, 24: 303-312.

[BKRFD05] Burns, M., Klawe, J., Rusinkiewicz, S., Finkelstein, A., & DeCarlo, D. (2005). Line drawings from volume data. *ACM Trans. Graph., 24*, 512-518.

[GGSC98] Gooch, A., Gooch, B., Shirley, P. and Cohen, E. (1998). "A non-photorealistic lighting model for automatic technical illustration". In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques (SIGGRAPH '98)*. ACM, New York, NY, USA, 447-452.

[Dec96] Decaudin, P. (1996) : Cartoon-looking rendering of 3D scenes. Technical report 2916, INRIA.

[LMHB00] Lake, A., Marshall, C., Harris, M., and Blackstein, M. (2000). Stylized rendering techniques for scalable real-time 3D animation. In *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*, NPAR '00, page 13–20, New York, NY, USA, 2000. ACM.

[BTM06] Barla, P., Thollot, J., Markosian, L. (2006). X-Toon: An extended toon shader. Douglas DeCarlo and Lee Markosian. *International Symposium on Non-Photorealistic Animation and Rendering (NPAR'06)*, Jun 2006

[SMGG01] Sloan, P.-P., Martin, W., Gooch, A., and Gooch, B. (2001). The lit sphere: A model for capturing NPR shading from art. In Graphics Interface 2001, 143–150.

[PHWF01] Praun E., Hoppe H., Webb M., Finkelstein A. (2001). : Real-time hatching. In Proceedings of ACM SIGGRAPH 2001 (2001), pp. 581–586.

[BG07] Bruckner, S and Gröller, E. (2007). Style Transfer Functions for Illustrative Volume Rendering. Computer Graphics Forum. 26. 715 - 724. 10.1111/j. 1467-8659.2007.01095.x.

[KHEK76] Kuwahara, M., Hachimura, K., Eiho, S. and Kinoshita, M. (1976). Processing of RI- angiocardiographic images. In Kendall Preston and Morio Onoe, editors, *Digital Processing of Biomedical Images*, pages 187–203. Plenum Press, New York, 1976.

[PPC07] Papari G., Petkov N., Campisi P. (2007): Artistic edge and corner enhancing smoothing. IEEE Transactions on Image Processing 16, 10 (2007), 2449–2462.

[KKD09] Kyprianidis, J. E., Kang, H., and Döllner, J. (2009). Image and Video Abstraction by Anisotropic Kuwahara Filtering. Computer Graphics Forum 28, 7, 1955–1963.

[TM98] Tomasi C., Manduchi R. (1998) : Bilateral filtering for gray and colour images. In Proceedings International Conference on Computer Vision (ICCV) (1998), pp. 839–846.

[CM02] Comaniciu D., Meer P. (2002) : Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 5 (2002), 603–619.

[RE01] Rheingans P., Ebert D. S. (2001) : Volume illustration: Nonphotorealistic rendering of volume models. IEEE Transactions on Visualization and Computer Graphics 7, 3 (2001), 253–264.

[NSW02] Nagy, Z., Schneider, J. and Westermann, R. (2002). Interactive volume illustration. In Proc. of Vision, Modeling, and Visualization, pp. 497–504, 2002.

[PB07] Preim, B. and Bartz, D. (2007). *Visualization in Medicine: Theory, Al- gorithms, and Applications (e Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[HMIG01] Hauser, H., Mroz, L., Italo Bischi, G. and Groller, M. E. (2001) "Two-level volume rendering," in *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 3, pp. 242-252, Jul-Sep 2001.

[MRMH09] Meyer-Spradow, J., Ropinski, T., Mensmann, J. and Hinrichs, K. (2009). "Voreen: A Rapid-Prototyping Environment for Ray-Casting-Based Volume Visualizations," in *IEEE Computer Graphics and Applications*, vol. 29, no. 6, pp. 6-13, Nov.-Dec. 2009.

[DFRS03] DeCarlo, D., Finkelstein, A., Rusinkiewicz, S. and Santella, A. (2003) Suggestive contours for conveying shape. *ACM Trans. Graph.*, 22(3):848–855, 2003.

[LB00] Langer, M.S. and Bültho, H.H. (2000). Depth discrimination from shading under di use lighting. *Perception*, 29(6):649 – 660, 2000.

[CF07] Caniard, F. and Fleming, R.W. (2007). Distortion in 3D shape estimation with changes in illumination. In *Proceedings of the 4th symposium on Applied perception in graphics and visualization*, APGV '07, page 99–105, New York, NY, USA, 2007. ACM.

[Pho75] Phong, B.T. (1975). Illumination for computer generated pictures. Commun. ACM 18, 6 (June 1975), 311-317.

[RD092] Redmond, N. and Dingliana, J. (2009). Influencing user attention using real-time stylised rendering. In *TPCG 2009: Proceedings of eory and Practice of Computer Graphics Conference*, Cardi , UK, 2009. Eurographics UK Chapter.

[Cor10] Corcoran, A., Redmond, N. and Dingliana, J. (2010). Perceptual enhancement of two-level volume rendering, Computers & Graphics

[shadertoy.com, 2013] Shadertoy.com. (2013). Shadertoy. [online] Available at: https://www.shadertoy.com/view/4dfGDH [Accessed 24 May 2018].