Enhancing Real-Time Focus and Context Direct Volume Rendering

Andrew Corcoran

16th November 2012

A thesis submitted to the University of Dublin, Trinity College in candidacy for the degree of Doctor of Philosophy in Computer Science.

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work. I agree that Trinity College Library may lend or copy this thesis upon request.

Andrew Corcoran
16th November 2012

Acknowledgements

Thanks to my supervisor John Dingliana, all my friends, both in GV2 and in the 'real world' and of course to my family. None of this would have been possible without all your support.

This research is funded by the Science Foundation Ireland Research Frontiers Grant SFI08/RFP/CMS1076.

Abstract

Direct volume rendering is a challenging problem both due to the technical difficulty of displaying large volume datasets on limited hardware and due to the difficulty of rendering such complex information in a manner which is easy for a user to understand. Furthermore, the complexity of volume data and the continued increase in rendering demands makes real-time volume rendering an increasingly challenging problem. Focus and context volume visualisation attempts to solve both these problems by allowing the user to specify a focus area in the data which is displayed in high quality while the surrounding data is abstracted but sufficiently retained to provide context. This approach allows for a reduction in rendering complexity in the context region of the data, thus allowing for faster speeds. In addition, this two-level focus and context approach allows for increased understanding of the dataset.

In this thesis we explore new methods and styles for enhancing focus and context volume rendering and propose novel techniques which can reduce the rendering time of complex volume visualisations. Through the use of non-photorealistic line stylisations and advanced lighting techniques we develop a volume rendering framework which simultaneously enhances perception of the shape of the focus surface while also highlighting and enhancing internal structures in the context layer. We build upon this work by developing a system which can improve the rendering speed and image quality of focus and context rendering by taking advantage of temporal coherency to guide sampling rate optimisations.

We verify our results through the use of numerous user experiments which show that our techniques increase user understanding of shape perception and increase the amount of visual information in the rendering. Through analysis of these results and careful combination of our techniques we have developed a focus and context rendering framework which improves rendering speed, increases shape perception of the focus object, improves the ability of users to perceive structures in the context layer and provides aesthetically pleasing results.

Related Publications:

1. **Real-Time Illumination for Two-Level Volume Rendering**:
   Andrew Corcoran and John Dingliana;
   International Symposium on Visual Computing 2012, LNCS, 7431:544–555, 2012.

2. **Image Space Adaptive Volume Rendering**:
   Andrew Corcoran and John Dingliana;
   Visualization and Data Analysis 2012, Proceedings of the SPIE, 8294(1):8294M, 2012.

3. **Time-Critical Ray-Cast Direct Volume Rendering**:
   Andrew Corcoran and John Dingliana;
   Technical Report TCD-CS-2011-13, Trinity College Dublin, September 2011.

4. **Perceptual Enhancement of Two-level Volume Rendering**:
   Andrew Corcoran, Niall Redmond and John Dingliana;
   Computers & Graphics, 34(4):388–397, 2010.

Related Posters:

1. **Perceptually Optimised Non-Photorealistic Rendering of Biological and Medical Volume Data**:
   Andrew Corcoran, Niall Redmond and John Dingliana;
   VCBM 2010: 2nd Eurographics Workshop on Visual Computing for Biology and Medicine. Poster Session, July 2010

2. **Line Drawing Enhancement of Volume Rendering**:
   Andrew Corcoran and John Dingliana;
   NPAR 2009: 7th International Symposium on Non-Photorealistic Animation and Rendering. Poster Session, August 2009

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Volume visualisation is an important technique with widespread use in a number of scientific and engineering fields. Popular application areas include medical imaging, scientific visualisation, and flow visualisation. In recent years, more advanced graphics hardware and new algorithms have facilitated highly optimised pipelines for fast volume rendering. However, congruent advances in the capabilities of acquisition, storage, and simulation technologies have also led to a demand for visualising increasingly complex data.

In addition to the increased performance demands, successful interactive rendering of complex datasets is problematic due to the visual complexity of the data. Rendered images of complex volume data tend to have excessive detail, which is often multi-layered and overlapping. This causes significant demands on the human visual system to process what is often a perceptually challenging type of graphical output. In interactive visualisation, this problem is further exacerbated as the view of the data is continuously changing and there is even less time per frame for a user to assimilate the information. Thus, there is a clear need for strategies to address perceptual issues in interactive volume visualisations.

Focus and context rendering is a common technique used in a variety of traditional illustrations and is highly prevalent in medical illustration in order to emphasise key anatomical objects. The central idea of this technique is to emphasise regions of interest (focus) while sufficiently maintaining the remaining information in the image as it can be beneficial for orientation (context). This technique was originally developed as a method to simplify complex data, thus allowing it to be displayed in constrained environments. However, the human visual system has been shown to perceive both local detail and global context, and therefore the use of focus and context techniques are no longer indicated purely for data simplification [CCF97]. Focus and context rendering is a valuable imaging technique with scope for application to the enhancement of user understanding of complex datasets, and therefore there is a clear need for techniques which can improve this rendering paradigm.

1.2 Scope

The focus of this thesis is on methods for enhancing user understanding and computation speed of real-time, focus and context volume rendering. We investigate real-time techniques as we believe that allowing user interaction with the dataset is necessary in order to provide meaningful exploration of complex data, and this interaction can only be facilitated through real-time rendering speeds.

Furthermore, we investigate techniques which are applicable to consumer level devices rather than expensive dedicated visualisation hardware. As such, this constrains the amount of memory and processing power available, and therefore fast, low memory enhancement techniques are required. In addition, mobile and streaming devices are becoming more prevalent and techniques which allow for streamed data and which do not require large amounts of preprocessing are of importance.

1.3 Contributions

We have developed a *system for focus and context volume rendering* which allows for *multiple levels of enhancements and optimisations for rendering speed.* Our technique allows for *varied NPR stylisations* to be applied to both the focus and context layers of the rendering. In addition, we develop a technique for *real-time illumination* to be applied to focus and context rendering. Our illumination solution takes advantage of the focus and context rendering paradigm in order to *selectively enhance regions based on their importance*. Furthermore, we develop a number of techniques which allow for both *time-critical rendering and an improvement in the performance of two-level volume rendering*. All of these techniques have been combined together into one system which allows for flexible *stylistic enhancement and improvement of volume rendering through the use of both NPR and advanced illumination techniques.*

We conduct *numerous user studies* which provide a thorough investigation into how our techniques affect user perception of volume data. These studies show that our NPR techniques can *enhance user perception of shape* and provide interesting results regarding the efficiency of shadows for enhancing shape and depth perception for focus and context rendering. Through *statistical analysis* of these results we provide *recommendations for future practitioners* on the optimum methods for enhancing focus and context rendering for shape and depth perception and also provide *novel insights into functional versus aesthetic preference regarding illumination techniques*.

1.4 Summary of Chapters

The rest of this thesis is structured as follows:

Chapter 2 provides an overview of the background and related work in the field of focus and context volume rendering. It is split into three sections: volume rendering techniques, non-photorealistic volume rendering, and volume shading and illumination.

Chapter 3 describes the technique we have developed for real-time, non-photorealistic enhancement of focus and context rendering. In addition, we describe a comprehensive user experiment which investigates how our enhancement techniques affect shape perception.

Chapter 4 describes an approach we have developed which allows for an increase in the rendering speed of focus and context rendering by leveraging temporal coherence.

Chapter 5 describes a technique we have developed for real-time volume illumination for focus and context renderings. Furthermore, this chapter describes the details of a series of user experiments we performed which investigate how illumination affects user perception.

Chapter 6 summarises our contributions and discusses possible avenues of future work.

Chapter 2

Related Work

2.1 Introduction

VOLUME rendering is used to display a 2D image of a 3D dataset. The majority of datasets are discretely sampled along a 3D grid and contain scalar values usually acquired from medical imaging devices such as CT or MRI machines. The data then takes the form a 3D array of "voxels", a 3D extension of pixels. Volume rendering can be performed using two main techniques, either by extracting a number of surfaces from the data and rendering these surfaces to the screen, called isosurface rendering or by rendering the volume itself as a complete block of data with no intermediary structures, usually called direct volume rendering (DVR). A wide variety of techniques have been developed in order to improve volume rendering performance, enhance perception of the data, and increase the rendering speed.

- We first introduce the techniques used to display volume rendering data. Both isosurface rendering approaches and direct volume rendering techniques are discussed. We then go on to discuss methods which are used to increase the speed at which volume data can be rendered.

- Next we discuss non-photorealistic rendering (NPR) methods which are used to both simplify and enhance renderings, in addition to being used to mimic certain artistic techniques such as pen-and-ink drawings and painterly styles. We expand on these NPR techniques to discuss how they can be applied to volume rendering in order to enhance user perception.

- Finally, we discuss methods for lighting, shading, and shadowing of volume data. This includes techniques for precomputing lighting information as well as on the fly computation of lighting solutions.

2.2 Volume Rendering Techniques

Numerous approaches exist for volume visualisation which range from the very accurate to the very interactive [CKY01, Sal07, PTD05]. In recent times, the gap between these two extents has narrowed with technologies now enabling the interactive visualisation of very complex volume data [KKG09]. While some techniques first obtain a surface representation from the volume before applying standard 3D renderings [LLVT03, EKE01], DVR techniques operate directly on the volume data [CMH$^+$01]. This can be achieved through the use of 3D textures, generated from view aligned slices of volume data and optimised using GPU hardware for real-time performance [CN94, CCF94]. Other approaches employ techniques such as real-time volume ray casting [HSS$^+$05], facilitated through adaptive optimisations.

2.2.1 Splatting

Early attempts at direct volume rendering utilised techniques such as volume splatting [Wes89, Wes90] which solve the volume rendering problem with an object order approach. These techniques work by calculating the contribution of each individual voxel, determining their optical footprint and then combining the results into a final image. Volume splatting is no longer widely utilised as more advanced methods for volume visualisation have since been invented. However, there are still a few modern algorithms which utilise an object order approach to volume rendering [Bau11, SDGPG07, CRZP04].

2.2.2 Isosurface Extraction

Isosurface volume rendering techniques were introduced by Lorensen and Cline [LC87]. These methods work by extracting a polygonal representation of a specific isosurface from within the volume data. This polygonal mesh is then displayed using traditional rasterisation based rendering methods. The marching cubes implementation introduced by Lorensen and Cline was further developed by Müller and Wehle [MW97]. This marching tetrahedrons implementation is more complicated and has higher memory requirements but resolves some ambiguities inherent in the marching cubes implementation and also has a slightly better sampling resolution.

Additional refinements to the marching cubes algorithm have since been published. Lewiner et al. [LLVT03] ensure a topologically correct result, something which is not guaranteed by the original implementation. Raman and Wenger [RW08] propose extending the marching cubes lookup table in order to eliminate degenerate polygons. Other researchers have investigated extending the marching cubes algorithm to multiple dimensions [BWC04].

High quality methods for isosurface extraction have also been proposed. Schreiner et al. [SSS06] generate more polygons in areas of the volume where surface curvature is high, which results in evenly spaced polygons and smooth surfaces. Labelle and Shewchuk [LS07] propose a fast isosurfacing technique which increases mesh smoothness. Dietrich et al. [DSS+09] propose a high quality, extremely computationally efficient isosurface extraction algorithm which eliminates degenerate polygons and can easily be integrated with existing marching cubes acceleration techniques.

2.2.3 Texture Slicing

Shear-warp volume rendering [LL94] is a CPU specific volume rendering technique which traverses the volume data in a slice by slice fashion. This technique first transforms the volume into a sheared object space in order to allow for efficient sampling of the volume data which results in a fast but distorted rendering. This rendering can be undistorted by performing a 2D warp to form an undistorted final image. This shearing of the volume data allows for several optimisations which results in an extremely fast rendering algorithm.

With the development of programmable GPU's, researchers began to exploit hardware approaches in order to achieve significant increases in computational speed. A number of researchers independently implemented texture based volume rendering [CCF94, CN94, GK96] on the RealityEngine [Ake93] system. This technique divides the volume into slices and then uses the graphics hardware to render slice polygons textured with volume data. These polygons are then blended together to form the final image. This technique is closely related to shear-warp rendering. The shear-warp technique generates a final image and then warps it to correct distortion whereas texture based slicing warps each slice and then composites the undistorted slices. With the advent of consumer GPU's, texture slicing became the dominant method for GPU-based volume rendering.

Initial implementations of texture slicing on commercial GPU's was limited to 2D textures and object aligned slicing due to the lack of hardware support for 3D textures. This resulted in flickering when the slice axis changed, inconsistent sampling rates, and a lack of filtering in the z dimension. With the advent of hardware support for 3D textures, view-aligned slice based rendering became possible which provided superior image quality and removed many of the drawbacks inherent in a 2D texture based implementation.

2.2.4 Pre-Integrated Rendering

Pre-integrated volume rendering [EKE01, RKE00] is a technique which increases the image quality of volume rendering without increasing the number of slices or sampling rate.

Standard slice based volume rendering only samples the volume data at each slice polygon which can result in a large loss in image quality if the volume data changes dramatically between slices, for example, due to a high frequency transfer function [Sha49]. Pre-integrated rendering separates the sampling rate of the volume data from the sampling rate of the transfer function. The transfer function integral is pre-calculated for every possible data value and stored in a 2D texture. At each slice location, the intensity of the volume at the current slice and the intensity of the volume at the previous slice are used as a lookup into the texture, therefore, accounting for possible changes occurring between the two slices. Pre-integrated volume rendering has a high memory requirement but ensures that the maximum number of slices needed depends only on the size of the volume data and not on the frequency of the transfer function.

Unfortunately, pre-integrated rendering does not pre-integrate normal information which is generally a key piece of data required for accurate volume shading. The standard approach for normal calculation is to sample the normal at the centre of each pre-integration slab and perform lighting calculations using this data. Unfortunately, normal information can change dramatically within one sampling step and this technique can produce artifacts. Lum et al. [LWM04] propose interpolating the lighting values between the two slices and show how this technique improves lighting quality. Their assumption, that the lighting information varies linearly between sample points, is not necessarily true, and therefore does not eliminate all possible artifacts. Guetat et al. [GAMD10] propose using a non-linear interpolation model which more closely models how a constant length vector traces a curve when changing value rather than the standard linear solution which can result in normals of less than length 1. They show that this interpolation scheme more accurately models lighting information when compared to single sample lighting or dual sample linear interpolation.

Technically, pre-integrated rendering depends on both the distance between samples and the value of the front and back samples, and therefore the calculated information should be stored in a 3D texture. In order to reduce the calculation time of the lookup texture and to reduce memory requirements, the sampling distance is generally assumed to be constant and the texture is reduced in dimension. This constant sampling rate is a reasonable assumption for slice based rendering but makes pre-integrated rendering unachievable in real-time for adaptive sampling techniques such as those widely used in ray cast based volume rendering.

2.2.5 Ray Casting

Current state of the art volume rendering methods increasingly utilise image-order ray casting approaches [Lev90, Gar90, SSKE05, MSRMH09] as this technique provides the same image quality as texture slicing methods but with a lot more flexibility. Modern techniques utilise programmable GPU's and 3D textures in order to produce high quality computationally efficient raycast renderings of volume data. Most ray casting techniques consist of two passes. In the first pass, the front and back faces of the volume bounding box are rendered to an off screen buffer in order to generate the start and end positions for each ray. In the second pass, for each pixel on the screen a ray is cast into the volume using the start and end positions generated in the first pass. The algorithm then iterates along the ray, sampling the volume data and accumulating opacity and colour information until the ray terminates and the final result is output to the screen. Multiple optimisations to this approach have been proposed to further improve rendering speeds [MFS06].

Early ray termination [Lev90] is a widely used approach to optimise direct volume ray casting. This technique tries to determine when a ray has accumulated enough opacity that it can be terminated early without adversely effecting the final image quality. Once the accumulated alpha value of a ray reaches a certain threshold value (usually 95% alpha), the opacity of the ray is set to maximum and its current colour is outputted to the screen. This avoids shading calculations in areas of the volume which are highly occluded and do not significantly contribute to the final image, therefore improving rendering speed.

Several researchers have suggested exploiting temporal coherency and using information from the previous frame in order to optimise rendering performance. Yagel and Shi [YS93] propose modifying the start location of rays in order to reduce the number of sample lookups. They create a map of the first hit locations of each ray in the previous frame and then warp the map to take account of any camera movement that has occurred. Klein et al. [KSSE05] propose an extension to this technique which solves the problem of inconsistent regions in the ray start map caused by the warping process.

Empty space skipping is a more advanced sampling reduction technique used to accelerate volume rendering. Krüger and Westermann [KW03] propose using an octree structure containing the minimum and maximum values of the volume data which is accessed during ray traversal and used as a dependant texture lookup in order to determine areas of empty space which can be ignored. In similar work, Li et al. [LMK03] propose an empty-space skipping technique for texture based volume rendering. Çelebi and Çevik [ÇÇ10] propose a method for increasing the speed of isosurface ray casting by allowing the user to specify a leaping value. This value is used to skip sampling steps along the ray which provides a significant speed increase without reducing image quality as long the volume data is smoothly varying. When the sampling algorithm crosses the target isosurface value, it back steps through the last set of skipped samples in order to determine the exact position of the isosurface. The final image quality is only degraded in regions of the volume where the data changes quickly and the ray crosses the target isovalue twice between skipped samples.

One of the key advantages of ray cast volume rendering is that the sampling rate of each ray can be changed independently from other rays, thus allowing adaptive sampling techniques to be utilised. Roettger et al. [RGW+03] introduce the concept of an importance volume which is used to determine the sampling rate when iterating through the volume. The importance volume is usually of lower resolution than the volume data and is sampled during ray iteration to determine the next sampling point instead of using a fixed sampling distance.

Deferred shading [HSS+05] is an optimisation specific to isosurface rendering which defers shading calculations. Instead of performing shading calculations for every voxel that a ray encounters this technique outputs only the voxel's depth and intensity. In a later pass, this information is used to calculate normal and lighting information only for voxels which contribute to the final image, thus eliminating unnecessary lighting calculations.

A common artifact seen due to low sampling rates is a wood-grain effect caused by a sudden change in depth between neighbouring fragments. A common method used to hide this artifact is to randomise the sampling locations of each ray using stochastic jittering [HKRs+06]. In practice, the sampling rate is kept constant for each ray but the start points of the rays are randomised. Another approach to eliminating artifacts due to low sampling rates is virtual sampling, proposed by Lee et al. [LYS+10]. This technique uses the third order Catmull-Rom spline [CRBR74] to interpolate between sampling points and generates additional transfer function samples at discrete points along the spline. This avoids the additional volume data texture accesses required by a higher sampling rate but still requires additional transfer function sampling.

Howison et al. [HBC12] have explored how ray cast volume rendering can be accelerated when calculated on many core, many GPU supercomputers through the use of a hybrid parallelism approach.

2.2.6 Advanced Transfer Functions

Transfer functions assign colour and opacity properties to the scalar values contained in a volume dataset. Basic transfer functions map each scalar value in the dataset to a RGBA colour and opacity, but more advanced classification metrics have since been developed. Multidimensional transfer functions were proposed as a method to give the user more control of object segmentation and to extend the number of tunable parameters available to a user [KD98]. Since this initial work, multidimensional transfer functions have been expanded in many different ways to incorporate advanced metrics such as texture and structure size [CM08, CR08].

Hladůvka et al. [HKG00] propose using principle curvature magnitude as an additional transfer function domain. This technique is extended by Kindlmann et al. [KWTM03] who propose a method for displaying ridge and valley lines from volume data by specifying them in the curvature domain. This method uses the maximum and minimum principle curvature magnitudes to extract ridge and valley lines and avoids expensive third derivative calculations that more traditional curvature estimation methods use.

Kniss et al. [KKH01, KKH02] proposed using a 2D transfer function in order to simplify the process of extracting material boundaries. Their technique uses the scalar values, as a standard transfer functions does, but adds an additional factor: the gradient magnitude. They show how these two properties can be represented in a 2D widget to allow easy interaction and transfer function creation.

Csébfalvi et al. [CMH$^+$01] propose a method for visualising object contours in volume datasets based on the strength of the gradient magnitude and the angle between the view vector and the gradient vector. Their method provides a simplified, computationally efficient rendering of the internal structures of the volume data but does not provide the high level of detail that traditional direct volume rendering or isosurface volume rendering approaches normally achieve.

Lum and Ma [LM04] propose using higher level transfer functions in order to control lighting parameters. Their technique samples the volume data one unit in both the forward and backward gradient direction in order to create a multidimensional transfer function which enables enhancement of boundary regions while still illustrating material thickness.

Bruckner and Gröller [BG07a] propose style transfer functions that allow them to combine multiple different rendering styles into one rendering. Their technique uses sphere maps to specify styles based on the angle of the normal at each voxel, which allows them to simulate multiple different NPR styles easily. In addition, they extend their technique by including curvature as a classification feature, thus allowing contours to be rendered and allowing them to combine a multitude of different styles into one rendering. In related work, Rautek et al. [RBEG07] use multidimensional transfer functions to map voxels to a semantic style in order to generate illustrative volume renderings.

Ambient occlusion is proposed as a new classifier for use in multidimensional transfer functions by Correa and Ma [CM09]. They precompute an ambient occlusion volume and use this in conjunction with the original intensity volume to classify voxels based on intensity and ambient occlusion. In related work, the frequency distributions of areas surrounding each voxel have been used for classification [JH09]. Statistics such as mean, skewness, standard deviation, and covariance are all used as classifiers in order to segmented the volume data using this multidimensional transfer function approach.

Volume painting and sketching has been explored as a method for colouring volume rendering by a number of different researchers [BKW08, GXY12, BG05]. Guo et al. [GMY11] propose a technique which allows the user to apply eight different visualisation effects by sketching directly on the rendering. Their system analyses the sketch patterns in order to detect what features and areas of the volume the user means to manipulate. However, their technique operates on global parameters which means that all objects in the data with the same properties change with user interaction, not just the area that the user interacted with.

2.2.6.1 Combining DVR and Isosurface rendering

Several researchers have investigated how high quality DVR and isosurface rendering can be obtained simultaneously without increasing the sampling rate.

Knoll et al. [KHW$^+$09] analyse the transfer function for local maximums, which they assume to be isosurfaces. They then generate a 2D texture, similar to a pre-integration texture, which contains the locations of possible isosurfaces. This texture is queried during the rendering loop and if a potential isosurface is present an extra root solving step is performed in order to locate its position accurately.

Interval volumes combined with DVR and isosurface rendering are proposed by Ament et al. [AWC10]. This technique allows for the specification of sharp, isosurface like transitions in a transfer function with no aliasing artifacts. The technique models an isosurface as a range of scalar values rather than a single value as is normally done. This ensures that the isosurface is sampled and no aliasing occurs due to a low sampling rate, but does affect the opacity as multiple samples for one isosurface may occur. This opacity inconsistency is solved through the use of pre-integration tables to ensure that each interval volume has a set opacity value no matter how many times it is sampled.

In more recent work, Yang et al. [YLX$^+$12] propose combining DVR and isosurface rendering in a single ray casting pass. Their technique requires separate transfer functions for the isosurface and the DVR rendering and multiple tuning parameters. In addition, a CPU preprocessing step is required whenever the transfer functions changes and they are only able to achieve interactive rendering times for small datasets.

Advanced transfer functions give the user access to a multitude of parameters and techniques which allow for detailed data classification, however, this flexibility comes with the cost of additional complexity. Basic transfer function generation requires a skilled user in order to achieve pleasing results, and multidimensional transfer functions introduce a plethora of additional options and tunable parameters which further increase this skill requirement. As a result, multidimensional transfer functions either require long periods of user training or the development of specialised user interfaces in order to allow timely user interaction.

2.2.7 Memory Reduction

In order to improve memory requirements, researchers have investigated bricking as a method for increasing cache coherency and reducing the storage requirements of volume data. Initial techniques focused on rearranging voxels in memory in order to improve locality of reference and hence rendering times [PSL+98, WWE04].

Hadwiger et al. [HSS+05] also investigated reducing memory requirements through bricking. Their approach uses a low resolution 3D texture as a lookup into a dynamic high resolution texture which contains only the bricks required for rendering the current isovalue. This high resolution texture only maintains data for areas of the volume which can contribute to the final image, thus allowing for the possibility of a large reduction in memory requirements depending on the dataset characteristics.

Lum and Ma [LMC02] have investigated data compression to allow interactive volume renderings of temporal datasets, however, their technique requires lossy compression and there is scope to investigate non lossy solutions.

Guthe et al. [GWGS02] propose a lossy level of detail compression method for volume data in an attempt to reduce memory requirements for large scale datasets. They use a wavelet compression scheme and prioritise block resolution based on reducing reconstruction error. In similar work, Ljung et al. [LLYM04] use a wavelet compression scheme, in combination with transfer function analysis, in order to guide adaptive compression in an attempt to minimise compression in visually important areas of the volume. Wang and Ma [WM08] use a wavelet scheme to analyse volume compression schemes. In related work, Wang et al. [WGS07] use an image based level of detail technique to select which compression level to render in their wavelet memory reduction system. Their method takes account of occlusion and view dependence in order to render the data in a way which results in the least compressed image rather than the least compressed data, thus resulting in higher quality renderings for a given computation budget.

A vector quantisation based approach to memory reduction is proposed by Fout and Ma [FM07]. An interesting modification of their technique is that they utilize several different quantizers. Each block of the volume is classified based on its frequency content, and blocks with higher frequency information and thus more perceptually relevant information are given more weighting and less compression.

Crassin et al. [CNLE09] propose a technique for the rendering of large voxel datasets through the use of a multiple resolution, level of detail technique. Their method uses a form of octrees to store the dataset in multiple resolutions, which allows for efficient multisampling and avoids unnecessary calculations in occluded regions.

Another method for memory reduction, that has not yet been widely explored, is the use of tensors. Suter et al. [SIGM+11] compress volume data using bricking combined with a tensor approximation and then reconstruct the data on the GPU using the CUDA framework. This tensor technique allows for varying levels of reconstruction quality allowing for level of detail approaches to rendering. Their technique is able to perform real-time lossy rendering but requires preprocessing times of several hours for large datasets.

The vast majority of memory reduction techniques either require lossy compression, or long pre-processing times, or both. This limits memory reduction techniques to applications which do not require high quality renderings and which have no requirement for on the fly renderings of streamed or temporally variant datasets.

2.2.8 Time-Critical Rendering

Time-critical rendering is a method of dynamically modifying the quality of a rendering in order to achieve a target frame rate. It is used in a wide variety of fields in order to ensure that tasks are computed in a reasonable time frame.

2.2.8.1 General techniques

Although adaptive sampling has been explored in rendering and volume visualisation, it has traditionally been used in dynamic simulation through the use of adaptive time-stepping [JLA96, BW98].

Related to this, time-critical techniques in computer graphics employ progressive refinement processes which can be terminated before completion if processing time available for the rendering or simulation of the scene has elapsed. A coarser solution is returned whilst the rendering remains constrained to within a defined budget of computation time, thus ensuring target frame rates. Graceful degradation techniques will return this coarser result without excessively sacrificing precision, correctness, or stability of the frame being rendered, thereby optimising the tradeoff between speed and accuracy. Some early examples of this are the adaptive rendering technique of Bergman et al. [BFGS86], virtual environment walkthroughs by Funkhouser and Séquin [FS93], and the collision detection system of Hubbard [Hub96].

Gobbetti and Bouvier [GB99] investigated time-critical rendering for multiresolution polygonal models and proposed a method for ensuring time-critical rendering which could be implemented with any generic multiresolution data structure. Zach et al. [ZMK04] investigated dynamic switching between point rendered objects and discrete polygonal geometry in order to achieve real-time rendering of large scale datasets.

2.2.8.2 Time-Critical Volume Rendering

Time-critical computing for volume rendering has been employed by Li and Shen [LS01] who propose a fuzzy logic system for automatically selecting the correct level of detail threshold to use in order to achieve a user specified frame rate. They classify the current frame rate into three possibilities: high, medium, and low and define maximum sampling rate changes for each possibility. They are able to achieve a reasonably fast convergence on the targeted frame rate for texture slicing volume rendering by modifying the global sampling rate.

In an extension to their previous work, Li and Shen [LS02] investigate a technique for locally varying the sampling rate in order to achieve time-critical rendering of volume data. They propose generating a hierarchical octree structure for storing multiple level of detail (LOD) representations of the volume data and generating an importance value for each node. Their algorithm automatically selects LOD's from the volume structure in order to achieve the targeted rendering time. Their technique generates a texture for every LOD at the initialisation of their program and stores these textures on the GPU. This decision dramatically increases the memory requirements of the data and also requires a complete update of every texture whenever the user changes the transfer function. In addition, their technique requires that a new set of view aligned texture slices are generated for each level of detail node selected for rendering, and they do not comment on the overhead required for this step.

Weiskopf et al. [WWE04] propose a technique for texture based volume rendering which ensures constant frame rates. They propose splitting the volume data into bricks and orientating the data in the bricks in different directions in order to average out GPU caching behaviour.

In other work, Liao et al. [LLC04] investigate time-critical updates for time varying volume data. Their technique does not guarantee that the current volume data being rendered is accurate. Instead, they prioritise data upload to the GPU in areas of the volume which have changed the most from frame to frame. While their technique is innovative, it is only applicable to time variant volume data and does not provide any speed up for static datasets.

Bruckner [Bru04] proposes a three level technique for reducing the quality of volume rendering in order to achieve time-critical renderings. In the first instance, the resolution of the output image is reduced in order to improve rendering speed. Once a user specified minimum resolution is reached the overall sampling rate for the volume data is reduced and once this reaches a minimum the gradient filtering method is reduced.

2.3 Non-Photorealistic Rendering

Although some existing visualisation approaches already exploit a degree of multi-level rendering using NPR or other visual enhancements, there are a wide variety of more compelling perceptual techniques to be exploited in the field of NPR, which is closely related to perception and intuitive human techniques for creating imagery. Researchers have developed a wide variety of NPR techniques. The methods used range in scope from techniques such as, line drawings and cartoon shading which attempt to create a simplified version of the rendering; to enhancement methods which use NPR to augment rendering in order to enhance user understanding and perception of the dataset.

Non-photorealistic rendering is a key technique used in medical illustration and is an extremely important teaching tool. Certain NPR techniques are evident in illustrations from the medieval ages [PB07, DaV78], and NPR diagrams have now become a staple of medical education [Net97].

2.3.1 Image Space Line Extraction

Of particular relevance, there is a large body of work in NPR that has explored the perceptual and mathematical basis of line drawings, which are particularly useful in a multi-level rendering approach as they provide a non-occluding, self-contained dataset that can be integrated relatively easily to augment other rendering styles. There are numerous algorithms to extract lines from image space but some of the more common methods are: Difference of Gaussian edge detection [MH80], Sobel edge detection, and Canny edge detection [Can86]. Saito and Takahashi [ST90] give an overview of how some of these approaches can be used for line extraction from polygonal data.

Difference of Gaussian (DoG) edge detection [MH80] works by applying a Gaussian blur to the target image using two different blur levels. The resulting edges can then be found by calculating the difference between the two blurred images and finding all pixels with a difference over a user specified threshold.

Sobel edge detection is a simplistic and computationally efficient detection filter, which computes the colour gradient in the horizontal and vertical directions before thresholding the gradient to find scene edges.

Canny edge detection [Can86] requires a number of iterations over the image, each iteration with a different filter. First a Gaussian blur eliminates noise and then a Sobel filter is run on the image to find the basic edges. The edges are then thinned using non-maximal suppression, which operates on each edge dependant on its direction. Finally, the edges are traced and thresholding takes place using hysteresis.

Lee at al. [LMLH07] propose an image based technique to extract a large amount of lines that usually require expensive object space extraction methods. They generate a blurred tone image which is then analysed for ridges and valleys using a GPU fragment shader.

Image space line extraction techniques can quickly extract lines from an image, but the lines extracted are not represented in vector format and are therefore difficult to stylise. In addition, while these techniques are able to extract a wide variety of lines they are unable to classify them and therefore unable to stylise lines based on their classification as is done in many line rendering systems.

2.3.2 Object Space Line Extraction

More advanced lines can be extracted and displayed by considering the nature of the underlying 3D object. Instead of focusing just on the final image, object space line extraction techniques take account of the 3D nature of the datasets, which allows for the use of more advanced metrics such as surface curvature. Object space line extraction generally introduces additional challenges when compared to image space line extraction in addition to an increase in computation time.

2.3.2.1 Line Extraction for Polygonal Data

Well known classes of lines which can be automatically extracted in object space include: silhouettes, occluding contours, geometric ridges and valleys [KWTM03], suggestive contours [DFRS03], apparent ridges [JDA07], suggestive and principal highlights [DR07], and demarcating curves [KST08].

Suggestive contours are a class of lines proposed by DeCarlo et al. [DFRS03] which are drawn in areas where a true contour line would appear were the viewpoint to undergo a minimal change in position. They are an extension of contour lines and convey shape more effectively than contour lines alone.

Kim et al. [KYYL08] use curvature direction to guide stroke generation for polygonal scenes. Their technique is able to achieve real-time computation but is reliant on smooth, high quality normal information which is generally not available for volume data.

Cole and Finkelstein [CF09, CF10] investigated methods for computationally efficient visibility calculations for line drawings; while Kalnins et al. [KDMF03] describe a method for ensuring temporal coherency of silhouette lines as flickering and popping can be an issue with animated data.

Cole et al. perform experiments to determine correspondences between automated line drawing techniques and hand-drawn line illustrations [CGL+08]. The results provide an indication of lines that artists believe would most intuitively indicate relevant shape and structure and it is observed that these tend to largely correspond to image space lines and occluding contours.

A more recent study by Cole et al. [CSD+09] suggests that users are able to interpret certain shapes almost as well from line drawings as from shaded images and that current computer generated line drawing techniques can match the effectiveness of artists drawings in depicting shape.

Grabli et al. [GTDS10] describe a system for extracting line drawings from 3D data using GPU shaders. By analysing the surface properties of the 3D data their technique allows the user to create a programmable shader which specifies style and stroke attributes which can then generate renderings of the same style for arbitrary datasets. While this technique allows for line stylisations to easily be applied to arbitrary data it is very computationally expensive and is not currently applicable to interactive renderings.

2.3.2.2 Line Extraction for Volume data

Interrante et al. [IFP95] propose using ridge and valley lines to enhance transparent isosurfaces. They use the maximum principal curvature as a threshold in order to eliminate ridge and valley lines in regions which are not perceptually important. In addition, thin ridge and valley lines are eliminated by analysing the rate of change of the gradient in first principal direction. In later work, [IFP96, Int97] Interrante proposes using principle curvature direction as a method for orientating the lines of a stroke texture in order to enhance surface shape. This technique randomly distributes points across the isosurface and then uses a filtering kernel in order to trace streamline points in the principal curvature direction.

Salah et al. [SBS05] use point based rendering and draw lines using ellipses in order to extract lines from segmented medical data.

Hadwiger et al. [HSS+05] and Kindlmann et al. [KWTM03] use curvature based transfer functions to display contour lines and ridge and valley lines in their systems for real-time isosurface volume ray casting. These techniques do not extract vector lines, but instead colour the transfer function in areas where lines would occur. This is a minor point but can be important for line stylisations where actual line segments are required.

Burns et al. [BKR+05] extract a number of classes of lines directly from volume data, although they do not address how this could be combined with other rendering modes for a more complete volume visualisation. Their technique allows for interactive extraction of contour, suggestive contour and cutting plane intersection lines. Unfortunately, in order to achieve real-time performance they are forced to analyse only a subset of the volume data, which results in a number of lines being missed. They propose a number of techniques to improve these inaccuracies such as using the position of lines in previous frames as a seed point for searches and searching along the gradient direction.

Svakhine et al. [SEA09] present image based, outlining techniques that can generate pure line drawings or helpful feature enhancements for illustrative volume visualisation through the use of an unsharp mask. Their technique performs Difference of Gaussian line extraction on the depth buffer in combination with lookups into a user specified feature transfer function in order to extract contour lines.

Zhang et al. [ZHX+11] introduce Laplacian lines which are the zero point of a smoothed Laplacian operator which is equivalent to the image space second-order differential technique of the same name [Lin98]. They use the same approach as Burns et al. [BKR+05] to calculate Laplacian lines for volumetric data in real-time, and as such they suffer the same drawbacks as they are only able to analyse a subset of the volume data each frame.

Object space line extraction techniques are able to draw upon much more detailed metrics than image space techniques in order to generate extremely advanced lines. However, this additional complexity requires a large increase in computation when compared to image space extraction. Furthermore, this additional computation is exacerbated in volume rendering due to the large datasets frequently utilised. In order to achieve real-time object space line extraction, lossy rendering techniques or transfer function based approaches which sacrifice classification are necessary.

2.3.3 Illustrative Volume Rendering

Techniques for mimicking illustrative artistic styles have been investigated by a number of researchers. These techniques focus on the automatic generation of renderings which simulate hand drawn techniques such as paintings or stipple drawings.

Treavett and Chen employ a stylistic rendering approach resembling pen-and-ink illustrations for volume visualisation [TC00]. They note that such renderings can be overlaid with photorealistic rendering to create selective abstraction of detail and to improve, for instance, user understanding of translucent parts of the volume but perform no validation of this theory.

Lu et al. [LME$^+$02, LMT$^+$03] propose a technique for rendering volume data using stippling techniques in an effort to simulate hand drawn illustrations. Their method uses gradient magnitude and silhouettes to guide stipple distribution.

Yuan and Chen [YC04] combine illustrative renderings with direct volume rendering. They first generate geometric information by splatting point based primitives. Image space filters are then applied in order to generate line drawings and hatchings. This image is then applied as an overlay to direct volume rendering.

Researchers have also investigated hatching as a method for improving shape information. Cai and Dong [CD05] propose a technique for producing hatching on isosurface volume renderings by drawing strokes along the direction of minimum curvature. Other researchers [NSW02, DCLK03] propose similar techniques but instead of generating hatching on an extracted isosurface they instead compute the hatching directly from the volume data, resulting in less surface artifacts.

In further work, Dong and Clapworthy [DC05] propose using texture synthesis to stylise volume renderings. They compute a texture orientation for each voxel and then synthesise textures from a given sample texture and orientate it to the volume data. Their technique allows for a variety of different stylised renderings but takes several minutes to generate each image. Lu and Ebert [LE05] use texture synthesis to generate volume renderings which simulate the style of medical illustrations.

van Pelt et al. [vPVvdW10] use a particle based system called VolFilesGPU to generate illustrative volume visualisations. The system places particles near features on a user defined isosurface and then redistributes them to provide an even spread across the surface. The system implements a variety of NPR styles and incorporates splat based hidden surface removal at interactive frame rates.

Recent research has investigated how hatching parameters can be extracted from artists drawings in order to apply specific artistic styles to arbitrary data and views [KNBH12]. Currently, this technique has only been applied to polygonal models, but there may be scope to use this method for illustrative rendering of volume data.

2.3.4 Focus and Context Rendering

Focus and context techniques have been widely used for centuries as they allow users to view a particular object within the dataset at high quality while the remaining data is abstracted but sufficiently retained to provide context information. Cutaway and exploded views can be traced back to the fifteenth century and were used by Leonardo da Vinci for the illustration of human anatomy [DaV78]. Stemming from the original work of Furnas [Fur86], modern approaches for focus and context rendering using computational graphics have been proposed by numerous researchers for a wide variety of visualisations [LA94]. Preim et al. [PTD05] survey a number of existing areas where various enhancement and NPR techniques can be used to improve tasks in medical visualisation, and more recently Bruckner et al. [BGM+10] provide a survey of current techniques.

A variation of this focus and context approach which has been investigated for volume rendering is two-level rendering where different regions of the volume data are rendered in different rendering styles. A typical approach used by many researchers is to render the focus area using isosurface rendering and to render context regions using a transparent rendering techniques such as Maximum Intensity Projection (MIP) or DVR.

2.3.4.1 Distortion Lenses

LaMar et al. [LHJ01] propose a technique for focus and context volume rendering which utilises a magnifying lens to allow the user to zoom in on selected areas of the volume while keeping the surrounding area at the standard magnification level.

Ikits and Hansen [IH04] propose using a lens tied to the mouse position to allow the user to eliminate the context rendering and display the focus region without obscurance.

Wang et al. [WZMK05] demonstrate a number of different lens techniques for focus and context volume rendering, including modification of the magnification, sampling rate, and free-form lenses, all accelerated on the GPU.

2.3.4.2 Two-Level Rendering

Hauser et al. [HMBG00, HMBG01] propose the use of two-level volume rendering, which merges DVR and MIP techniques in an interactive tool. They argue that such an approach, based on a focus and context strategy, provides intuitive benefits to users, allowing them to peer inside inner structures, while keeping surrounding objects integrated for spatial reference. Their technique uses DVR with a step transfer function to render the focus area and MIP for the surrounding context.

Contour enhancement and tone shading are employed in combination with MIP compositing and DVR approaches for focus and context two-level volume rendering of pre-segmented datasets by Hadwiger et al. [HBH03] but they have not investigated image space NPR techniques.

2.3.4.3 Cutaway Views

Mcguffin et al. [MTB03] propose numerous tools for exploring volume data through the use of exploded views. Unfortunately, these techniques are implemented using a point based rendering system which results in very poor quality renderings and in addition, their implementation is unable to support transparent voxels.

Chen et al. [CSW+03] introduce the concept of spatial transfer functions, which describe geometric transforms for volume data, and use them to render exploded views for ray cast based volume rendering. Correa et al. [CSC06] extend this work to allow their splitting techniques to follow surface and segment alignment. Islam et al. [ISC07] further extend this work by combining spatial transfer functions with volume splitting.

Viola et al. [VKG05, VGKG04] discuss methods for highlighting and enhancing pre-segmented volume data through the use of cutaway and ghosted views of volumetric data. Each segment is assigned an importance which is used by the algorithm to ensure that the most important segment is always displayed clearly. They discuss a number of different composition methods and show how certain compositing techniques can be used to control either the visibility of the focus object or the border thickness of the context object.

The volume rendering system of Bruckner and Gröller [BG05] combines techniques such as cutaway and ghosted views in order to highlight pre-segmented volume data. In later work, Bruckner and Gröller [BG06] allow the user to select a focus object using volume painting and then define the surrounding volume as context information and automatically modify the geometry to generate an exploded view of the focus surface.

2.3.4.4 Level of Detail Approaches

Several researchers have investigated focus and context approaches for increasing rendering speed. These techniques reduce the quality of the rendering in context areas while high quality rendering techniques are applied in the focus area.

Levoy and Whitaker [LW90] propose using an eye tracker to track user gaze direction and increase the resolution of the output image in an area surrounding the gaze point.

Westermann et al. [WKE99] perform focus directed isosurface rendering. The user specifies a region of interest, which is extracted with a high quality marching cubes algorithm, while the surrounding region is extracted using a lower quality adaptive technique.

Weiler et al. [WWH+00] use a user specified focus point in combination with texture based rendering and texture bricking to adaptively vary the level of detail to ensure that focus areas are rendered in high quality.

Muigg et al. [MHDH07] split the volume data into bricks whose importances are classified by the user. This information is then used in order to generate a focus and context approach to rendering in which lower importance bricks are rendered at a reduced resolution.

2.3.4.5 Advanced Techniques

Nikolov et al. [NJA+01] propose using focus and context techniques to combine CT and MRI data to create a single region enhanced rendering.

Zhou et al. [ZDT04] propose using the distance of the voxel from the camera to modify opacity in the context region. Marchesin et al. [MDM10] eliminate opacity from the transfer function and instead modify volume opacity based on a function quantifying the relative importance of each voxel.

Tietjen et al. [TIP05] combine surface rendering, DVR, and line drawings to create a combined volume rendering system for surgery planning. Their technique achieves pleasing results but requires pre-segmented datasets and an object space line extraction algorithm as well as having some limitations when rendering semi-transparent surfaces.

Bruckner et al. [BGKG05] combine a number of metrics including the current accumulated opacity, gradient magnitude, and distance to the camera in order to produce a context-preserving volume rendering system which does not require pre-segmented data. In other work [BG05], they suggest a combined system for focus and context rendering.

Ropinski et al. [RSH06] propose combining virtual reality and focus and context rendering to create a system for visualising large scale seismic datasets.

Krüger et al. [KSW06] present a focus and context visualisation framework called Clearview, assigning varying levels of importance across a pre-segmented volume and rendering the volume adaptively using volume ray casting. Their technique can be utilised with both an isosurface and a DVR context layer and provides real-time renderings. Their technique takes into account a user specified focus region and modifies the transparency of the context region using a variety of techniques such as absolute distance, view distance, and surface curvature to produce a number of different renderings.

Kim and Varshney [KV06] use a visual saliency operator to compute an emphasis field, which is used to elicit viewer attention to relevant parts of a volume rendering.

In the field of medical visualisation, Gasteiger et al. [GNBP11] demonstrate a focus and context approach to blood view visualisation . Their technique combines streamlines, isosurface rendering, and a lens concept to allow the user to specify the focus area and visualise complex blood flow data.

A novel technique for focus and context rendering for volume data is proposed by Wang et al. [WWLM11]. They classify the importance of each voxel and then modify the size of the voxel based on its importance by deforming the volumetric mesh. Their technique increases the size of important areas of the image, but they fail to perform any user experiments to analyse whether their technique succeeds in enhancing relevant regions and if this large change in the shape of the dataset can cause visual distractions.

Focus and context techniques have been widely used in numerous research areas. Volume rendering focus and context techniques originally required the use of pre-segmented datasets, which had to be classified manually in a time consuming pre-processing step. More modern approaches have been developed which allow for automatic classification of the focus and context regions. Despite the large amount of research conducted in this area, there has been very little investigation into how focus and context approaches influence user perception. The renderings generated from these systems are clearly compelling, but their effectiveness at increasing user understanding of the data has received little scrutiny.

2.3.5 NPR Enhancements to Improve Perception

NPR techniques have been widely used to simulate artistic styles such as line drawings or painterly renderings, but they can also be used in to enhance visualisations and improve perception.

Ebert and Rheingans [ER00, RE01] propose combining several different NPR techniques to create a NPR volume rendering system which they claim enhances the structural perception of volume models. They combine several different techniques such as contour enhancement, distance based colour blending, and tone shading to create a complete NPR system to enhance the perception of volume data.

In similar work, Lum and Ma [LM02] employ hardware accelerated, parallel, non-photorealistic styles in their volume renderings, utilising object space GPU techniques for enhancement of the image. They use tone shading, silhouettes, and depth cues to provide perceptual improvements to the rendered volume data, but they rely on lossy compression in order to generate interactive renderings.

Similarly, Stompel et al. [SLM02] combine silhouettes, shading, gradient, and temporal enhancement to create a system for the improved perception of multivariate time-variant volume data. Their technique works at interactive rates and is also able to render strokes in order to give vector direction cues.

Lum et al. [LSM03] propose adding a moving particle overlay to volume renderings in order to enhance the 3D shape. Still images are replaced with animations that contain a particle system moving along the surface in the principal curvatures directions, and they show through a user experiment that their technique can enhance depth perception.

Chan et al. [CWM+09] propose a technique for opacity modification of pre-segmented datasets in order to enhance isosurface perception.

The addition of halos to DVR in order to enhance perception has been explored by a number of researchers [BG07b, DVND10]. Diaz et al. [DVND10] combine halos with a fast method for ambient occlusion in order to emphasise segmented structures and enhance depth perception. They propose two possible methods for generating these renderings, a lower quality image space technique which can be computed on the fly but is unable to account for 3D information and a view independent object space technique which can achieve interactive rendering but requires a preprocessing step. Their image space method is unable to calculate ambient occlusion for semi transparent voxels and is, therefore, unsuited for DVR rendering and can only achieve realistic results for isosurface display. Their object space technique requires a preprocessing time of only a few seconds, but unfortunately the resultant acceleration data requires a large amount of memory storage. While this technique can conceivably calculate ambient occlusion for DVR renderings, interactive performance can only be achieved for isosurface renderings.

In other work, Ruiz et al. [RBFS10] use the high frequency information in volume data to guide saliency information. By taking the difference between the original volume data and filtered version of the data, an unsharp mask can be created which is used to modulate colour and opacity and also to guide stipple rendering.

Šoltészová et al. [ŠPV11] use insight from medical rendering in order to render NPR shadows to improve perception. They propose using a shadow map to colour shadows rather than the more traditional method of luminance reduction. This coloured shadows technique still provides the depth cues that traditional shadow rendering provides but does not reduce image contrast. They verify their technique with a number of user experiments which show that their technique improves surface and shape perception but results in a reduction in depth cues.

In somewhat similar work, Chen et al. [CCB11] propose modifying the colours used in DVR rendering in order to produce images which can be easily viewed by users who suffer from colour vision deficiency (CVD). Previous techniques have modified the colouring of the final image to produce CVD-friendly colours. Chen et al. perform a number of user studies which show that their method of modifying the colouring of the transfer function and performing CVD-friendly colour blending can achieve pleasing and consistent results.

2.3.6 Other Relevant NPR Techniques

In this section we discuss some other NPR techniques which are relevant to the work described in this thesis.

Halper et al. [HMH+03] discuss the potential of NPR to affect a user's response to a rendered scene and show that a user's judgement or choices can be influenced by certain rendering and modelling choices.

Santella and DeCarlo describe how variable levels of abstraction can be generated in NPR to achieve modulated saliency and to visually prioritise certain regions of an image over others [DS02, SD04].

Cole et al. [CDF+06] discuss specifically directing user gaze in 3D with stylised focus. This is achieved by variable levels of NPR abstraction and stylisations, creating higher saliency in the most relevant parts of a scene. In similar work, McNamara et al. [MBG08] propose an image space modulation to the scene, which they call subtle gaze direction (SGD), in order to direct a viewers gaze to certain regions in the scene.

In similar work, Redmond et al. utilise a combination of object and image based strategies, accelerated on the GPU, to provide adaptive abstractions, resulting in user attention being drawn to selected parts of a rendered 3D scene [RD08, RD09a, RD09b].

Gooch et al. [GGSC98] have investigated using colour temperature in order to enhance the depth perception of an image. They note that traditional illustrators have used colour temperature in order to enhance the depth perception of their drawings, and they describe how to position light sources in such a way to exploit this method and enhance the depth cues of a rendered image but have not investigated using this technique to enhance volume renderings.

Cartoon shading [LMHB00] has been implemented in graphics hardware in an attempt to mimic the large areas of constant colour usually present in cartoons.

Kuwahara et al. [KHEK76] have developed an image space, edge preserving blur technique which works on a per-pixel basis in order to abstract the final image. The filter works on a per-pixel basis throughout the image and calculates the mean colour and variance in the four adjacent regions of each pixel. These regions are generated by dividing the surrounding area into four overlapping windows, each containing the centre pixel itself. The output colour for each pixel is the mean colour for the adjacent window with the smallest variance. This has the effect of smoothing internal regions while keeping the edges between colours sharp.

Although the approaches described in this section deal with 2D images and/or 3D polygonal data, they serve as evidence of a strong potential for exploiting similar strategies to aid or influence a users interaction and understanding of volume visualisation.

2.4 Volume Shading and Illumination

Illumination calculations can have a wide variety of effects that increase overall understanding of images, thus making illumination very important for scientific visualisation. Depth discrimination, local surface properties, and spatial structure awareness can all be enhanced through the addition of shading and shadowing [LB00, Pue89, KKMB96, ŠPV11, CF07]. In addition, inconsistent lighting has been investigated by a number of researchers as a method for enhancing features [LHV04, RBD06].

Various different techniques have been proposed to generate illumination calculations for volume rendering. Max and Chen [Max95, MC10] provides a high level summary of the various different optical models used in volume illumination schemes. Ropinski et al. [RKH08] give an overview of ray casting illumination techniques, and Lindemann et al. [LR11] compare how various different illumination techniques can improve perception. Advanced lighting techniques such as refraction and caustics [RC06, ARC05] have been investigated for volume rendering. In addition, volume techniques and data structures are now being used in rasterisation engines to compute high quality, real-time, interactive lighting for polygonal scenes [CNS$^+$11].

2.4.1 Local Illumination

Gradient based shading is one of the simplest way of adding shading to volume information. This technique calculates a gradient for every voxel in the volume. This gradient can then be converted into a normal direction and used in a standard lighting model such as Blinn-Phong illumination [Bli77].

A number of different techniques can be used to extract the gradient information, Bentum et al. [BLM96] provide an overview of some of the popular methods. In general, on the fly computation of gradients delivers better results than pre-computation due to the aliasing the occurs when gradient data is stored in GPU texture memory. Unfortunately, gradient calculation can be computationally intensive, and most on the fly computation schemes generally use a simple finite differences scheme in order to compute gradient direction.

Researchers have explored several advanced techniques for computing gradient direction and magnitude. Neumann et al. [NCKG00] propose using 4D linear regression in order to improve surface smoothing. Kalbe et al. [KKG09] propose a spline fitting solution which provides high quality visualisations of the surface and provides smoothly varying normals. Unfortunately, their technique is extremely computationally expensive and is currently limited to isosurface rendering.

As well as using the gradient direction, researchers have also proposed using the gradient magnitude in order to vary the shading of volume rendering. Levoy [Lev90] proposed using the gradient magnitude to scale the opacity of the volume data in order to emphasise the boundary between data of different intensity values. Ebert and Rheingans [ER00] expand on this technique by combining both transfer function and gradient magnitude opacity and allowing the user to set the contribution of each.

In recent work, Kroes et al. [KPB12] develop a system for photorealistic volume rendering which they call Exposure Render. Their system uses Monte Carlo ray tracing combined with a physically based lighting model, which allows for real world effects such as depth of field. However, in order to achieve interactive results they require low resolution preview renderings and single scattering lighting calculations.

2.4.2 On the Fly Illumination

Phong lighting [Pho75, Bli77] has been widely used to illuminate and shade volume data [Lev88], but in recent years more advanced lighting techniques have been developed in order to provide more realistic images.

Half angle slicing [KPHE02, KKH02, KPH$^+$03] is a technique proposed by Kniss et al. which simulates light propagation for texture based volume rendering. Texture slices are aligned along the half angle between the light direction and the view direction instead of aligned parallel to the view direction, as is normally done. This allows the illumination and the final image to be iteratively calculated one after the other, thus eliminating a separate illumination calculation step. This half angle slicing technique causes artifacts to occur as the angle between the light and the view direction increases. Šoltészová et al. [ŠPBV10] propose a modification to this technique which uses view-aligned slices and a cone shaped illumination kernel. This illumination kernel is projected onto the slices, resulting in an elliptical footprint, which allows for high quality shadowing effects without the artifacts associated with half angle slicing.

Sundén et al. [SYR11] propose a technique for ray cast volume rendering illumination which requires no preprocessing, achieves interactive results, and does not increase memory requirements. Their technique divides the image plane into sweep lines and casts rays iteratively for each line rather than in parallel for the full image. This reduces GPU parallelism but allows them to utilises a plane sweep paradigm in order to iteratively generate the final image by propagating the illumination forward from each sweep line. Their technique requires that the full volume be present on screen in order to achieve accurate lighting as they are unable to calculate light interaction for sections of the volume which are not contained within the image frustum. In addition, due to the iterative nature of plane sweeping their technique is unable to account for any backwards scattering effects.

2.4.3 Shadow Volumes

Shadow volume techniques precompute a light volume texture which stores luminance and scattering information [BR98]. During the rendering process, the volume texture is queried in order to access the luminance value of the current voxel and to modify the Phong lighting colour.

Deep shadow maps [LV00] are used in geometry rendering, whereby a map is created which stores a visibility function which represents how the visibility changes as light passes through each piece of geometry. More closely related to volume rendering, opacity shadow maps [KN01] store the visibility function at discrete sampling steps. Hadwiger et al. [HKSB06] introduce a GPU based version of deep shadow maps for volume rendering which utilises bricking structures in order to reduce memory requirements.

Desgranges et al. [DEP05] suggest using a dilated version of the shadow volume in order to produce a diffuse shading effect. Ropinski et al. [RDRS10] propose computing a shadow volume using texture slices aligned to the bounding box of the volume. In order to avoid artifacts when the slice axis is changed they compute light propagation in two directions and blend the results together to create a final shadow volume. This volume is then compressed using knowledge of the human visual system in order to reduce its memory footprint.

Schlegel et al. [SMP11] utilise summed area tables (SAT) and extinction based shading in order to simulate ambient occlusion, colour bleeding, and soft shadows. Their technique requires several preprocessing steps to compute the SAT, and this preprocessing needs to be recomputed on transfer function change or light source movement. In order to achieve interactive rendering times they make a number of assumptions and simplifications and are unable to perform the computation of hard shadows. The preprocessing step generates a 3D texture which stores the extinction coefficients for each voxel. The resolution of this texture can be reduced in order to reduce memory requirements and increase computation speed but this results in a blurring of shadows and lighting.

Several shadow volume techniques exist for isosurface volume rendering [BGB+06, WPSH06, BB07]. The proposed methods precompute radiance information for multiple isosurfaces, store the result in a volume texture and then interpolate between the isosurface illumination values at runtime. These methods work well for soft shadows but the linear interpolation scheme generally used, results in poor modelling of sharp discontinuities in lighting such as those caused by hard shadows.

Shadow volume based lighting techniques can generate very advanced lighting calculations but due to their nature, increase the memory requirements and pre-processing required to generate a final rendering, and therefore can be unsuited for certain rendering applications. In addition, most techniques require a recomputation of the shadow volume whenever the lighting parameters or transfer function change.

2.4.4 Ambient Occlusion

Ambient occlusion is a shading technique which takes into account reduction of light due to surrounding material rather than directly calculating lighting propagation [ZIK+98]. Stewart [Ste03] describes a technique for vicinity shading which simulates ambient occlusion for volume datasets by calculating the occlusion in a local area surrounding each voxel and suggests some optimisations, but this technique still requires run times in the order of minutes. This method searches a local area surrounding each voxel and reduces the luminance if any voxel in this area has a higher opacity than the voxel being analysed.

Ruiz et al. [RBV+08] modify Stewarts algorithm by weighting the reduction in luminance according to the inverse square root of the distance between the current voxel and the occluding voxel which they claim results in more realistic renderings. Their technique was implemented on the CPU so no accurate performance numbers were given. In similar work, Schott et al. [SPH+09] use view aligned slices and conical projection to produce a shading model which approximates ambient occlusion to achieve interactive performance.

Desgranges et al. [DE07] propose a technique for computing ambient occlusion by casting rays into a small sphere surrounding each voxel. By using pre-filtered volume datasets they are able to achieve fast recomputation of the ambient occlusion term. Similarly, Hernell et al. [HLY07, HLY10] propose a method for computing ambient occlusion by casting rays into a sphere surrounding each voxel but require a preprocessing step to compress the volume data in order to achieve real-time results. Ropinski et al. [RMD+08] propose a technique for ambient occlusions which requires an offline preprocessing step but does not require recalculation on transfer function changes. Their technique computes a histogram of intensity values surrounding each voxel which is then compressed in a vector quantisation step. This compressed histogram is used in the rendering stage to calculate the colour of the surrounding voxels and approximate ambient occlusion.

Salma [Sal07] proposes a technique for high quality isosurface rendering, incorporating scattering and ambient occlusion, using a Monte Carlo ray casting technique. This technique simulates scattering by casting rays in random directions and calculates ambient occlusion by casting rays with a large step size in a hemispherical area centred on the gradient direction but does not achieve interactive performance.

Penner and Mitchell [PM08] precompute a volume which stores the average and the squared average of the intensity of the surrounding voxels. This volume is then blurred and down sampled in order to approximate the effect of a local sampling area. During the rendering stage, this shadow volume is sampled and the average and variance of the surrounding voxels is calculated and used as an approximation of a histogram of the surrounding area in order to calculate ambient occlusion. Ruiz et al. [RSKU+10] extend this method by analysing a local bounding box around each voxel and computing the minimum, maximum, and mean of the surrounding density values. This information is calculated in a preprocessing step on the GPU, implemented in CUDA and produces ambient occlusion maps which are smoother and less noisy than previous work.

Janiek et al. [JBB+08] combine ambient occlusion with polygon rendering in order to visualise MRI and fMRI data.

2.4.5 Spherical Harmonic Lighting

Spherical harmonics (SH) are a method for decomposing a spherical function and expressing it as a series of coefficients that represents how the function varies over a unit sphere. They can be thought of as the spherical equivalent of decomposing a 2D function into a Fourier series. These spherical coefficients can be used to reconstruct the original function and are suited for use in radiance calculations as they can be easily used to represent how occlusion and lighting information changes in a sphere around each point in a scene.

Sloan et al. [SKS02] propose a method for volume shading using precomputed radiance transfer functions. This technique precomputes a mapping for every point, which given any incoming radiance, transforms it into the corresponding outgoing radiance. This transfer function models complex lighting interactions and is generally a detailed spherical function stored in a SH basis. The preprocessing time for this technique can take several hours and requires a new update every time the transfer function is changed.

Wyman et al. [WPSH06] propose using spherical harmonics to store precomputed illumination information for isosurface volume rendering. They compute the global illumination for each voxel at multiple isovalues and then interpolate the results depending on the user selected isovalue. Unfortunately, this technique is prone to artifacts in areas with hard shadows due to their interpolation scheme not modelling the abrupt change in luminance that hard shadows typically exhibit.

Ritschel et al. [Rit07] use Monte Carlo ray casting to compute a 3D visibility function. This is done in an offline preprocess step by casting rays in multiple directions from every voxel, through the dataset, accumulating luminance information. The results of these raycasts are converted to a spherical harmonic basis, packed into a 3D RGBA texture and uploaded to the GPU for use in the rendering pass. For medium to large datasets changing the transfer function and thus initiating a new preprocessing step requires several minutes and rendering times are on the order of seconds.

Lindemann and Ropinski [LR10] precalculate lighting information in order to calculate volume renderings with advanced material properties. Their technique accounts for shadows, scattering, and bleeding effects by generating separate volumes representing the illumination information for each effect. These volumes store spherical harmonic representations of the lighting effects and are combined together at runtime to produce the final image. Lindemann and Ropinski show that interactive results can be achieved for small volumes with just shadowing effects enabled, but large volumes with full lighting require processing times in the order of minutes for transfer function changes.

Kronander et al. [KJL+11] suggest a two pass algorithm using spherical harmonics in order to produce DVR renderings which support dynamic illumination. In the first pass, local visibility is computed by casting rays in a small radius surrounding each voxel and the results are stored in a SH basis. In a second pass, global visibility is computed by casting rays through the entire volume and sampling the results of the first local illumination pass. This second pass is computed at much lower resolution than that of the volume data in order to achieve reasonable computation times (usually on the order of 1–2 seconds). Their technique requires an update of the visibility calculation on transfer function changes which takes several seconds but achieves real-time rendering performance.

In order to achieve practical computation times most techniques require that the number of SH coefficients is limited to a finite number, resulting in the loss of high frequency luminance changes. This means that spherical harmonic techniques are generally limited to rendering only soft shadows and have difficulty displaying accurate hard shadow renderings.

2.4.6 Illumination to Improve Perception

Several researchers have investigated how different illumination techniques can improve perception of complex images.

For polygonal models, O'Shea et al. [OBA08] perform a number of user experiments on diffusely lit objects and show that something as simple as the light position can influence users shape perception. They show that a light positioned approximately twenty degrees above the horizontal is optimum for surface perception. Sun and Perona [SP98] show that there is a preference for a light positioned to the left of the camera rather than to the right.

Langer and Bülthoff [LB00] show, through a number of user experiments, that depth discrimination can be improved by positioning the light above the horizontal.

Hubona et al. [HWSB99] show that the addition of shadows enhances the accuracy of depth estimation but interestingly does not improve the speed of estimation. Additionally, they show that the use of stereoscopic viewing can further enhance depth perception and that the addition of a second light source decreases depth estimation accuracy.

Puerta [Pue89] displayed stereoscopic photographs of an object to participants. The object was photographed in such a way that the object itself was identical in both images and therefore provided no depth cues, however, the shadows were manipulated to show parallax. Participants were shown these images, with and without the shadows displayed, and they stated that they experienced a sense of depth for the images containing shadows, and in most cases they thought they were looking at a standard stereo pair of images, showing that the addition of shadows alone can enhance depth perception.

Lindemann and Ropinski [LR11] test a number of different illumination styles for volume rendering and analyse how each technique influences various factors such as shape and depth perception.

2.5 Conclusion

We have presented a review of the related work in the field of volume rendering. Specific areas which were reviewed in detail include focus and context rendering, performance improvement techniques, NPR additions, and lighting and shading solutions. Focus and context rendering techniques are relevant to all the following chapters as this rendering method is a central theme of this thesis. NPR stylisations and their application to volume rendering are particularly relevant to Chapter 3 and also discussed in Chapter 5. Performance improvements for volume rendering and time-critical techniques are relevant to Chapter 4 and lighting and illumination solutions for volume rendering relate to Chapter 5.

Chapter 3

Non-Photorealistic Volume Rendering Enhancement

3.1 Introduction

Non-photorealistic rendering (NPR) techniques have been investigated by a wide variety of researchers for numerous reasons. Some techniques focus on mimicking artistic styles in order to generate images which can pass as facsimiles of artistic drawings. Other techniques focus on producing simplified renderings of complex objects in order to convey a complex image in a minimised manner. Yet other techniques focus on adding NPR enhancements to renderings in order to improve user understanding and perception of a complex scene.

In this chapter, we discuss techniques to enable focus and context renderings for volumetric data. We investigate current NPR methods that can enhance and improve user understanding and apply them to the focus area of the scene. We also investigate NPR techniques that abstract and simplify data and consider how they can be applied to the context region of a scene.

We develop a number of GPU based NPR methods for both focus enhancement and context abstraction. By combining these two NPR approaches together we create a direct volume rendering (DVR) system which utilises NPR techniques to create a two-level, focus and context rendering of complex volume data. We describe the technical implementation of this system and suggest novel methods to ensure interactive performance.

In addition, we evaluate our framework by using saliency metrics to show how our combined focus and context rendering system can modify the most salient point in the image. We perform a number of user experiments, and the results indicate that our approach can provide a significant improvement in user perception of shape in complex visualizations, especially when a user has little or no prior knowledge of the data. The results of these user tests also allow us to recommend specific NPR styles for optimum image enhancement and provide novel insights into optimising shape perception.

3.2 NPR Enhancements

This section describes the NPR techniques that were developed using GPU shaders and integrated into our NPR focus and context framework. Most of the techniques described here were originally proposed as CPU based enhancements and they required some modifications in order to allow them to work in real-time on the GPU. The Difference of Gaussian, Sobel, Canny, Kuwahara, and saturation techniques were implemented in conjunction with another researcher.

3.2.1 CPU Based NPR

The NPR techniques described in this subsection are line extraction methods that have been implemented on the CPU. These techniques allow for high quality extraction of line segments which can be subsequently stylised but are very computationally expensive to compute.

3.2.1.1 Contours

Contour lines, or silhouette lines as they are sometimes called, are lines drawn in areas of an object which are perpendicular to the view direction. They can be extracted by finding areas in an image where the dot product between the object normal and view vector is zero. These view dependant lines are strong shape descriptors which can enhance the extents of an object.

3.2.1.2 Suggestive Contours

Suggestive contours were first proposed by DeCarlo et al. [DFRS03] and complement contours in conveying shape effectively. They can be thought of as regions of the image where contours would be drawn if the view point was moved slightly and in fact extend contour lines.

Burns et al. [BKR+05] propose a technique for extracting contours and suggestive contours from volume data but are unable to achieve real-time performance for a complete line extraction. Instead, they propose a number of methods which reduce the

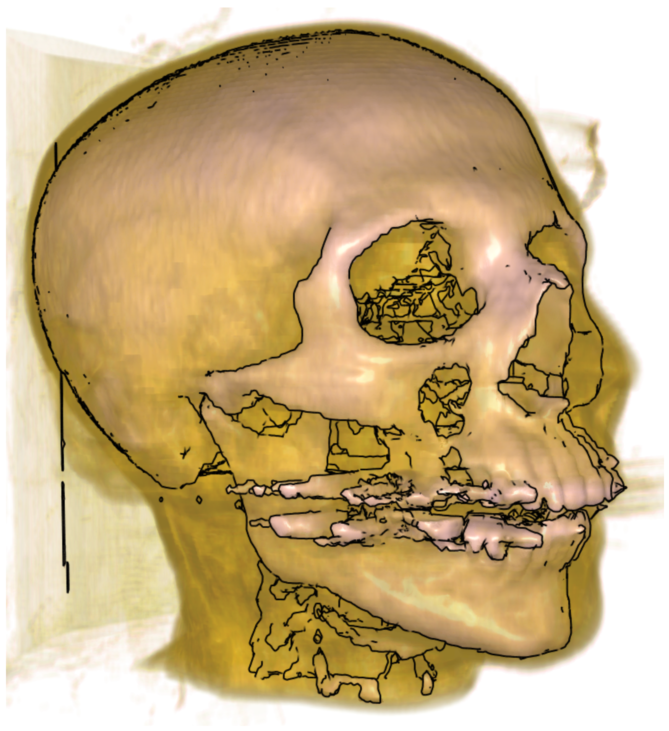


Figure 3.1: Contour lines.

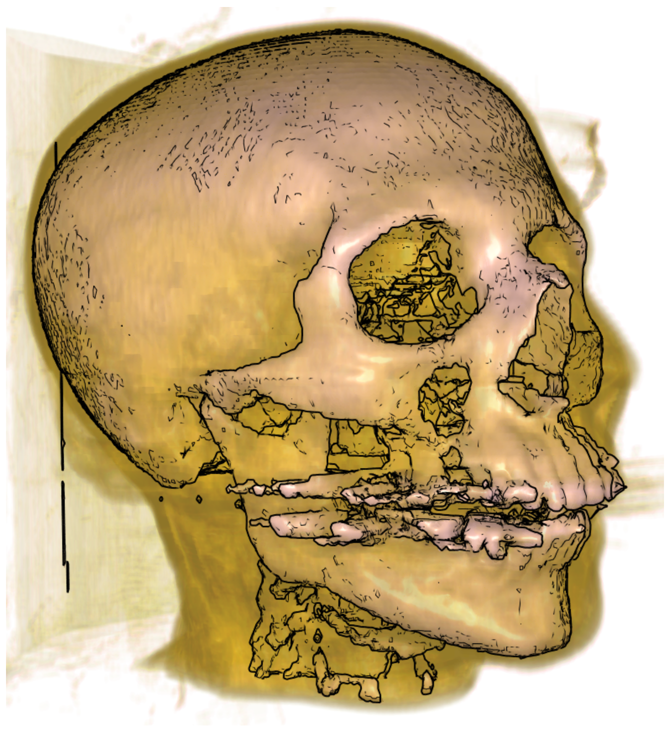


Figure 3.2: Suggestive contours combined with contour lines.

computation time but result in some line extractions being missed. Additionally, they have not investigated how contours and suggestive contours can be used to enhance volume shape perception.

Their technique splits the volume data into cubes where each corner of the cube corresponds to a voxel data value. By analysing the voxel intensity values at each corner of the cube and using linear interpolation, a surface corresponding to the user selected isovalue can be extracted. In a similar manner, by linearly interpolating the contour function ($n \cdot v = 0$), a surface representing all possible contours can be extracted. By intersecting these two surfaces together, contour lines for the user specified isovalue are extracted. In order to achieve interactive performance their technique only analyses a subset of the volume data and is unable to extract all possible lines.

3.2.2 GPU Based Image Space NPR

Image space NPR techniques work by analysing the final 2D image generated from a rendering system. In this section, we describe a number of different image space NPR techniques all of which have been ported to the OpenGL Shading Language (GLSL) or to C for Graphics (CG) and implemented on a GPU. Modern consumer level GPU's or graphics cards typically contain many hundred processing cores working in parallel and are ideally suited for interactive implementations of highly parallel algorithms. The image space NPR techniques introduced in this section are all highly parallel in nature and are thus ideally suited to GPU implementations. By porting these techniques to GPU shaders we were able to achieve a dramatic decrease in computation time in comparison to the same algorithm implemented on the CPU.

The first three techniques we describe are image space line or edge extraction algorithms. Edge stylisation can play an important part in NPR. Emphasizing edges in a scene enhances distinction between regions and can increase the saliency of a particular object or area. A large number of different edge extraction algorithms have been proposed, but the basic idea of all these techniques is to find and draw lines in areas of the image where brightness changes quickly. The majority of image space line extraction techniques involve a series of operations performed locally around each pixel in order to determine whether a line exists at that point or not. These operations are repeated for every pixel in order to generate the final line drawing image.

The final two techniques described in this section are peripheral abstraction techniques which reduce detail in a region. Previous work by Redmond et al. [RD09a, RD09b] has shown that a users perception of a scene can be altered by not only adding detail to the target object, but also by abstracting extraneous information. In our system, this is achieved by implementing two varied abstraction styles chosen due to their success in past work: saturation variation, and the Kuwahara filter. Both techniques were implemented using GPU shaders and achieve real-time frame rates.

3.2.2.1 Difference of Gaussian Line Extraction

Difference of Gaussian edge detection [MH80] works by subtracting two blurred versions of the original image from each other. Each image is blurred using a Gaussian image filter, but the two filters have different standard deviations. The difference image is then analysed and all pixels that pass the threshold value are classified as points on a line (Figure 3.3). In order to give the most effective result, we utilised the Marr-Hildreth [MH80] ratio of $1:1.6$ to decide the relative difference in standard deviation between the two Gaussian filters resulting in the two filters shown in Equation 3.1. Our CG implementation is shown in Listing B.1.

$$\sigma_1 = \frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix} \quad \sigma_{1.6} = \frac{1}{782}\begin{bmatrix} 13 & 23 & 28 & 23 & 13 \\ 23 & 42 & 51 & 42 & 23 \\ 28 & 51 & 62 & 51 & 28 \\ 23 & 42 & 51 & 42 & 23 \\ 13 & 23 & 28 & 23 & 13 \end{bmatrix} \tag{3.1}$$

3.2.2.2 Sobel Line Extraction

The Sobel operator approximates the gradient direction and magnitude for each pixel in an image and is used in a number of different edge detection algorithms. For 2D images it consists of two 3x3 filters which are applied to every pixel, one filter in the horizontal direction and the other in the vertical.

$$S_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \tag{3.2}$$

These two filters give the magnitude of the gradient in the horizontal and vertical directions and the results can be used to calculate the gradient magnitude: $\sqrt{{S_x}^2 + {S_y}^2}$ and the gradient direction: $\arctan(\frac{S_y}{S_x})$. Pixels are only considered to be edge points if their gradient magnitude is greater than some user defined threshold (Figure 3.4). Our CG implementation is shown in Listing B.2.

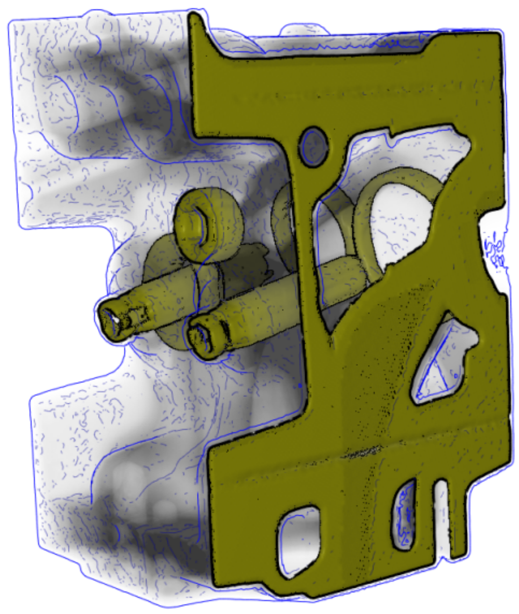


Figure 3.3: Volume rendering of an engine block with the target isosurface highlighted using Difference of Gaussian edges and the surrounding volume enhanced using suggestive contours.

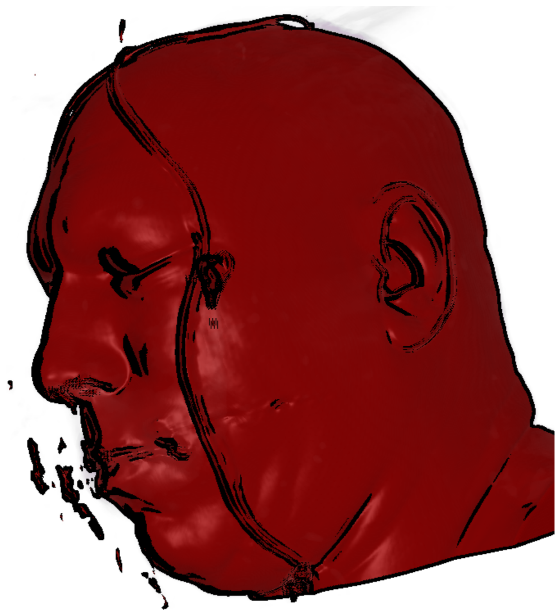


Figure 3.4: Sobel edge detection.

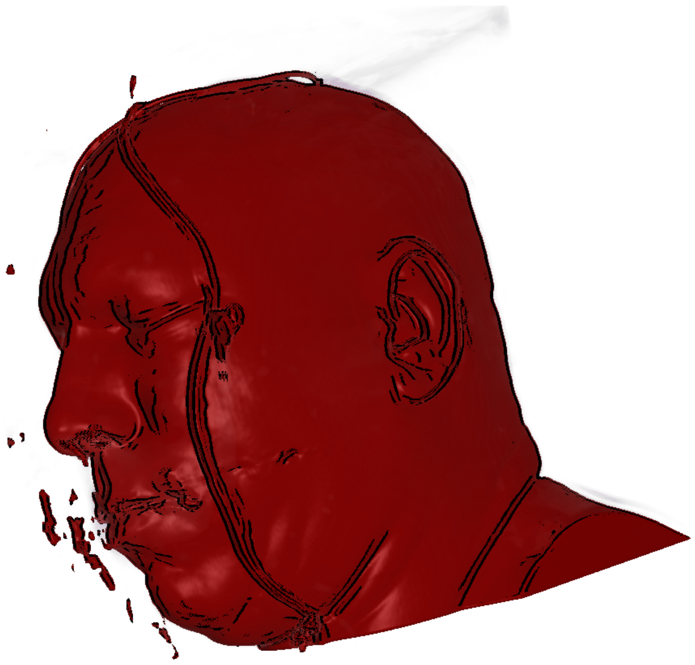


Figure 3.5: Canny edge detection.

3.2.2.3 Canny Line Extraction

Canny edge detection [Can86] is an advanced image operator that involves multiple stages.

In the first stage, the image is blurred to eliminate noise and reduce the possibility of false edge detection. This is accomplished by convolving the image with a Gaussian filter (Equation 3.3) similar to those used in the Difference of Gaussian approach described on page 38.

$$\frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{3.3}$$

In the second stage, the gradient information of the blurred image is calculated using the Sobel operator as described in the previous section, but any useful gradient operator may be used. Since we have both the magnitude and direction of the gradient an extra computation step called non-maximum suppression is performed. This tests for local maximums of gradient magnitude perpendicular to the gradient direction in order to thin the edges.

In the final step of the technique, edges are traced through the image and two-level thresholding is performed. This two-level or hysteresis thresholding requires the user to input two threshold values: upper and lower. If a pixel has a gradient magnitude higher than the upper threshold it is accepted as an edge. Using pixels classified as edges as seeds, lines are traced through the image using the *lower* value as a threshold and classified as edges only if they contain a continuous line of pixels all greater the the lower threshold, eventually connecting to a pixel with a magnitude greater than the upper threshold. This final step of the algorithm is not easily parallelised and is therefore not suited for a GPU shader implementation. A CPU implementation is unable to achieve interactive results, and therefore this final step was omitted from our edge detector. Even without this final step of the Canny algorithm, we were able to achieve plausible results as can be seen in Figure 3.5. This technique was developed early on in the course of this research and since then the Nvidia CUDA framework has been developed which would now allow this final step of the algorithm to be implemented on the GPU. Our CG implementation is shown in Listing B.3.

3.2.2.4 Kuwahara Filtering

The Kuwahara filter [KHEK76] is an image based, edge preserving blur filter. It smooths an image while maintaining important edges and produces painterly like results. The filter analyses a local neighbourhood around each pixel and modifies the pixel colour based on the content of the surrounding region. It works by analysing the pixels surrounding each point in four different overlapping regions and setting the current pixel value to the mean of the region which has the least variance. This results in only relatively homogeneous regions contributing to the output, and therefore insuring the edges are not averaged. The resulting image (Image 3.6) preserves the position and size of edges in the original image while blurring smooth areas. Our CG implementation is shown in Listing B.4.

3.2.2.5 Saturation Variation

Saturation variation is a technique which performs simple operations on RGB values in order to fade the image's colour. Previous work [RD08, RD09b] has shown this technique to attract user attention and provide perceptual improvements in user studies.

This technique was implemented in an image space GPU shader by converting RGB colours into the HSL/HSV colour space and then reducing their saturation component by a set amount. The resulting colour was then converted back into RGB space, output from the shader and displayed, resulting in a muting or reduction in colour (Image 3.14).

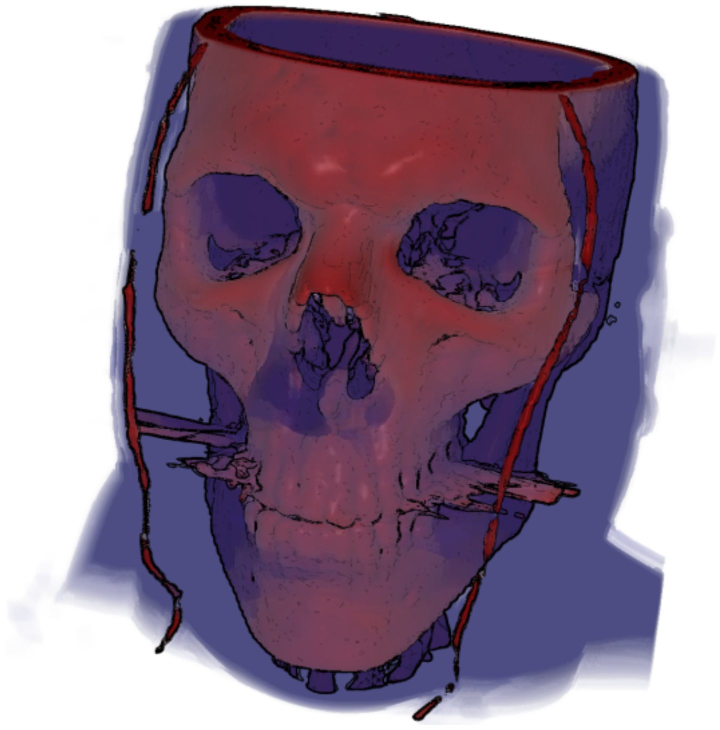


Figure 3.6: Sobel edge detection combined with suggestive contours and Kuwahara filtering applied to a volumetric model of a skull.

3.2.3 GPU Based Object Space NPR

Object space enhancements work by analysing the 3D information available in a scene. They rely on having access to the full scene information and use information about the 3D nature of an object rather than just its 2D projection. In general, for volume rendering these techniques are calculated during the ray casting pass and integrated into the rendering pipeline as opposed to the image space techniques described earlier which are calculated as a post-processing step after volume rendering has been performed.

3.2.3.1 Tone Shading

Tone shading, also know as Gooch shading, is a technique proposed by Gooch et al. [GGSC98] which attempts to make the shape, structure, and material composition of an object easier to determine by using an alternate lighting model that exploits knowledge of the human visual system (Image 3.7).

The standard Phong lighting model varies the luminance of the shading based on the angle between the normal and the light vector. This approach reduces the luminance in areas of the object which are not orientated towards the light source. This technique is an accurate approximation of real world lighting but when luminance is reduced, image

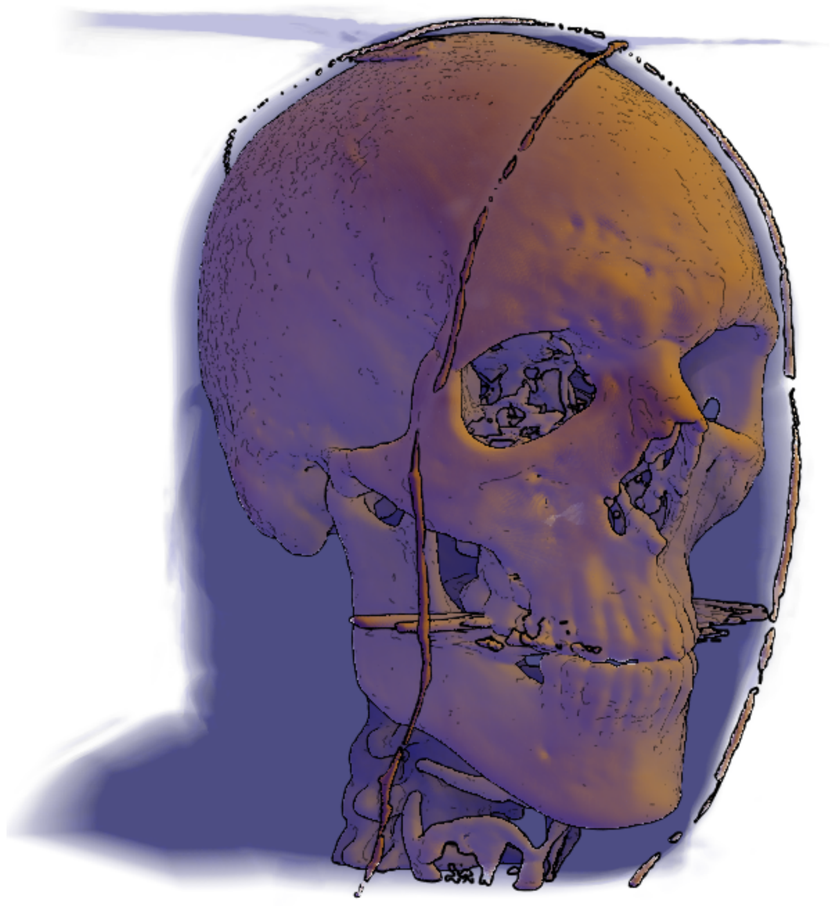


Figure 3.7: Tone shading combined with suggestive contours.

contrast is reduced and it can be harder to perceive surface details.

Gooch shading utilises two shading colours, a warm colour (red, yellow or orange) and a cold colour (blue, violet or green). The human visual system perceives cold colours as receding into the background and warm colours as coming into the foreground, and by carefully utilising these colours, user's perception of the structure of an object can be enhanced. Instead of reducing the luminance of the image based on the angle between the normal and the light vector, Gooch shading interpolates the colour between warm and cold tones. Areas that are pointing away from the light source are coloured using a cold tone, and areas that are facing the light are coloured using a warm tone (Equation 3.4). This allows the user to perceive the shape of an object accurately without reducing contrast in regions which are orientated away from the light source.

$$C = C_{cool}\left(\frac{1 + N \cdot L}{2}\right) + C_{warm}\left(1 - \frac{1 + N \cdot L}{2}\right) \tag{3.4}$$

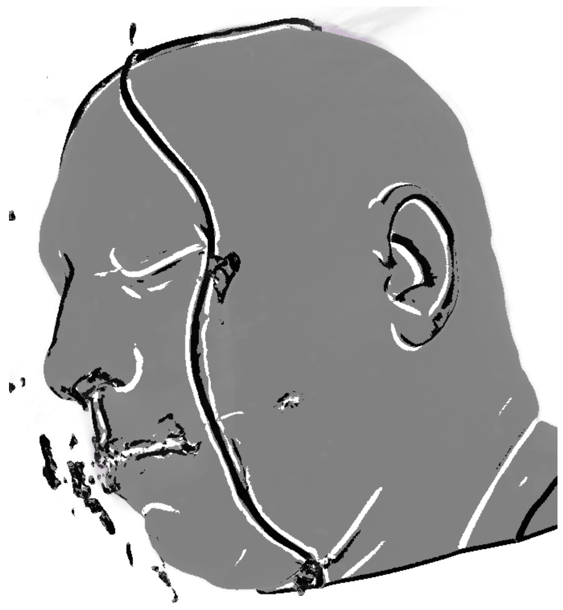


Figure 3.8: Ridge and valley lines.

3.2.3.2 Ridge and Valley Lines

Ridge and valley lines are object space lines which are drawn on the high and low points of a surface and are powerful shape descriptors used by numerous researchers to convey shape and curvature information [OBS04, IFP95, KWTM03, HSS+05]. They can be defined mathematically as the local maximum and minimum of principle curvature in the corresponding principle direction.

The method we use to calculate ridge and valley lines is based on the techniques by Kindlmann et al. [KWTM03] and a code listing is available in Appendix B.6. First the Hessian matrix H is constructed which is a 3x3 matrix containing the second partial derivatives of the volume data.

$$H = \begin{bmatrix} \partial^2 f/\partial x^2 & \partial^2 f/\partial x\partial y & \partial^2 f/\partial x\partial z \\ \partial^2 f/\partial x\partial y & \partial^2 f/\partial y^2 & \partial^2 f/\partial y\partial z \\ \partial^2 f/\partial x\partial z & \partial^2 f/\partial y\partial z & \partial^2 f/\partial z^2 \end{bmatrix} \tag{3.5}$$

Then a matrix P is constructed by combining the normalized gradient n and identity matrix I.

$$P = I - nn^T \tag{3.6}$$

P and H are combined together to create the matrix G called the Geometry Tensor.

$$G = \frac{-PHP}{|n|} \tag{3.7}$$

Finally the trace T and the Frobenius norm F of the geometry tensor are computed and used to calculate the two principle curvatures, k_1 and k_2.

$$k_1 = \frac{T + \sqrt{2F^2 - T^2}}{2} \quad k_2 = \frac{T - \sqrt{2F^2 - T^2}}{2} \tag{3.8}$$

The values of k_1 and k_2 are then used to determine the areas where ridges and valleys are present.

3.2.3.3 Cel Shading

Cel shading or cartoon shading is a type of NPR shading which attempts to create renderings which mimic the style of comics or cartoons. This technique was implemented as it was hypothesised that quantising the colour values of an object could alter object perception due to the increased colour contrast which results from using less shades. Cel shading maps lighting to discrete values, resulting in sharp transitions between colours as opposed to the smooth variation seen in photorealistic shading techniques.

Cel shading uses the same lighting equation as the Blinn-Phong model but modifies the intensity of the light contribution by limiting it to a number of discrete values, therefore, ensuring that large blocks of colour are displayed instead of smoothly varying lighting (Image 3.9). Equation 3.9 shows an example of this modification where n is the number of discrete values required and $\lfloor\rfloor$ represents the floor function. In addition, outlines along the objects contour lines are drawn by checking the angle between the normal and view vector. Our GLSL implementation is shown in Listing B.5.

$$\text{intensity} = \frac{\lfloor \text{intensity} \times n \rfloor}{n} \tag{3.9}$$

3.2.3.4 Volumetric Contours

Volumetric contours are a technique for viewing surface boundaries within volume data without explicitly specifying surface or isosurface locations [CMH+01].

The gradient magnitude of a voxel is used to measure how close a voxel is to a surface boundary. A high gradient magnitude means that the intensity value of the data is changing very quickly in the local area, and therefore there is a high likelihood that the voxel is located near a change in surface (Equation 3.10).

In order to reduce surface overlap, only the silhouettes of potential surfaces are displayed. This is accomplished by weighting voxels based on their normal direction.

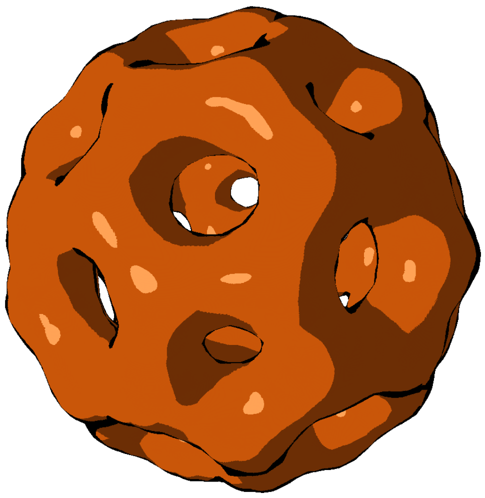


Figure 3.9: Volumetric cel shaded rendering of the bucky ball dataset.

Voxels with a normal perpendicular to the viewing direction are given more weighting, thus ensuring that voxels that are potential surface silhouettes are displayed (Equation 3.11).

These two weightings are multiplied together and the final result is the opacity of the current voxel (Equation 3.12). Variables a, b and c are user defined. a specifies the gradient magnitude value below which voxels are considered not to be part of a surface boundary and therefore not rendered. b specifies the gradient magnitude above which voxels are considered to be part of a surface boundary and therefore rendered opaque. Any voxels with a gradient magnitude between a and b have their transparency linearly interpolated. c is a user defined variable which controls how thin the silhouette lines are drawn. This technique ensures that voxels are opaque in regions where there is a high probability of a surface change and where a surface is perpendicular to the viewer and transparent at all other times. This results in a rendering (Image 3.10) which is similar to a silhouette line drawing but without the need to specify which isosurfaces to extract lines from.

$$\text{Surface Weighting} = \begin{cases} 0 & \|g\| < a \\ 1 & \|g\| > b \\ \frac{\|g\|-a}{b} & \text{Otherwise} \end{cases} \tag{3.10}$$

$$\text{Silhouette Weighting} = (1 - |N \cdot V|)^c \tag{3.11}$$

$$\text{Opacity} = \text{Surface Weighting} \times \text{Silhouette Weighting} \tag{3.12}$$

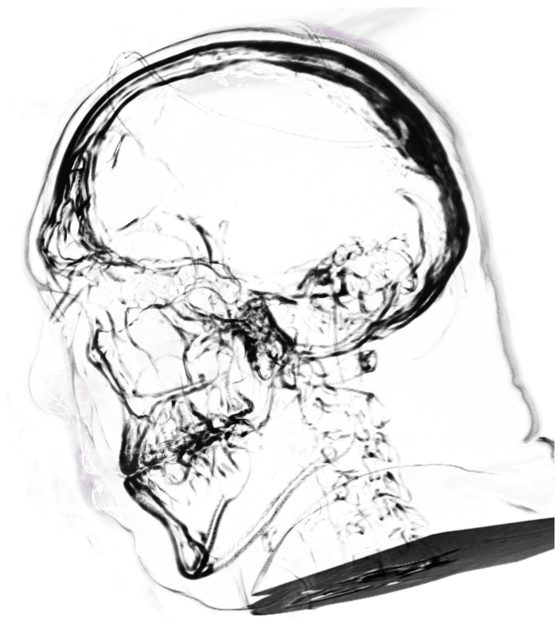


Figure 3.10: Volumetric contours.

3.3 Combined NPR and Focus and Context Framework

We have integrated our NPR techniques into a volume rendering system which combines CPU line extraction, two pass volume rendering, and GPU based NPR techniques in order to generate a final perceptually enhanced image (Figure 3.11). In this section, we provide a general overview of our implementation before explaining each aspect in detail. This framework was implemented in the early stages of this research where the main focus was on NPR techniques. As such, we choose to utilise an easy to use, slice based volume renderer in order to allow for rapid prototyping of the NPR techniques. In our later work, we utilise a more advanced and flexible raycast volume renderer.

3.3.1 Pipeline Overview

The first step of our implementation is to perform a pre-processing step on the CPU in which we calculate normal information.

The next step of our implementation performs two pass volume rendering. Using the normal information calculated in the previous step and by generating a custom transfer function based on the user's input, we are able to generate two separate images from this process: one of the user selected isosurface and one of the remaining volume. These two images are passed on to our GPU image shaders for further post-processing.

In the third stage of our pipeline, we extract object space lines directly from the

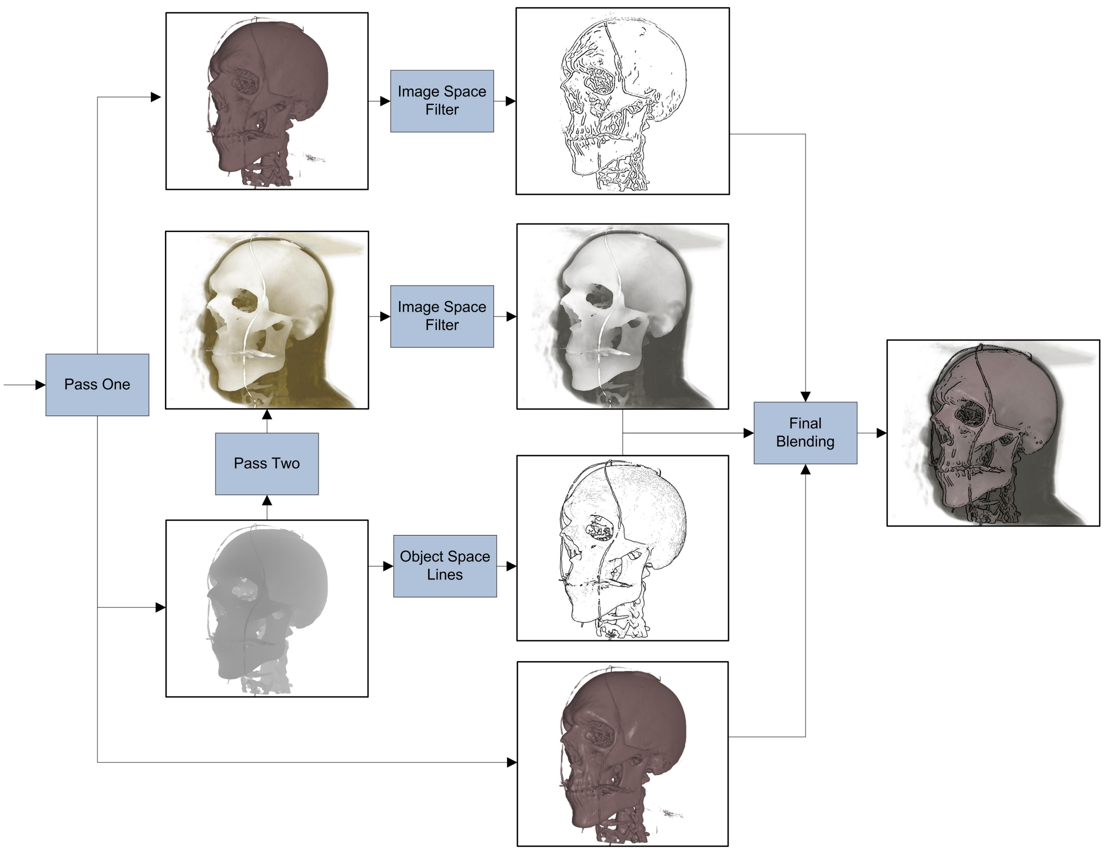


Figure 3.11: Pipeline overview.

volume data using an optimised and parallelised version of the algorithm proposed by Burns et al. [BKR$^+$05]. Our optimisations result in approximately a 50% increase in performance in comparison to the original implementation. As this stage is performed primarily on the CPU, it is a major bottleneck for performance and unfortunately even our optimised algorithm is unable to render both silhouettes and suggestive contours in real-time. One of the largest computational costs for the object space line drawing is in the line visibility calculations. We are able to completely eliminate this step by utilising the depth image generated by our volume rendering stage in order to depth cull the generated lines.

We dynamically change the complexity of the lines based on user interaction with the program. This is done by utilising the algorithm by Burns et al. which exchanges line drawing reliability for reduced computational costs. This algorithm is used by our system when the user is interacting with the volume dataset. When the scene is static, our optimised line drawing technique, as described in Section 3.3.4.2, is enabled which displays all possible lines. In contrast, the Burns et al. algorithm does not guarantee that all lines will be drawn.

The penultimate step of our implementation is the application of post-processing filters to the volume images, implemented through the use of GPU shaders. The types of filters we use fall into two general categories: edge extraction calculations applied to add detail to the target isosurface, and texture smoothing filters applied to abstract extraneous information from the surrounding volume.

The final step in our pipeline is to blend all the previously generated images together and display them to the screen.

3.3.2 Normal Calculation

We perform a pre-processing step which calculates the normal value for each voxel to a high accuracy using a 3D mask based on the Sobel filter. This calculation is performed on the CPU and the resulting information is stored for use by the CPU line drawing calculation and also converted into a 3D texture and uploaded to the GPU to be used in shading calculations. By performing this step as a pre-process and avoiding repeated GPU calculations every frame we were able to significantly increase the performance of our rendering system.

3.3.3 Volume Rendering

In order to apply different abstractions to the two levels of the volume, we modified a standard 3D texture slice volume renderer to output two images: one image of the areas of the volume that the user has chosen to focus on, and one image of the remainder of the volume. We modified the basic algorithm so that the different segments of the volume are rendered in separate passes, enabling us to use custom rendering parameters for each segment in the volume. We further modify the output images from each pass by applying various image space filters as a post-processing step. In a final stage, the two images are blended together and rendered to the screen resulting in a complete volume view. If no modifications are performed on the intermediary images, the resulting volume rendering is visually identical to a traditional one-pass render. Our two pass modification incurs a 20% computation penalty in comparison to an unmodified volume render but allows custom rendering parameters and post-processes to be applied separately to each volume segment.

3.3.3.1 First Pass

Our first pass is used to generate a solid rendering of the user selected isosurface. We flag voxels which are part of the user selected isosurface and render these voxels as normal. The pixel shader utilises multiple render targets in order to render several different images in one pass. We generate an image using Phong lighting rendering parameters, an image using Phong lighting without any specular terms, and also a depth image.

It was found that the majority of image space edge filters used to perform perceptual enhancement gave optimum results when using a volume rendering with only ambient and diffuse lighting, however, the addition of specular highlights produces a more aesthetically pleasing final image. Thus, we render both images and use the rendering with specular highlights for the final blending step while using the diffusely lit image for image space line extraction.

The depth image is used to perform depth culling for both the object space lines and the second volume rendering pass.

3.3.3.2 Second Pass

The second pass is used to render the remaining sections of the volume. The naive approach would be to render all parts of the volume that were not displayed previously. This follows the same approach as used by a normal single pass renderer, however, it neglects to deal with occlusions that would normally be caused by the target isosurface.

The isosurface rendered in the first pass would normally occlude various parts of the rest of the volume, however, since we are not rendering this isosurface in the second pass, no occlusion calculations occur, and areas of the volume are rendered when they should not be.

It is possible to avoid this problem by using a custom pixel shader which uses the depth buffer generated in the first pass to perform depth testing. This discards any voxels which would normally be occluded and results in the correct final image.

3.3.4 Non-Photorealistic Rendering Additions

In this final step of the pipeline we calculate the NPR stylisations which are going to be applied to the volume rendering.

3.3.4.1 Image Space NPR

As we mentioned previously, most of the image space NPR techniques work best when they use a diffusely lit image as their input. In order to facilitate this, but still incorporate specular highlights into our volume rendering, the first pass of our system outputs two rendered images. One image contains only the diffusely lit target isosurface, and the second image contains both diffuse and specular highlights. The specular image is used in the final blending step and the diffusely lit image is used as input into the image space NPR algorithms. The output of the NPR algorithms is written to a render target and saved for later use in the blending stage.

3.3.4.2 Object Space NPR

We propose a number of methods to extend and optimise the original contour and suggestive contour extraction algorithm of Burns et al. These techniques result in approximately a 50% improvement in performance speed and allow interactive extraction of line drawings from the full volume. We enable this through a CPU based multi-core implementation that utilises atomic instructions and exploits locality of reference.

The basic idea of our approach parallelises the original algorithm of Burns et al. by splitting each cube into a discrete work packet which is calculated using its own thread. This naive approach does indeed provide a performance increase, but we propose a number of modifications which further improve upon this technique.

A problem that frequently occurs in highly parallel algorithms is that the overhead of creating and destroying threads can be significant in comparison to the actual work that each thread performs. Instead of creating a thread for each cube and allowing the operating system to manage thread scheduling, we create a finite number of threads and allow them to work on multiple different cubes. We employed a thread pool structure that allows threads to continually pull work jobs from a centralised queue, thus eliminating the overhead of thread creation and destruction.

This technique requires that only one thread is able to access the queue structure at the same time to ensure safe results. This requires that a semaphore structure be implemented which unfortunately results in a bottleneck as threads are busy waiting to access the queue. A software based semaphore was implemented which ensured correct access to the queue structure but resulted in a large performance penalty. Through the use of hardware supported x86 atomic instructions, it was possible to create a faster implementation which used inbuilt CPU instructions resulting in a large performance increase.

Our final optimisation was to exploit locality of reference in order to speed up line extraction, by exploiting the fact that neighbouring cubes contain duplicate voxel information, since each cube contains information for the surrounding eight voxels as shown in Figure 3.12. Due to the nature of our threading implementation there was no guarantee that neighbouring cubes would perform calculations at similar times which results in new voxel data being loaded onto the CPU with every work packet. Ideally, we want the calculation for neighbouring cubes to be performed close enough together that the CPU cache still contains duplicate information and therefore speeds up the calculation. We ensure this happens by increasing the size of each work packet to contain a block of neighbouring cubes, rather than a single cube, resulting in increased cache hits. The number of cubes in each work packet needs to be chosen carefully as too small a number reduces locality of reference and too high a number reduces parallelism, but careful manipulation can ensure a performance increase. Code for the thread work loop can be seen in Listing 3.1.

The original version of the line extraction algorithm proposed by Burns et al. only analysed a subset of the volume data and was therefore unable to extract all possible lines. Our multi-core optimisation improves on this technique by analysing the complete volume data, ensuring that no lines are missed, while at the same time improving performance by 50% (Table 3.1). In addition, the original implementation uses a computationally expensive occlusion calculation in order to remove hidden lines. Since we are integrating the line drawing calculation into our volume rendering system we can take advantage of the hardware supported occlusion culling available on the GPU. In the first pass of the volume rendering, we output a depth buffer image which contains the depth of the target isosurface. During the rendering of the object space line drawing, we

Listing 3.1: Multi-core optimisations.

```
//thread work loop
void VolTP_C::run(){
  //get first job - atomic increment
  unsigned int jobNumber = InterlockedIncrement(&m_nextJob);

  //keep looping while there's still jobs left
  while (jobNumber < m_task_list.size()){
    vec3 temp = m_task_list[jobNumber];

    //perform calculations for every valid cube in the work packet
    //valid cubes range from (m_task_list[jobNumber], m_task_list[jobNumber] + ↩
        m_grouping)
    for(int x = temp.x; x < (temp.x + m_grouping); x++){
      for(int y = temp.y; y < (temp.y + m_grouping); y++){
        for(int z = temp.z; z < (temp.z + m_grouping); z++){
          //perform line extraction calculation
          t_checkCell_C(x, y, z);
        }
      }
    }
    //fetch next job - atomic increment
    jobNumber = InterlockedIncrement(&m_nextJob);
  }

  pthread_exit(NULL);
}
```

Table 3.1: Performance increase from our multi-core optimisations.

Dataset	Performance Increase
Skull	67%
Foot	54%
Brain	40%

upload the previously calculated depth buffer to the GPU and make use of the inbuilt depth testing hardware. This results in automatic occlusion culling of the line drawings without the manual calculation required by the initial implementation.

3.3.5 Blending

In order to combine the different aspects of the rendering system together into a final image, it is required to blend the outputs of the different rendering stages. The final output of each component of the system is a texture image with both colour and transparency information. We can take advantage of the GPU's fixed function hardware to combine these images for minimal cost. This is accomplished by rendering multiple view aligned quads positioned so that they exactly fill the viewport. By texturing these quads with the generated images we can utilise the GPU's fixed function pipeline to perform blending

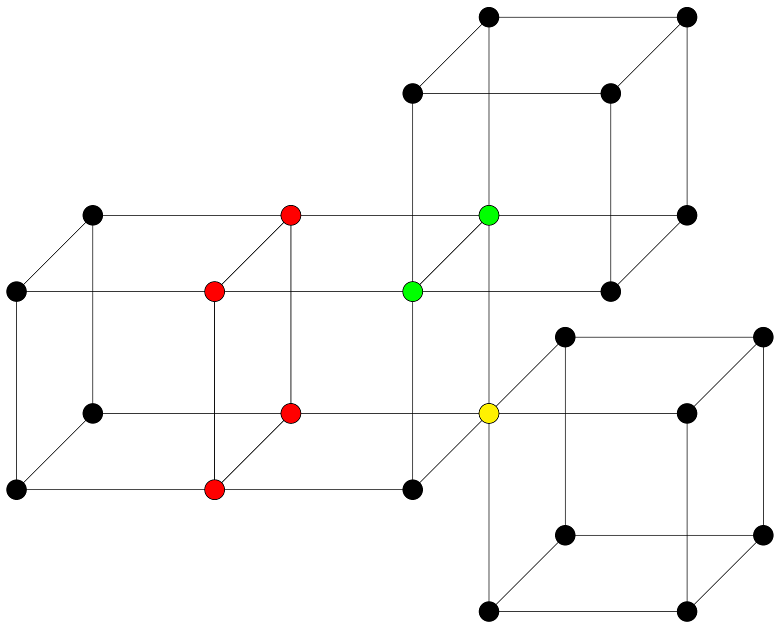
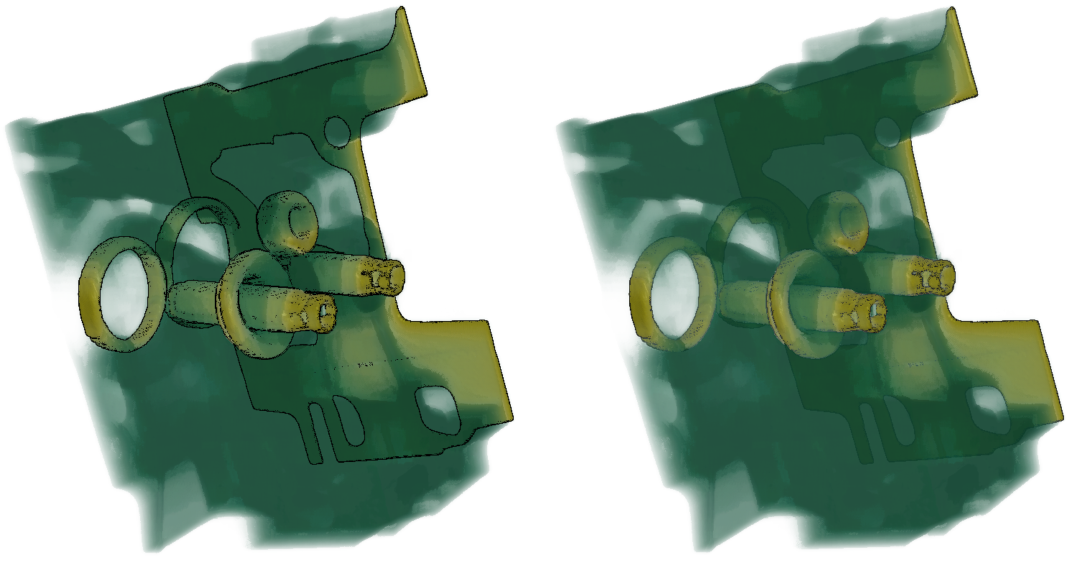


Figure 3.12: The central cube shares four voxels with the cube on its x axis (red), two voxels with the cube on its xy axis (green) and one voxel with the cube on its xyz axis (yellow).

and compositing between the images.

By blending the NPR focus enhancements at the final stage in the process we ensure the line drawings are not occluded and stand out from the image ensuring optimal viewing. If we were to blend the NPR enhancements between the focus and context rendering, the context information would obscure the enhancements, potentially diminishing their effectiveness (Figure 3.13).



(a) Blending order: Focus, NPR Enhancements, Context.

(b) Blending order: Focus, Context, NPR Enhancements.

Figure 3.13: An example of how blending order can affect NPR enhancements.

3.4 Evaluation

We devised two studies in order to test how our focus and context NPR technique affects a user's perception of volume rendering. Firstly, we used an automatic metric to investigate whether our technique can affect colour contrast and image saliency. The results of this metric were promising, but we wished to confirm whether these results would carry over into real world tests. Therefore, we expanded on this automatic metric by performing user studies, involving live participants, to investigate if our focus and context method can improves a user's perception of surface shape.

3.4.1 Saliency Tests

An automatic saliency metric [IKN98] was used to evaluate whether adding stylisation can increase the saliency of important elements within the scene. The metric uses a multi-scale approach for finding the location of salient regions based on local image structure. Regions of high contrast are found for colour, intensity, and orientation of an image on a number of scales. The results are then normalised and summed to create the final saliency map for an image.

A number of images were taken from the system using the various NPR styles, each was compared with unstylised renderings. Each set of images was taken from identical viewpoints. The images were then tested using the saliency metric and while results were

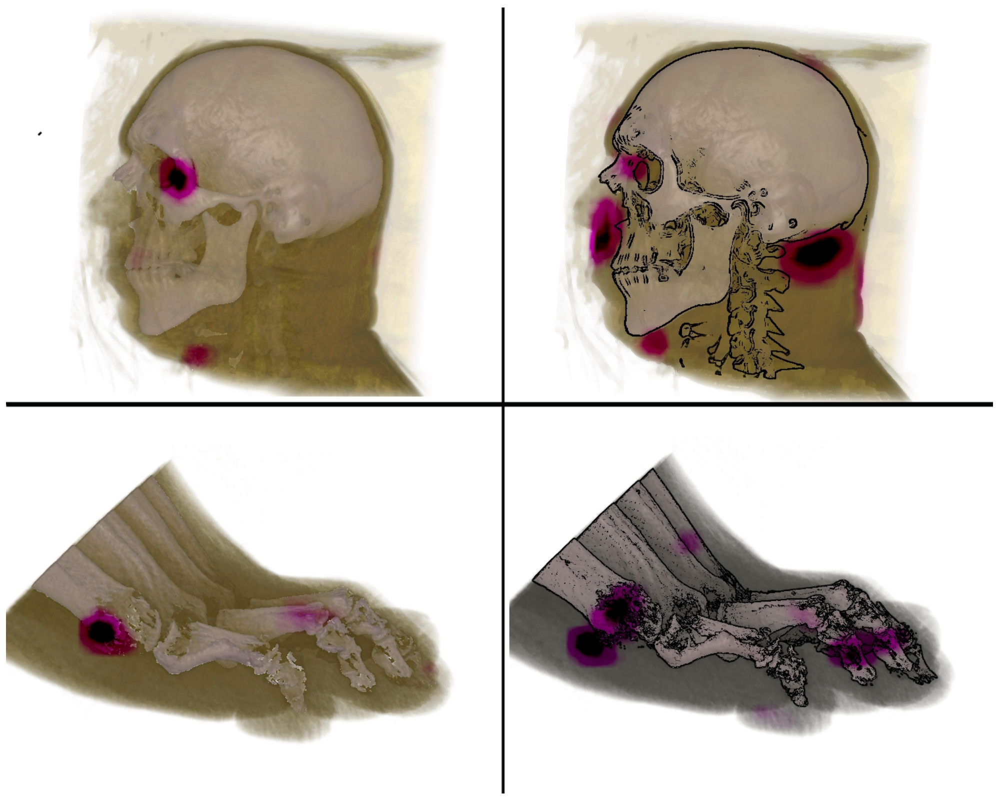


Figure 3.14: Examples of how adding stylisation can affect colour contrast within scenes. The left images are normally rendered while the right images use the Sobel filter (top) and a combination of suggestive contours and saturation variation (bottom). Colour contrast is highlighted and measured using the automatic saliency metric presented by Itti et al. [IKN98]. These images show how the saliency of the target isosurface can be increased and manipulated using various styles.

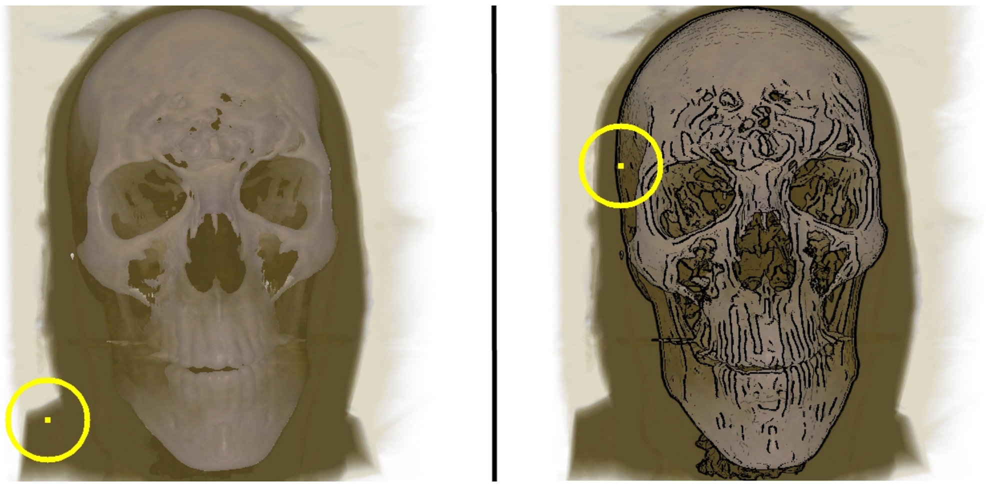


Figure 3.15: Example of how using stylisations can change the most salient point in the image, marked with a circle in each image. Normal rendering (left) and Canny (right). As can be seen, adding stylisation can change the most salient point to the target isosurface.

somewhat mixed, it was seen that by adding stylisation to the images, the saliency of the isosurface within the image can be increased while the impact of the surrounding volume can be reduced. This is especially true for increasing colour contrast within images using edge detectors, as can be seen in Figure 3.14. There was no large difference in how the edge detectors performed against each other although suggestive contours increased the saliency within the centre of the isosurface more than others. Figure 3.15 shows that most salient point of each image (marked by a circle) can be changed using these stylisations.

These results from the metric show that by adding stylisation, visual saliency within scenes, and therefore user eye-gaze behaviour, can be manipulated. Previous authors [CDF+06, MBG08] have shown the benefits of gaze direction in emphasizing details in computer imagery. While the automatic metric is based on a low-level model of visual interest, the results show that the stylisations used could make scenes faster to comprehend as eye gaze is drawn to the most important parts of the scene.

3.4.2 User Experiment

An experiment was performed to investigate how each style affected a users ability to determine the shape of the target isosurface. The primary aim of focus and context visualisations is to give a clear impression of a certain part of the dataset while showing the rest of the object data for reference. For this reason, it was necessary to test how well each style represented each dataset which meant a shape evaluation experiment was the most suitable test, and it was chosen over task-based or eye-tracking experiments.

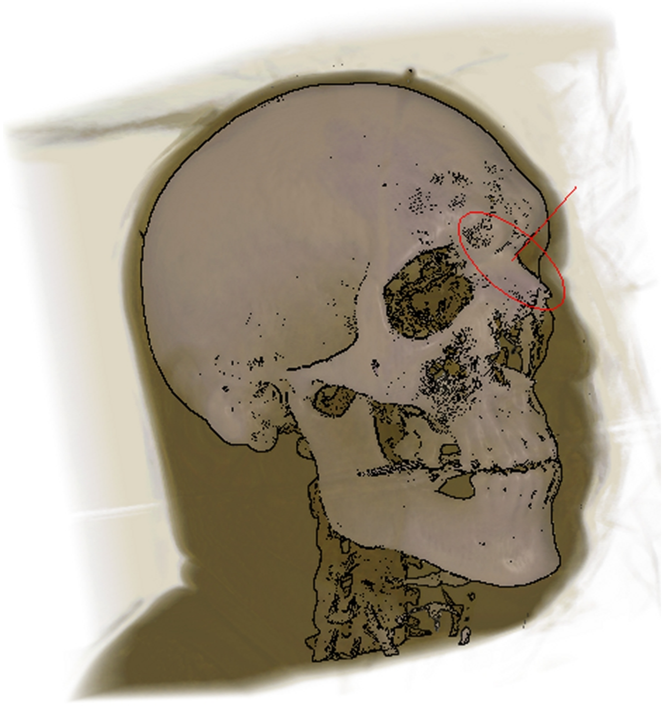


Figure 3.16: An example image shown to participants in the user study. The controllable gauge can be seen in red.

We hypothesised that by adding edges and abstractions to the dataset, the background volume would be de-emphasized and the target isosurface level would be clearer to the user, therefore, increasing user's understanding of the dataset.

3.4.2.1 Experiment Setup

To test a user's perception of shape, an experiment was run which involved placement of gauges on static images. This protocol was described in previous experiments [KvDK92, CSD+09]. Participants were shown a series of images from the system and asked to rotate gauges which overlaid the images to match the surface normal of the isosurface at that point. Participants had no control over gauge position, only alignment. The alignment of the gauge was controlled by the mouse and the space bar was used to indicate that the participant was happy with the gauge position and ready to move on to the next trial. Each gauge was drawn as an ellipse and a single line as seen in Figure 3.16.

Gauge placement was pre-determined and each gauge was placed in an area of interest for each dataset. Three diverse datasets were used in the experiment. These were scans of a male head, a brain, and a foot. The scale of each dataset was constant, although the rotation of the dataset was changed slightly to avoid any learning curve with regards to the normals. Each image was shown 8 times with different gauge placements in each trial to fully test a user's shape perception with each style. This resulted in a total of 360 trials. 14 naive participants took part in the experiment (10M-4F), each with a knowledge of graphics and therefore surface normals.

There were 15 styles tested on each dataset, which resulted in 45 images, a sample of these images can be seen in Figure 3.17. These styles can be seen listed below. Each of the 2 object space line drawing, 3 image space edge detection and 2 abstraction techniques were tested individually on each dataset. It was also tested how combining the image space and object space edges would affect user estimations, which resulted in 6 additional styles. Also tested was a style containing a combination of suggestive contours, Difference of Gaussian edges and the Kuwahara painterly effect. This style was added to determine how using 3 different stylisation types together could affect user perception of an image. Finally, a set of images using the focus and context volume rendering system was run with no stylisation at all as a baseline comparison. The focus and context rendering was chosen as a baseline instead of a single shaded isosurface as our focus is on investigating the effectiveness of different NPR techniques in order to emphasise and abstract data. A single isosurface only has one area of interest and does not allow for both emphasis and abstraction to be applied at the same time.

No Style Normal rendering

Style 1 Sobel edge detection

Style 2 Difference of Gaussian (DoG) edge detection

Style 3 Canny edge detection

Style 4 Silhouettes

Style 5 Suggestive contours

Style 6 Sobel / Silhouettes

Style 7 DoG / Silhouettes

Style 8 Canny / Silhouettes

Style 9 Sobel / Suggestive contours

Style 10 DoG / Suggestive contours

Style 11 Canny / Suggestive contours

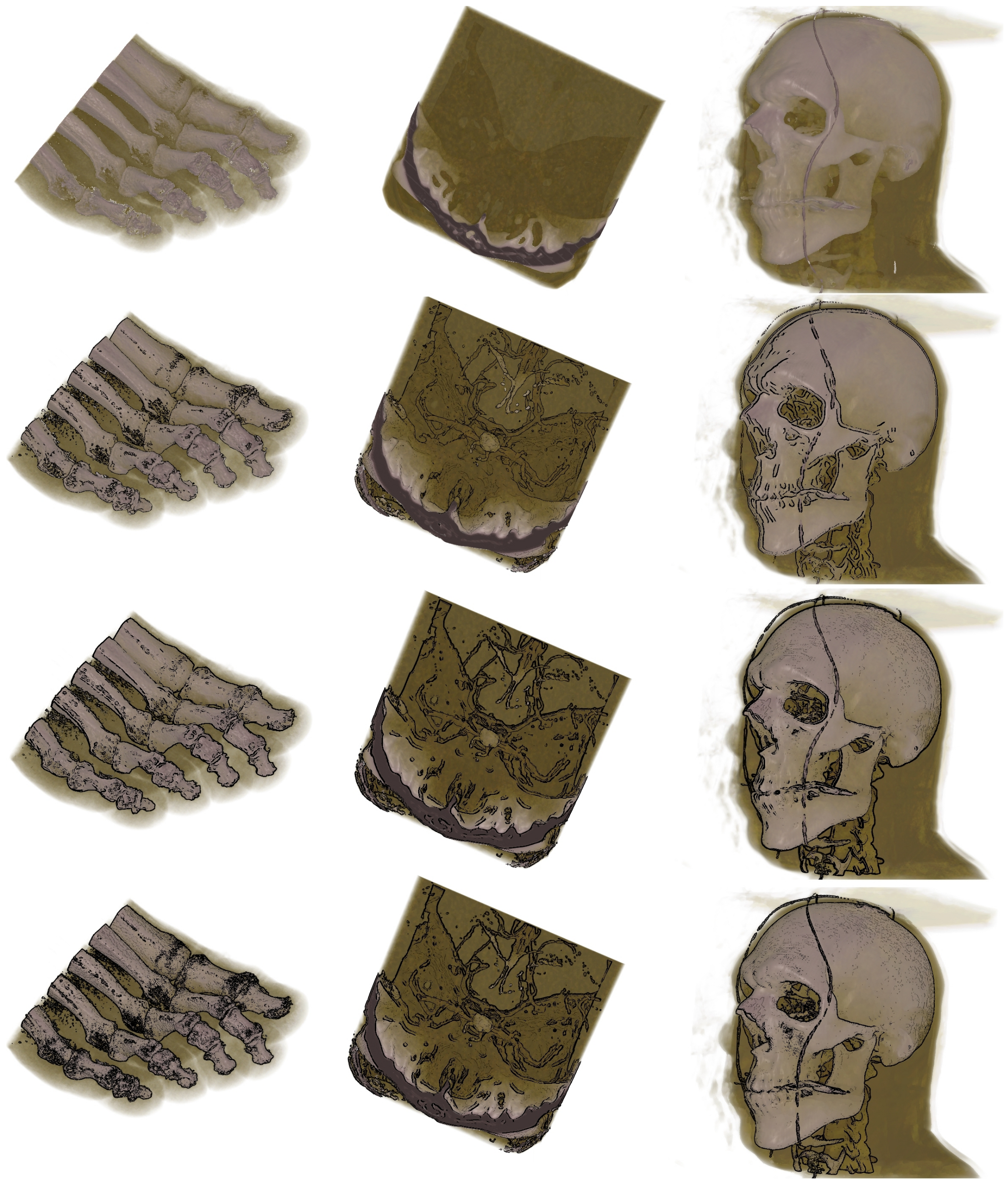


Figure 3.17: Some sample images from our system: First Row - normal rendering; Second Row - silhouettes, suggestive contours, Canny; Third Row - Sobel, Canny combined with silhouettes, Sobel combined with suggestive contours; Fourth Row - Difference of Gaussian combined with suggestive contours, Canny combined with suggestive contours, Difference of Gaussian combined with suggestive contours and Kuwahara filtering.

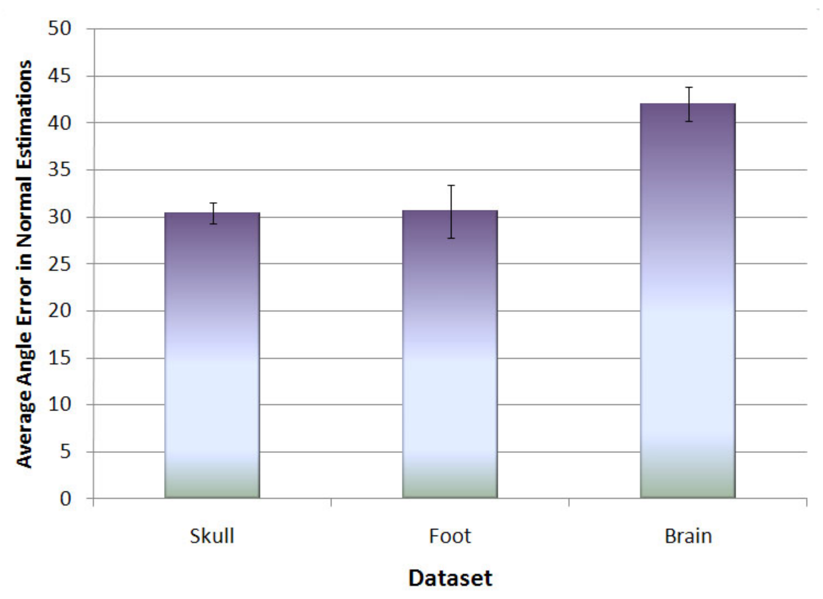


Figure 3.18: Average normal estimation errors for each dataset. Error bars represent standard errors of the mean in all graphs.

Style 12 DoG / Suggestive contours / Kuwahara filter

Style 13 Kuwahara filter

Style 14 Saturation variation

3.4.2.2 Results and Analysis

Dataset For each trial, the angle between the correct surface normal and the user estimated surface normal was calculated. The average angle was then calculated for each style, across each dataset, and an ANalysis Of VAriance (ANOVA) was performed on the results where the conditions were *Dataset*(3) and *Style*(15). It was found that there was a main effect of dataset where the type of dataset shown affected user accuracy ($F(2, 26) = 15.253$, $p < 0.0005$). Post-hoc analysis was then performed using a standard Newman-Keuls test for pairwise comparison among means. As can be seen from Figure 3.18, error was significantly higher for the brain dataset than the skull or foot datasets ($p < 0.0003$ in both cases). Several participants noted, after the experiment, that they had prior knowledge of what shape a skull or foot skeleton should be, whereas they had no knowledge of the localised shape of a brain dataset. Therefore, we hypothesise that this previous knowledge affected the experiment results.

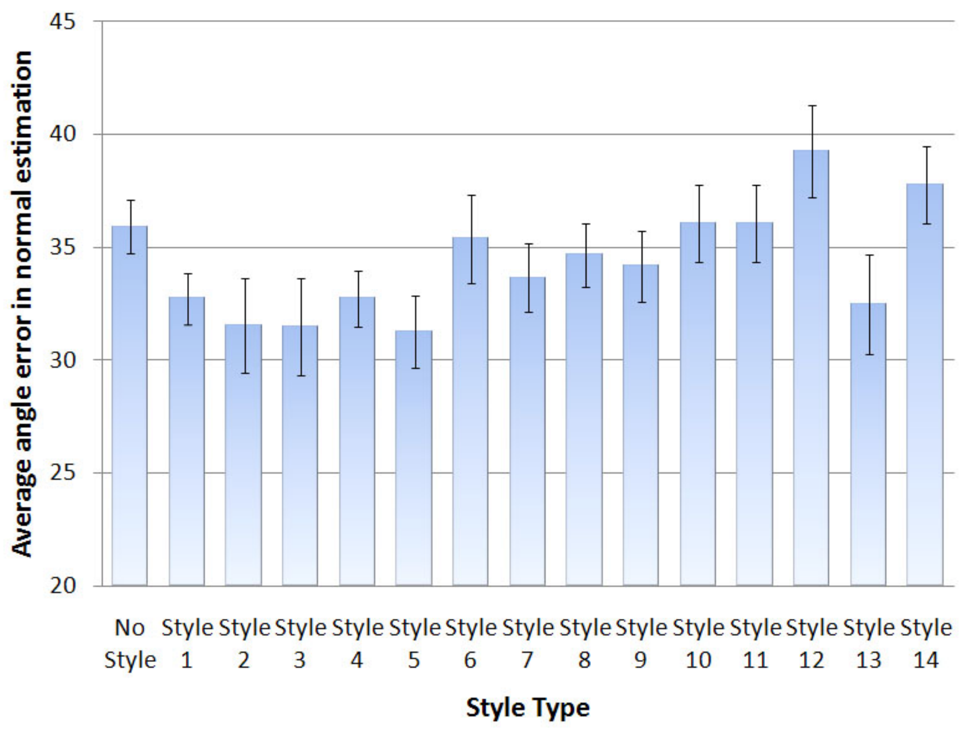


Figure 3.19: Average normal estimation errors for each style type over all datasets.

Style It was also seen that there was a main effect of style ($F(14, 182) = 7.6138, p < 0.00001$), where certain styles performed significantly better than others. Post-hoc analysis was performed on the data using a Newman-Keuls test, and it was seen that the angle estimations from images with Difference of Gaussian edges, Canny edges ($p < 0.015$ in both cases) and suggestive contours ($p < 0.007$) were significantly better than those from renderings with no stylisation. These differences can be seen in Figure 3.19. Results also show that estimations from the images which used either basic silhouettes or Sobel edges did not have any significant difference than those from the unstylised renderings. This was somewhat expected, as Sobel edge detection is a far simpler and faster image space technique than either Difference of Gaussian or Canny edge detection. Similarly, object space silhouette detection is a much faster method than suggestive contours. These results imply that speed cannot be traded off if reliable and effective results are to be obtained. It was also found there was an interaction between dataset and style ($F(28, 364) = 13.576, p < 0.00001$) which indicates that certain styles performed better within certain datasets. To investigate this effect, the results from each style were isolated and compared. While it was found that there was no change in the styles which performed well for the foot and skull datasets, a number of styles performed significantly better than the normal rendering in the brain dataset. As Figure 3.20 illustrates, all styles apart

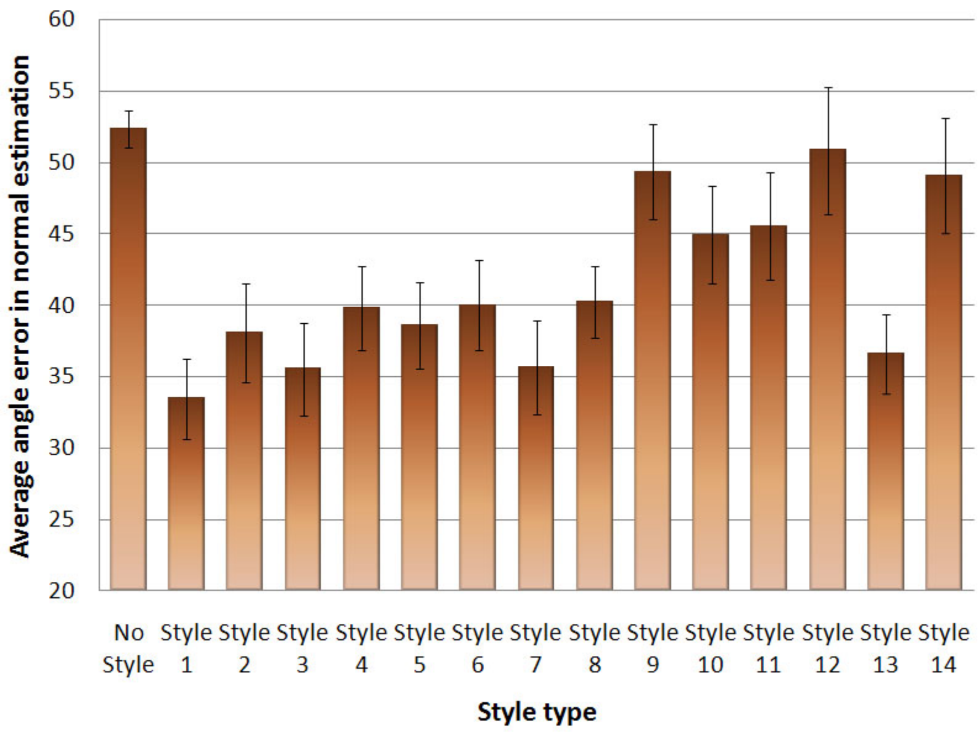


Figure 3.20: Average normal estimation errors for each style type for the brain dataset.

from the mixed style (Style 12), saturation variation, and the combination of Sobel edges and suggestive contours performed significantly better than normal rendering in this dataset ($p < 0.02$ in all cases). Within the brain dataset, there was much occlusion of the isosurface from the background volume data, this meant there were less cues for shape than in the other datasets, and the styles were therefore more effective in conveying shape. Also, as mentioned earlier, participants were less familiar with the shape of the brain dataset so the cues added made more of a difference than in the other datasets.

Normal Alignment It was noted that there was an effect of normal alignment in the experiments. It was found that if the surface normal was less than 15 degrees from the camera view vector then user accuracy was significantly increased. If the angle between the surface normal and the camera view vector was above 60 degrees then gauge estimation was significantly harder. There was no significant difference in accuracy if the normal was between 15 and 60 degrees, which represented approximately 80 percent of the gauges tested in the experiment. Within the experiments, the normals tested were distributed randomly which means that more experiments would be necessary to determine what effect, if any, normal alignment has on how each style performed. This is an interesting result and it is unexpected as such an effect was not noted in previous

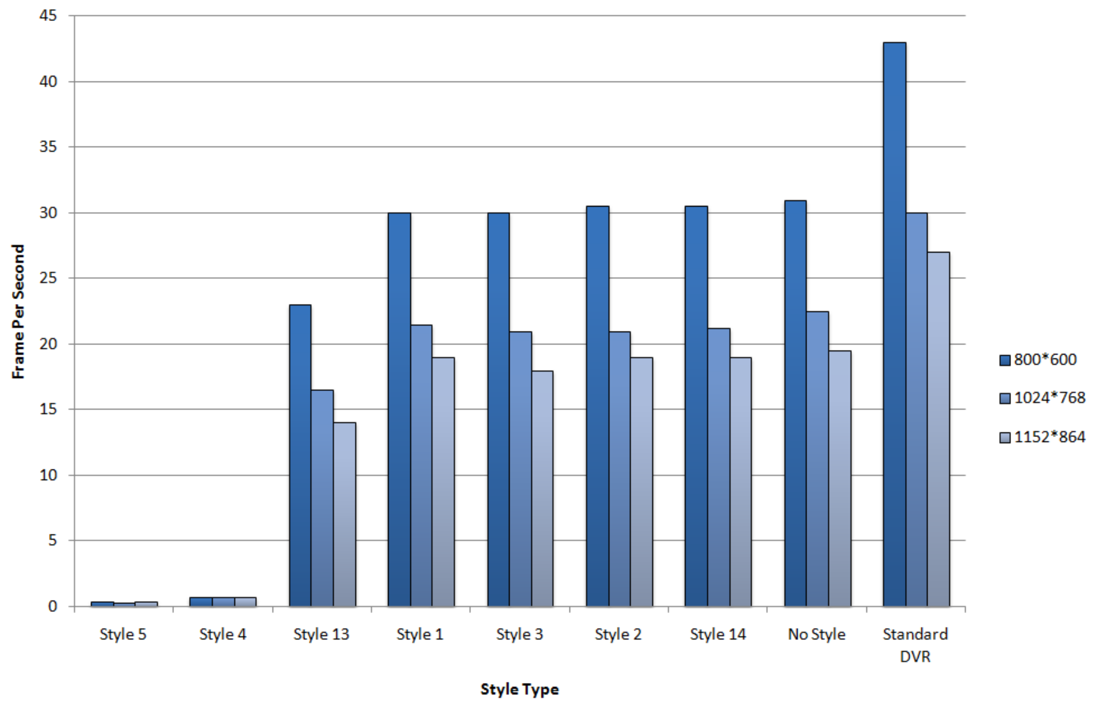


Figure 3.21: Average frames per second for the brain dataset at varying resolutions and styles.

work which used the same system [CSD+09].

Performance Figure 3.21 shows the average frames per second for the brain dataset at varying resolutions and styles. Our two pass modification incurs a computation penalty of between 16-40% in comparison to an unmodified volume render but allows custom rendering parameters and post-processes to be applied separately to each volume segment. Once the volume has been segmented, applying the majority of the image space NPR filters is negligible, however, both the object space lines and Kuwahara filtering incur a relatively large performance penalty.

3.5 Discussion

We have implemented a variety of NPR techniques both on the CPU and on the GPU through the use of programmable shaders. These techniques have been optimised for a volume rendering framework and for focus and context rendering.

We have presented a framework which integrates DVR and NPR components into a focus and context volume rendering system for various types of datasets. The DVR system is implemented on the GPU and employs parallelism to provide improved rendering speed for interactive visualisation. The NPR components include object space silhouettes and suggestive contours, implemented on the CPU, as well as image space edge detection and object space NPR, implemented as shaders on the GPU. A two-pass mechanism in the DVR system generates two separate images that are variably abstracted, post-processed and then merged to create a combined rendering featuring the various stylistic enhancements.

We carried out a number of experiments that show that our focus and context NPR technique is capable of increasing object saliency and improving user understanding of shape whilst preserving context with the surrounding peripheral data. Results from the experiments showed that certain edge detectors can have a significant effect on a users perception of object shape. We have shown that combining multiple edge detectors does not result in significantly better results than when the edge detectors are applied individually. It was also shown that the techniques described here are especially effective for datasets which contain a large amount of peripheral volume data, and also for datasets which users are not familiar with.

Chapter 4

Rendering Speed

4.1 Introduction

The size and complexity of volumetric datasets continues to grow, and to date graphics hardware has failed to keep pace with this increased demand for rendering power. Interactive rendering of high quality volume datasets is a challenging problem and, increasingly, researchers have investigated methods to accelerate the speed of volume rendering. Methods that have been explored include: dataset compression [LLYM04], early ray termination [KW03], volume bricking [HSS^{+}05], and many more.

Once fast rendering methods have been employed it is important to ensure that performance does not degrade below interactive rates. As the user interacts with a volume, the amount of data being displayed can change dramatically, and techniques to ensure that performance is maintained throughout user interactions are important. In this chapter, we discuss techniques to ensure interactive performance for focus and context volume rendering, enabled through the use of dynamic sampling rate adjustment coupled with adaptive resolution modifications.

In addition, by exploiting the theory of temporal coherency we are able to expand on this dynamic sampling technique in order to produce a system which increases the rendering quality of focus and context volume rendering. Our technique utilises information from the previously rendered frame in order to guide rendering in the current frame and results in a 20%–60% increase in sampling rate for the same rendering speed. By coupling these two techniques together, we have developed a framework which is able to both improve the rendering speed of focus and context DVR and dynamically adjust the rendering quality to maintain interactive performance, despite dramatic changes in rendering complexity. This contrasts with traditional volume rendering techniques which use a fixed sampling rate and require manual sampling rate adjustment when frame rates change.

4.2 Time-Critical Rendering

Time-critical rendering is a method of dynamically modifying the quality of a rendering in order to achieve a target frame rate. It is used in a wide variety of fields in order to ensure that tasks are computed in a reasonable time frame. This technique of dynamically modifying the rendering quality to achieve a desired frame rate is especially useful in volume rendering due to the view dependant nature of modern volume rendering. The rendering time for a volume dataset can vary by orders of magnitude when displayed from different views due to the view dependent nature of modern optimisations such as early ray termination. This means that a rendering quality which initially produced interactive rendering times needs to be continuously updated by the user in order to maintain interactivity as rendering time changes. By implementing time-critical rendering, through the use of dynamic sampling rate modification, this continuous manual tweaking of the sampling rate can be eliminated.

Time-critical computation for volume rendering has been proposed under another name by Bruckner [Bru04], nevertheless, we present our work in this area as we examine in detail how quickly time-critical rendering can adapt to the large changes in computation demanded by volume rendering and also as it is a precursor to our later work on image space adaptive rendering.

4.2.1 Framework

The rendering time for individual frames of a volume dataset may vary significantly depending on various view dependent factors. However, for a sufficiently high frame rate, localised frames should be broadly alike and therefore have similar rendering times. This allows us to use knowledge of the previous frames rendering time and sampling rate in order to tailor the sampling rate of the next frame in order to achieve the target rendering time.

By linearly interpolating the sampling rate of the previous frame between the previous frames render time and the desired render time we can automatically vary the sampling rate in order to ensure a constant frame rate. Equation 4.1 shows the calculation we perform to determine the new sampling rate where S_n is the new sampling rate, S_{n-1} is the previous frames sampling rate, f_{n-1} is the frames per second of the previous frame

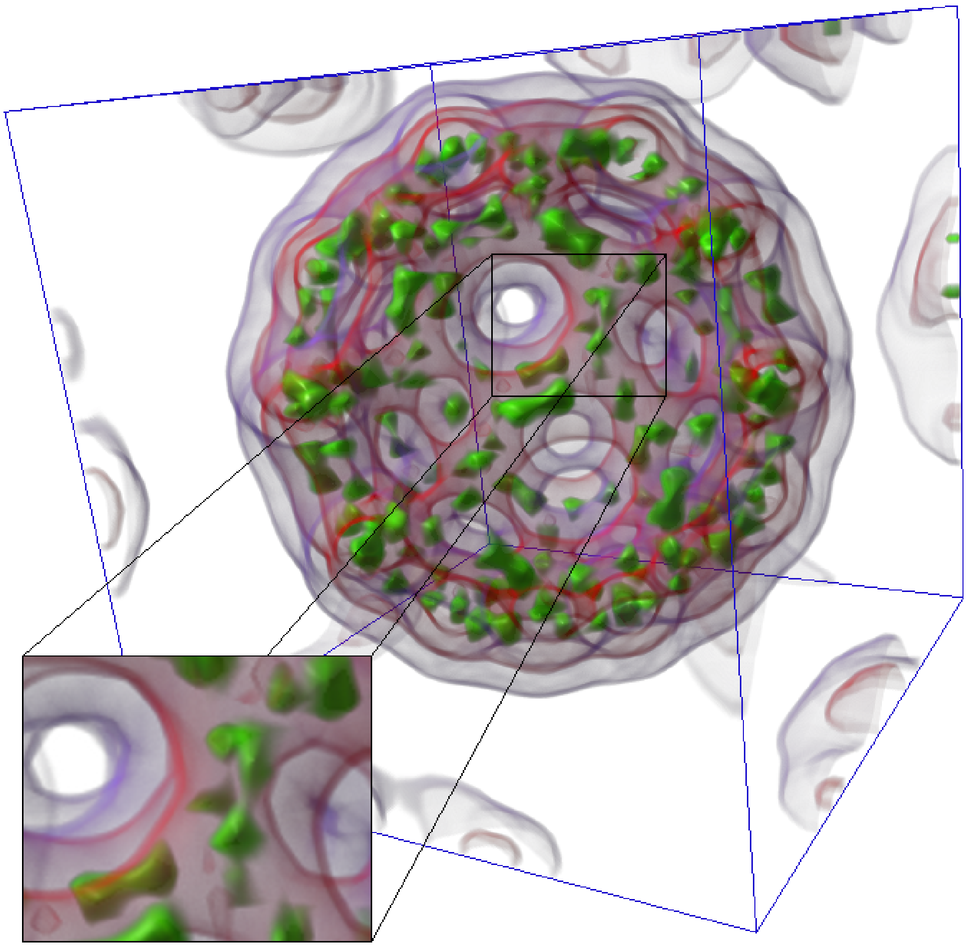


Figure 4.1: A screenshot from our implementation with time-critical sampling enabled. Our implementation has identified extra computation cycles and increased the sampling rate in order to generate a higher quality image while still maintaining the target frame rate.

and f_{target} is the user selected target frame rate.

$$S_n = S_{n-1}\left(\frac{f_{n-1}}{f_{target}}\right) \tag{4.1}$$

4.2.2 Windows/OpenGL Implementation

We implemented time-critical rendering for volume data on Windows using the OpenGL graphics system. We were able to take advantage of some of the features in the Windows and OpenGL API's in order to optimise our approach. Our implementation uses a high precision CPU timer, available through the Windows API, in order to accurately account for how long it takes to render each frame and is implemented in OpenGL

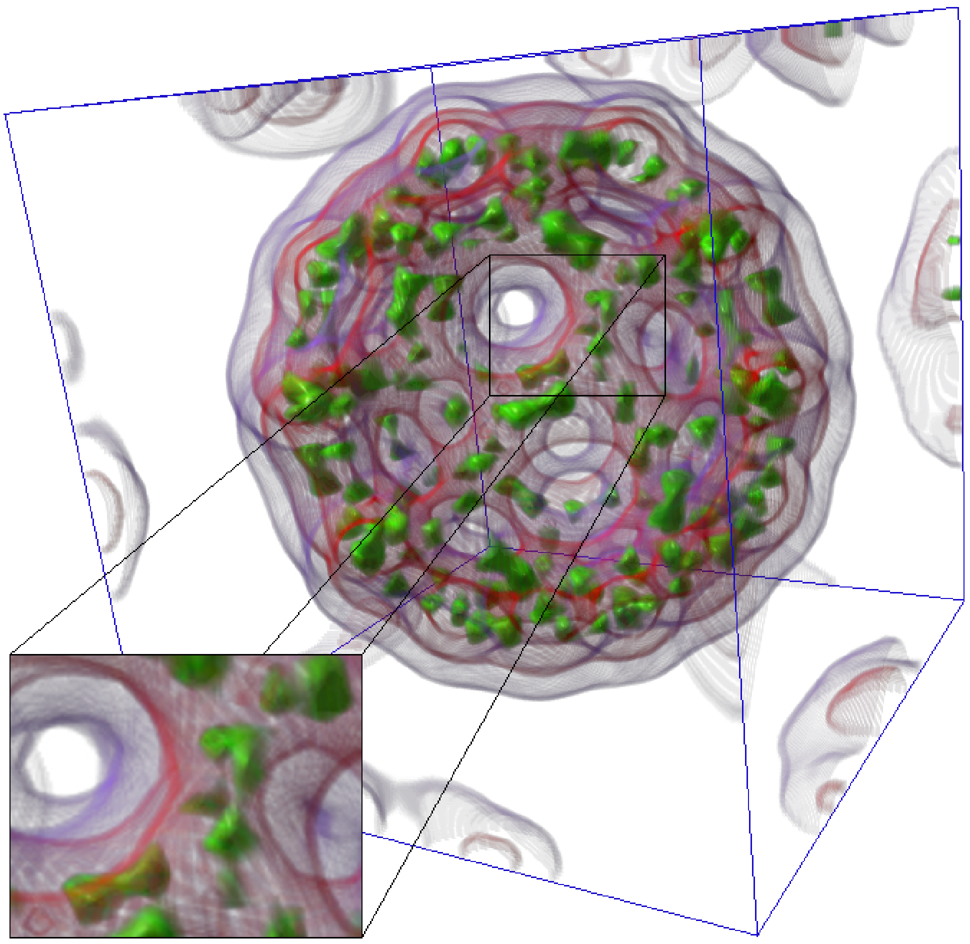


Figure 4.2: A screenshot from a standard volume rendering implementation. Notice how the quality of the image remains poor even though the frame rate has increased beyond the original frame rate selected by the user. The user had originally selected a sampling rate which provided interactive frame rates at low quality, but when the view was zoomed further out this reduced sampling rate failed to take advantage of the spare computational cycles available.

using the Voreen volume rendering engine [MSRMH09]. However, we need to take account of the buffered nature of the OpenGL API which allows the graphics system to buffer commands before executing them. We need to obtain an accurate and high precision timing of how long it took to generate each individual frame, but OpenGL by default does not guarantee that each command will be executed immediately. In order to circumvent this, we need to force the OpenGL graphics system to clear all its buffers and execute all waiting instructions before we stop our frame timer. The command glFinish() blocks waiting for all OpenGL commands to execute. This command causes a stall in CPU execution and performs a full round trip to the graphics card and can be quite computationally expensive if called indiscriminately. However, by ensuring that we have performed all necessary CPU instructions and by waiting till the very end of the pipeline before calling glFinish(), we can ensure that we have completed all the rendering steps and avoid any costly blocking overhead. In addition, once the user specified minimum sampling rate has been reached, our system reduces the resolution of the image in order to reduce rendering demand even further.

4.2.3 Results

We tested our implementation by recording the frame rate and the sampling rate for a 360° rotation around a number of random volume datasets (See Table 4.1). We then compared the differences between a standard rendering and our time-critical rendering. As expected, the frame rate for a standard rendering can vary quite dramatically, in some cases there can be as much as a 40% difference between the maximum and the minimum frame rate for the same dataset (e.g. the walnut dataset). In contrast, time-critical rendering quickly adapts to any changes in scene complexity and dynamically adjusts the sampling rate in order to maintain the target frame rate. As can be seen from Figures 4.3–4.6 and Table 4.2, time-critical rendering is successful in smoothing out variations in frame rate and providing the user with a constant interactive frame rate in order to easily navigate the volume.

As can be seen from Figure 4.1 and Figure 4.2 we can reduce the appearance of artifacts associated with low sampling rate renderings and improve final image quality. In addition, time-critical rendering requires very little modification to a normal ray cast volume rendering pipeline, and therefore can be integrated into many already existing volume rendering optimisations.

Our implementation integrates into the core rendering loop of any ray cast direct volume rendering system and only requires a very simple linear interpolation calculation and a timing query in order to work. Therefore, the overhead is minimal. From analysis

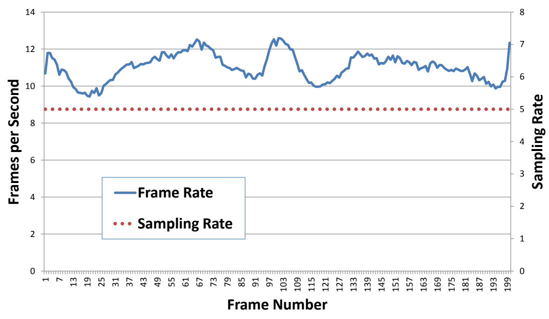
Table 4.1: The dimensions of the datasets we used in our evaluation.

Name	Size
Vismale	128 x 256 x 256
Walnut	128 x 96 x 114
Porsche	559 x 1023 x 347
Bucky	32 x 32 x 32

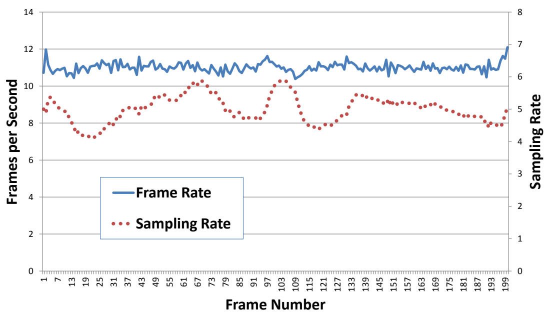
Table 4.2: The average frames per second and the standard deviation for each dataset. Our time-critical implementation significantly reduces the standard deviation resulting in smoother and less varied frames per second.

	Standard Rendering		Time-Critical Rendering	
Dataset	Average (fps)	σ	Average (fps)	σ
Vismale	11.03	0.73	11.01	0.25
Walnut	13.32	2.09	14.01	0.35
Porsche	6.25	1.13	6.27	0.17
Bucky	13.83	0.98	13.00	0.27

of the frames per second numbers produced from our testing we observed that the time-critical approach produced no significant overhead. Our method is able to quickly and consistently determine the ideal sampling rate in order to achieve the targeted frame rate. This ability to guarantee a given frame rate is invaluable in generating volume renderings for interactive user navigation.

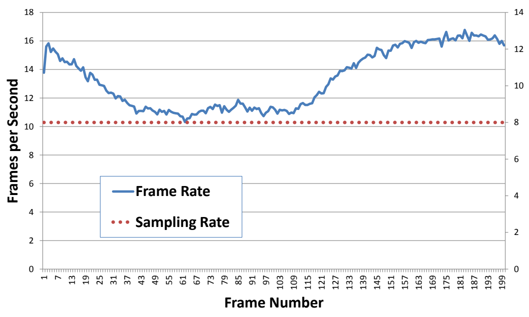


(a) Standard rendering.

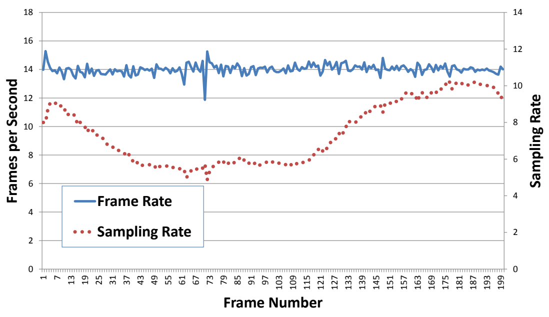


(b) Time-critical.

Figure 4.3: Time-critical and standard renderings of the Vismale dataset.

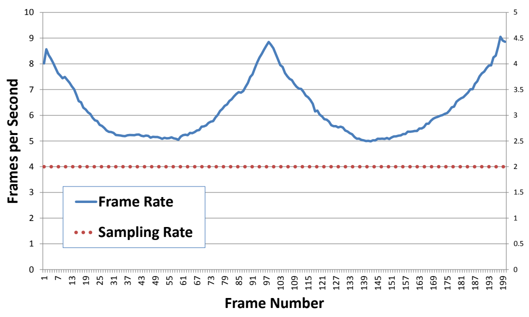


(a) Standard rendering.

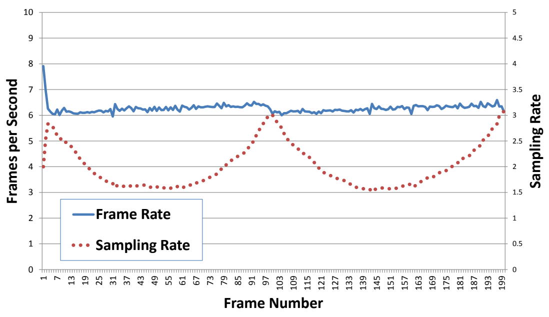


(b) Time-critical.

Figure 4.4: Time-critical and standard renderings of the Walnut dataset.

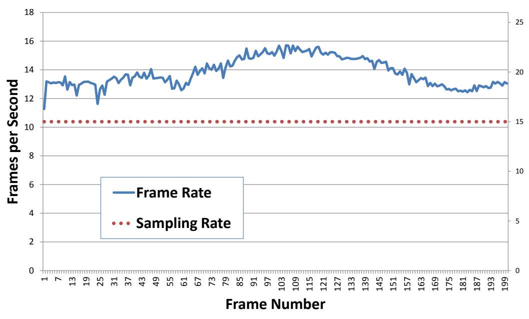


(a) Standard rendering.

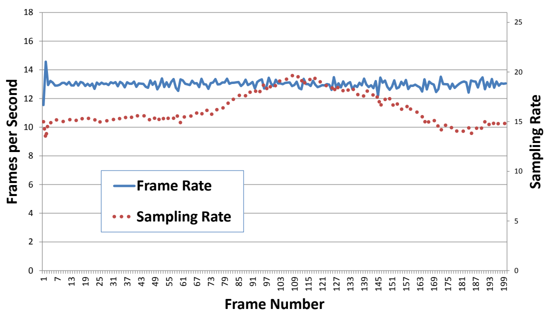


(b) Time-critical.

Figure 4.5: Time-critical and standard renderings of the Porsche dataset.



(a) Standard rendering.



(b) Time-critical.

Figure 4.6: Time-critical and standard renderings of the Bucky dataset.

4.3 Image Space Adaptive Rendering

Importance maps play an important role in increasing the speed of volume rendering. Researchers have exploited the sparseness of volume data, dynamically reducing sampling frequency in areas of the volume where no significant data is present, in order to increase overall rendering speed [LMK03]. Most importance map techniques require an expensive preprocessing step in order to generate a 3D importance volume which describes the important areas of the volume dataset and which is then sampled at run time in order to adjust the sampling rate. Additionally, this importance map is transfer function dependant and requires an expensive recomputation if the user changes the transfer function. This importance volume then needs to be uploaded to GPU memory, further reducing the already limited memory available for volume rendering and also increasing the number of texture accesses required to render a frame.

We propose a method for producing a 2D, image space importance map for focus and context rendering which reduces the high memory requirements of a 3D map. We exploit temporal coherence in order to rapidly compute the importance map which, in combination with its 2D nature, allows for on the fly computation in every frame. This enables our technique to account for transfer function changes without the expensive recomputation required by 3D importance maps. In addition, our method only requires one texture sample per ray, in contrast with a 3D importance map which requires an additional texture sample at multiple points along the ray. This further reduces the computation requirements of our method. We also propose two methods to increase the accuracy of our importance map technique: a ray casting technique to detect and correct inaccurately classified pixels, and a dilation filter applied to the importance map in order to expand the importance area.

We have compared our technique to a standard volume renderer [MSRMH09] and for the same computation time our method is able to increase the sampling rate of areas of the volume that have been classified as important. In addition, we analysed our importance generation method and found that it correctly classifies the vast majority of pixels with only a very small percentage of pixels incorrectly classified as unimportant. These incorrectly classified areas were analysed and found to have very low alpha values and thus very small contributions to the final image.

4.3.1 Framework

Our technique works by assigning higher sampling rates in areas of the image that are visually important while areas of the image that are classified as unimportant are

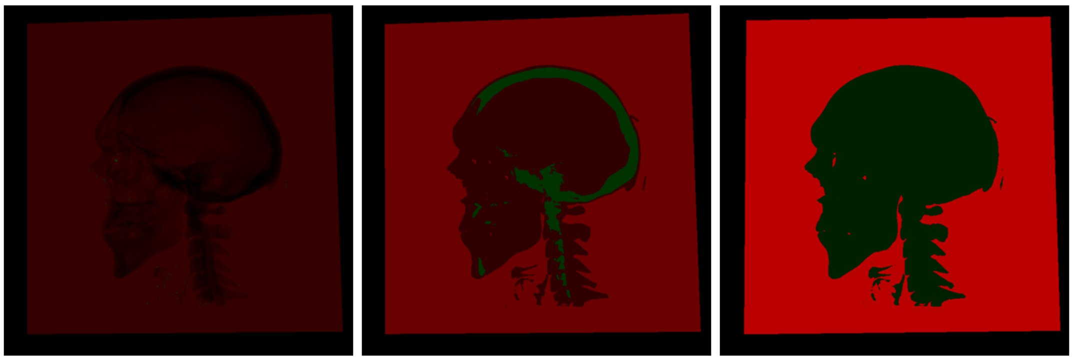

(a) 2^{32} Importance Levels. (b) 3 Importance Levels. (c) 2 Importance Levels.

Figure 4.7: How the number of importance levels affect rendering quality. The red regions of the image show areas where our technique has reduced the sampling rate and the green regions are areas where our technique has increased the sampling rate.

rendered at low sampling rates. By implementing this trade-off between high and low quality we are able to increase the rendering quality of important areas of the image while maintaining the same computation time as a standard volume rendering technique. This results in an overall higher quality image for the same computational cost. We initially investigated utilising variable sampling rates, however, as other researchers have found, limiting adaptive sampling to discrete values exploits GPU cache coherency and provides a faster result [HSS+05]. We found that a binary high and low quality sampling scheme (Figure 4.7) provides both faster computation and improved image quality when compared to multiple discrete sampling rates or a variable sampling rate.

4.3.1.1 Pipeline Overview

Assuming an importance map was generated by the previous frame, the first step in our technique is to modify the rendering loop in order to sample the importance map. For every ray being cast, the renderer samples the importance map once in order to determine what sampling rate to use.

In the second step, the renderer casts rays using the sampling rate calculated from the importance map in order to produce a rendered image. Simultaneously, the rendering system also calculates the importance of each pixel which is output into a 2D image.

During the third step, sampling correction is performed. If the importance map has inaccurately assigned a low sampling rate to a ray which, during the casting process was discovered to be important, we recast the ray at a higher sampling rate and update the importance map and rendered image accordingly.

In the final step of the technique, the importance map is enlarged using a dilation

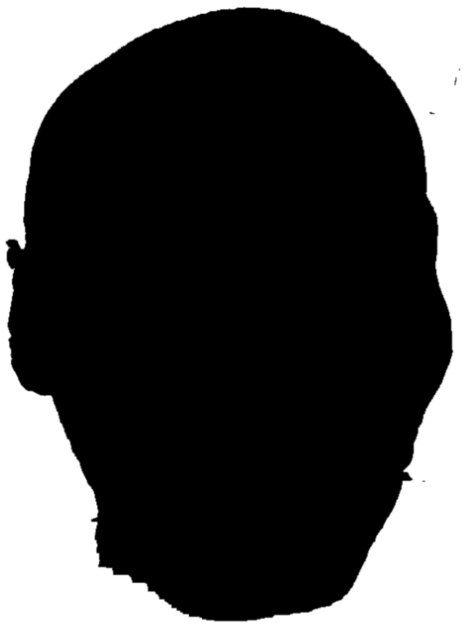


Figure 4.8: A sample binary importance map from our system.

operator in order to reduce the number of incorrect classifications and then stored for use in the next frame.

4.3.1.2 Importance Map Generation

Due to temporal coherence, areas of the image that were classified as important in the previous frame are highly likely to be important in the next frame. We can exploit this property by generating a 2D, binary importance map (Figure 4.8) from the previous frame and using it to guide sampling rate increases in the next frame. Traditional importance maps have been generated in three dimensions in order to allow for changes in camera position without requiring an importance map update. These importance maps consume large amounts of memory, take significant time to compute, and require regeneration whenever the transfer function changes. By limiting our importance map to two dimensions, we dramatically reduce its memory requirement and also decrease the computational cost of generating the map. In practice, this means we are able to recalculate a new map every frame in real-time which allows us to instantly handle importance changes due to variation in both the view and the transfer function, something which 3D importance map approaches struggle with.

We investigated a number of different techniques for determining if an area in the image is important: the user selected focus region, the number of samples required before early ray termination can be performed, the ratio of opaque to transparent samples,

Table 4.3: Alpha threshold variation for the Skull dataset.

Alpha Threshold	Detected	Corrected	Missed	Quality Increase	Alpha of Missed
0.01	97.9%	1.7%	0.4%	19.7%	0.08
0.05	97.9%	1.7%	0.4%	19.3%	0.08
0.10	97.8%	1.8%	0.4%	19.8%	0.08
0.20	97.5%	2.0%	0.5%	19.8%	0.09
0.25	97.2%	2.1%	0.6%	20.0%	0.13
0.50	86.4%	2.4%	11.2%	22.6%	0.36
0.60	83.8%	1.7%	14.5%	24.9%	0.39
0.70	82.1%	1.5%	16.4%	25.7%	0.41
0.80	80.8%	1.2%	18.0%	27.4%	0.43
0.90	79.2%	1.1%	19.7%	29.3%	0.44

and the alpha value of the final colour. However, we found that the more advanced techniques required a large amount of computation and that a simple threshold of the alpha value of the final colour gives a good result with a minimum of computation. In the future, formal tests on the effectiveness of the different importance measures versus their computational expense could be useful. Manual metrics such as user selected importance or head tracking guided importance were not investigated as they were outside the scope of this thesis.

We have modified a standard volume ray casting shader to output a second image which shows which areas of the scene are important. Within the ray casting shader, we analyse the alpha value of the final colour, and if it is above a certain threshold, the current pixel is deemed important and the importance map is updated accordingly. The threshold value can be changed on the fly and can even be set so that if an area of an image contains any information at all it is classified as important. In practice, we have found that setting an alpha threshold value of 0.05, in combination with sampling correction and dilation described in the following sections, provides a good trade-off between computation and image quality. Table 4.3 shows the results for alpha threshold variation on the Skull dataset, the terms in this table are discussed in more detail in Section 4.3.2.4. This 0.05 value also matches the commonly used threshold for optimisation techniques such as early ray termination and therefore allows us to combine two conditional checks, resulting in a small performance increase.

4.3.1.3 Sampling Correction

As described in the previous section, we have integrated our importance map generation into the same pass as the volume rendering shader. Our current implementation casts a ray for each pixel of the viewport and iterates along the ray accumulating the importance value for input into the next frame and the pixel colour. As we are using the previous frame to generate the importance map for the current frame, it may happen that some areas of the image that were unimportant in the previous frame have become important in the current frame and have incorrectly been sampled at low quality (Figure 4.9). It is possible to detect these false negatives on the fly and correct them without requiring additional shader passes. When the shader is looping along the ray it is accumulating a value for both the colour of the current pixel and for its importance. If we determine that the current pixel is important, and a low sampling rate was incorrectly used to calculate the final colour, we can exit out of the loop and restart the ray casting calculation but this time at the higher sampling rate. This avoids the additional overhead that a second complete GPU pass would require and allows us to quickly correct for pixels that have incorrectly been classified as unimportant.

Algorithm 1 Per-pixel sampling correction.

```
if (previous frame importance = false) then
    while (within volume) and (current frame importance = false) do
        raycast at low quality and compute current frame importance
    end while
end if
if (current frame importance = true) or (previous frame importance = true) then
    while (within volume) do
        raycast at high quality and compute current frame importance
    end while
end if
previous frame importance = current frame importance
return  raycast result
```

Our current importance map generation technique iterates along the ray, and the combination of each individual importance value contributes to the final importance value for the current pixel. If a pixel is being sampled at low quality, we check at each step along the ray to see if the pixel has passed the importance threshold in order to determine if the ray casting loop should be terminated early and restarted at a higher sampling rate. For the vast majority of pixels in the image, the importance map generated from the previous frame is accurate and the importance checks at each step along the ray are computationally expensive. A simplified version of our sampling correction algorithm is shown in Algorithm 1.

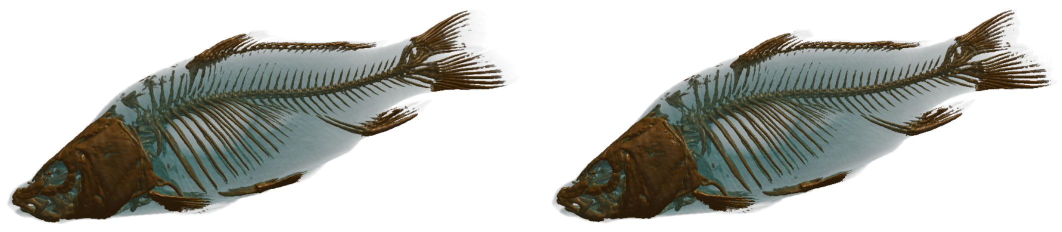
Figure 4.9: Classification image. Green pixels have been correctly classified, yellow pixels were corrected and blue pixels have been misclassified.

Table 4.4: Performance for the different importance check positions.

Check Position	Skull	Carp	Porsche
Outside Loop	56.0%	14.6%	17.2%
Every Sample	60.8%	27.7%	20.8%

Practical Considerations We postulated that the overhead of constantly checking if the ray was important at each ray casting step would cause a large performance penalty due to the parallel nature of GPU architecture and the small number of rays which require sampling correction. We investigated a method for eliminating the overhead of importance checks by reducing their frequency.

Since only a small number of rays require sampling correction, we investigated eliminating the importance check completely from the ray casting loop and only checking importance once a complete low quality ray had been calculated. This eliminates the need to perform an importance check every sample but means that incorrectly classified rays will perform two full computations, one at low quality and one at high quality. As can be seen from Table 4.4 this technique actually results in a performance penalty. The extra computations required to complete two full ray casts obviously outweigh the savings gained by reducing the number of importance checks.



(a) Standard rendering requiring 1785 milliseconds to generate.

(b) Our technique requiring 43 milliseconds to generate.

Figure 4.10: Comparison of our method to a standard volume rendering system. The image from our technique was generated 41 times faster than the standard rendering and still provides a high detail image.

4.3.1.4 Dilation

By utilising both an importance map and sampling correction we ensure that the vast majority of pixels are classified properly and are sampled at the correct quality. However, it is still possible that a small number of pixels may be incorrectly classified as unimportant. This usually occurs at low sampling rates when small areas of importance are more likely to be missed due to the large distances between sampling points. These important areas are never sampled by the raycaster, and thus it is not possible for our sampling correction algorithm to detect them and resample the area at a higher quality. We observe that most of these incorrectly classified pixels occur at the boundary between important and unimportant regions. We have implemented a dilation filter [Ser83] which expands the edges of the importance map in order to encompass these incorrectly classified pixels. This reduces the number of pixels which are incorrectly classified as unimportant but also increases the number of false positives i.e., the number of pixels that are incorrectly classified as important. These false positives result in some pixels incorrectly receiving a high sampling rate. This has no detrimental effect on the final image quality but does increase the computational cost. We implemented our dilation filter as a two-pass GPU shader and allow the user to specify the dilation filter size so that they can determine the trade-off required between quality and computation speed.

4.3.2 Results

In order to provide a fair comparison between our technique and a standard volume rendering implementation, we need to analyse images that are generated at the same frame rate in both methods. This gives us a like-for-like comparison between our method

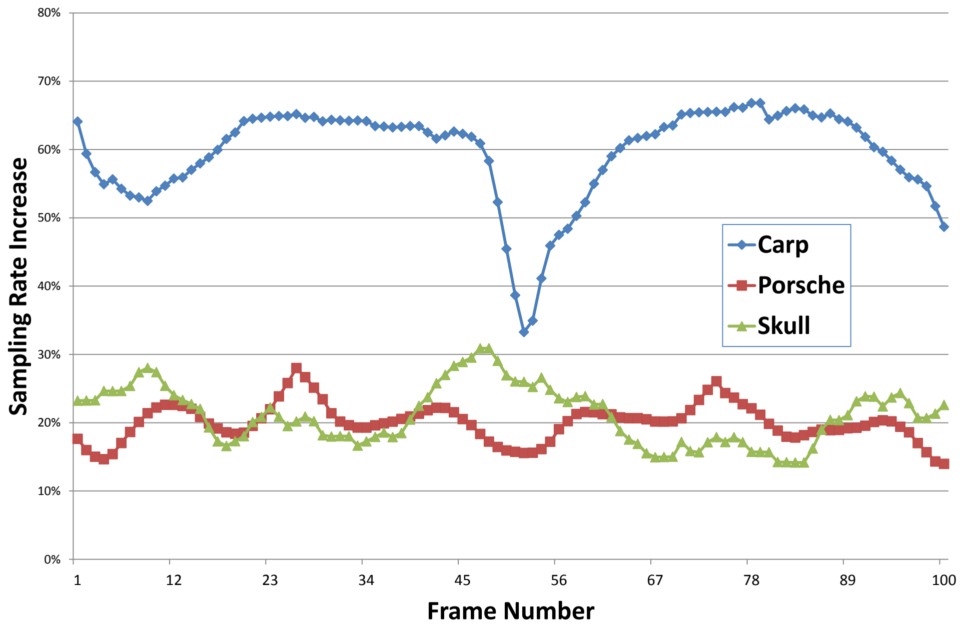


Figure 4.11: The sampling rate increase our method achieves for 100 frames of a 360° rotation of various datasets.

and a standard implementation and allows us to compare the sampling rates between the two techniques in order to determine the quality difference in the final images. We integrated our time-critical rendering algorithm (as described in Section 4.2) into the system, thus allowing us to easily determine the correct sampling rate in order to generate images from both methods with the same computation time.

4.3.2.1 Quality Increase

In order to determine the change in sampling rate that our method provides, we modified both a standard volume renderer and our technique to output the sampling rate used for each pixel. This allows us to compare the two techniques and show in which areas and by how much the sampling rate, and therefore the quality of the final image has changed. As can be seen from Table 4.5, our technique has increased the sampling rate in important areas of the image by between 20% and 60% (important areas were defined as those areas with an alpha value higher than 0.05). Figure 4.11 shows the quality results for a 360° rotation around each dataset. We believe that the large variation in quality for the carp dataset is due to its highly rectangular dimensions which cause a large amount of view dependence.

Table 4.5: Average sampling rate increase for each of the datasets tested.

Dataset	Size	Sampling Rate Increase
Skull	256x256x113	21.2%
Carp	256x256x512	59.5%
Porsche	559x1023x347	19.9%

4.3.2.2 Performance Cost

Our technique modifies a standard rendering pipeline to allow for a reduction in sampling rate in unimportant areas of the scene, balanced by an increase in quality in important areas. As we have shown in the previous section, this results in an increase in sampling rate for the vast majority of scenes. However, we wished to analyse the overhead of our approach and how our technique performs in the pathological worst case: where every pixel in the scene is classified as important, and no reduction in sampling rate can be achieved. Using the same technique we used in the previous section of comparing the sampling rate of our technique to that of standard volume renderer, we modified the scene so that more and more of the important area filled the viewport. As can be seen from Figure 4.12, when the full screen is occupied by important pixels our technique performs approximately 5% worse than a standard rendering. This graph also shows that it only requires approximately 5% of the pixels on the screen to be classified as unimportant for our technique to equal or better the performance of a standard rendering system. This analysis shows, that in the worst case scenario for our technique, it only suffers a loss in performance of 5%, and for the vast majority of scenes it performs better than a standard rendering system.

4.3.2.3 Optimum Weighting

We also investigated the optimum ratio to have between the high and low sampling rates. If the difference between the sampling rates is quite small, then the overhead of generating the importance map might outweigh the performance benefit of our system. If the difference between the sampling rates is quite high, then the quality of the lower sampled areas might not be acceptable. Figure 4.13 shows the relative weighting of the high sampling rate over the low sampling rate and the percentage of the screen classified as important in order to achieve the same quality as a standard volume rendering system. For a low relative difference between the sampling rates, a quality increase in important areas is only achieved if a low number of important pixels are present in the scene. As the difference between the sampling rates increases, the overhead of our system becomes

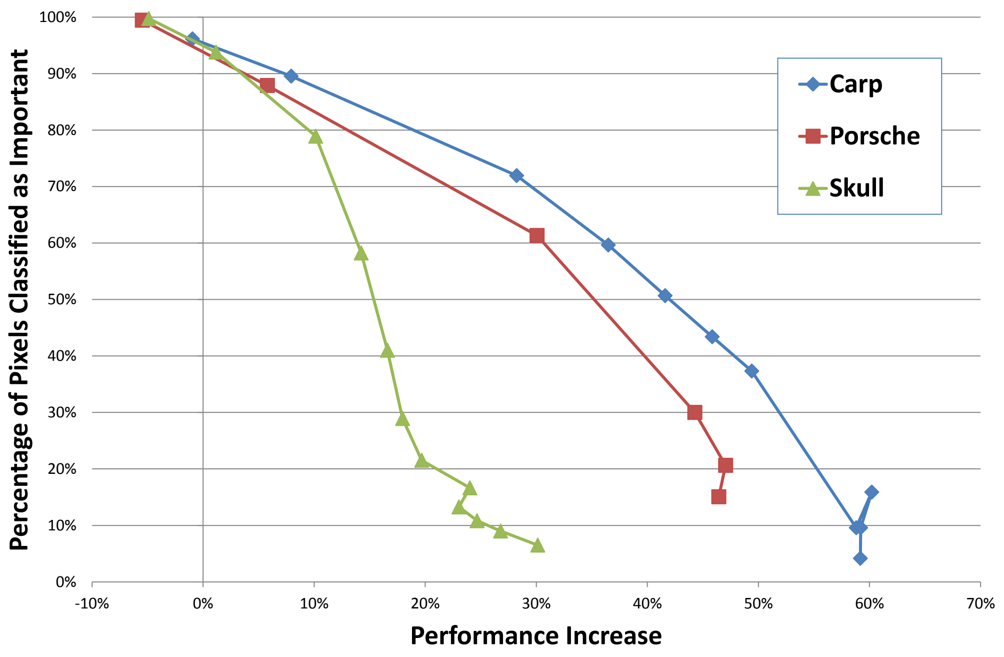


Figure 4.12: The percentage of pixels classified as importance versus performance for our technique.

less prevalent and a larger amount of the screen can contain important pixels. The graph begins to flatten out once the higher sampling rate is twice that of the lower rate, therefore, we set this 2:1 ratio as the low limit for the weighting. We do not set an upper limit on the weighting as we wished to allow the user flexibility to choose how important the higher regions are compared to the lower regions.

4.3.2.4 Error Correction Effectiveness

As our technique achieves high quality rendering in important areas of the image by reducing the quality in other areas, it is important to check if any of the areas classified as unimportant have been done so incorrectly. We compared our method to a standard volume renderer and stipulated that any areas of a standard rendering which contained colour information should be determined as important by our technique. For every frame we then compared the two renderings. We calculated the areas of the image which had received a high sampling rate, the areas which required sampling correction, and the areas which had been incorrectly sampled at low quality. As can be seen from Table 4.6, between 86% and 98% of significant pixels were correctly classified, 11% to 2% of pixels were correctly classified after sampling correction and 3% to 1% of pixels were incorrectly rendered at a low sampling rate. Figures 4.14–4.16 show the results of this classification analysis.

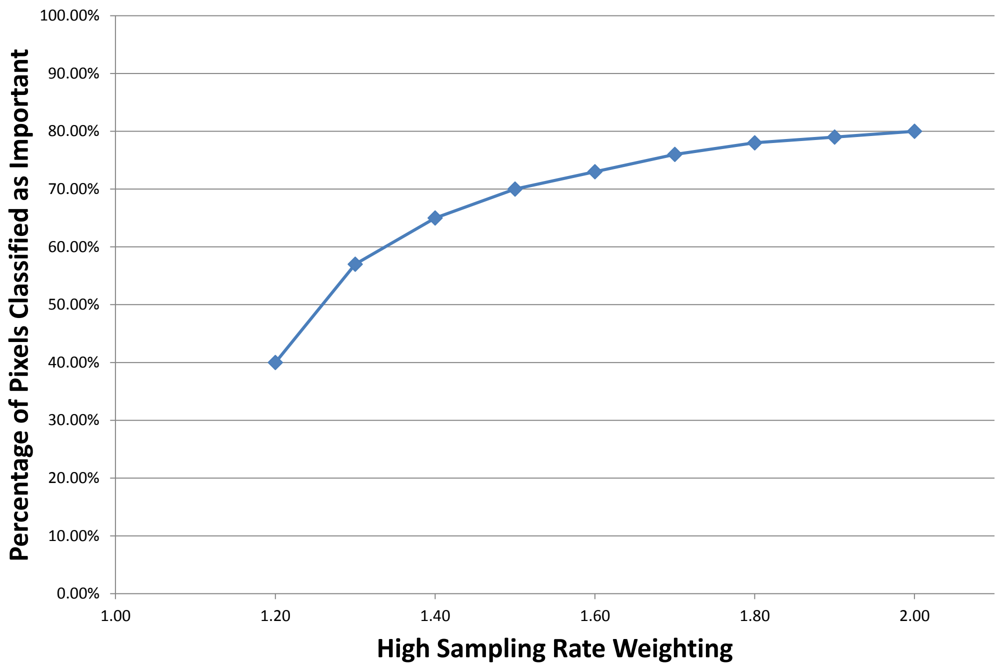


Figure 4.13: The percentage of pixels classified as important versus the high sampling rate weighting.

Table 4.6: Average classification results for each of the datasets tested.

Dataset	Detected	Corrected	Missed	Alpha of Missed Pixels
Skull	98.0%	1.7%	0.3%	0.06
Carp	96.1%	3.7%	0.2%	0.13
Porsche	85.6%	11.4%	3.0%	0.05

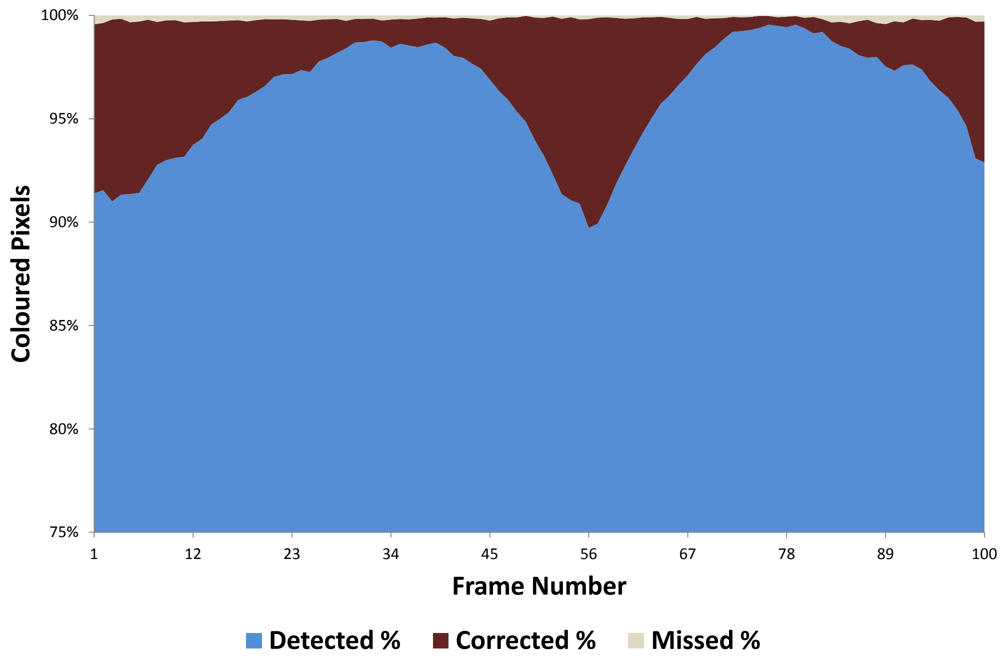


Figure 4.14: Classification results for 100 frames of a 360° rotation of the Carp dataset.

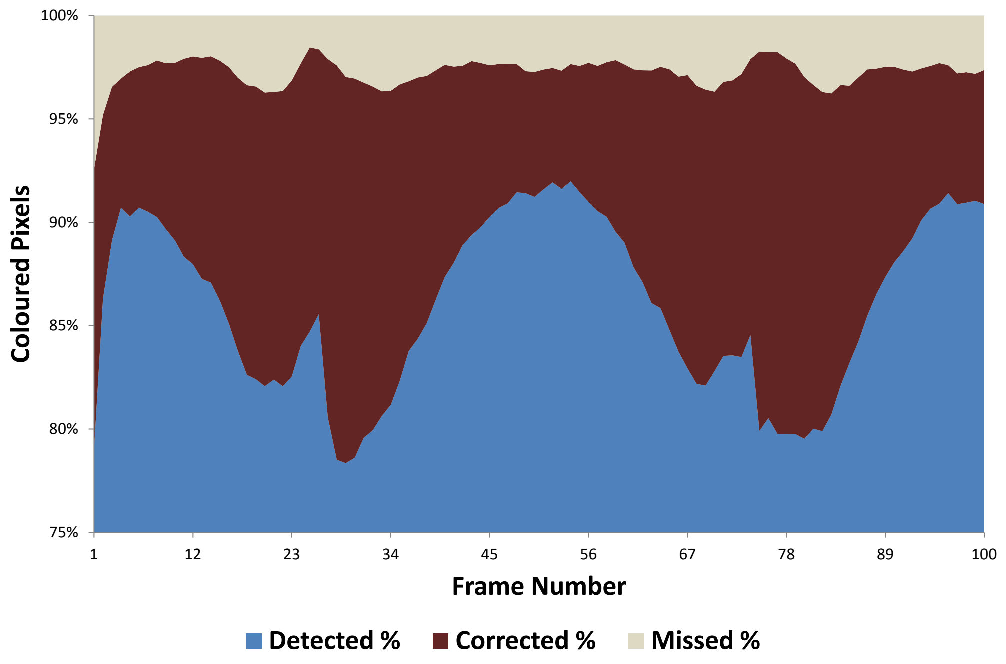


Figure 4.15: Classification results for 100 frames of a 360° rotation of the Porsche dataset.

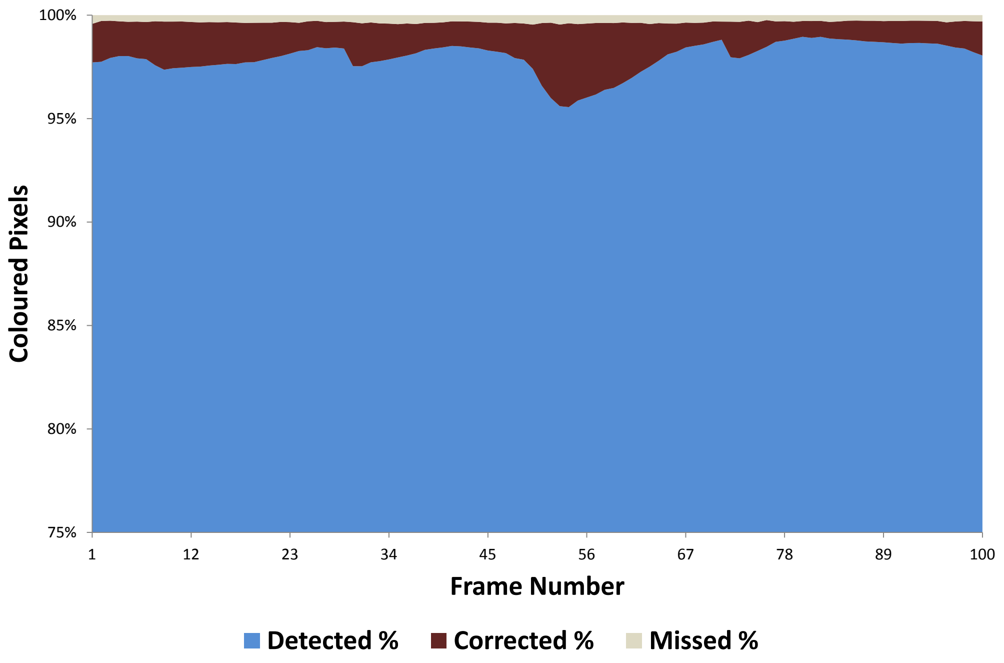


Figure 4.16: Classification results for 100 frames of a 360° rotation of the Skull dataset.

We hypothesised that the areas of the image that were incorrectly classified were undetected due to their small size, and therefore would have a very small contribution to the final image. We analysed the standard rendering images to determine the alpha values of the pixels that had been missed by our technique. As can be seen from Table 4.6 and Figures 4.17–4.19, the majority of the missed pixels have very small alpha values. This confirms that the majority of the pixels we miss have a low alpha value and are very transparent. Therefore, even though we are sampling some pixels at a lower sampling rate than a standard rendering, these pixels contribute little to the final image.

4.3.2.5 Dilation Effectiveness

Tables 4.7 and 4.8 show the sampling rate increase and classification results for the Porsche dataset with a number of different sized dilation filters applied to the importance map. The results show that applying a dilation filter can dramatically improve classification accuracy, with up to a 67% reduction in the number of pixels requiring correction and a 50% reduction in the number of missed pixels for even a small 3x3 dilation filter. In addition, when a dilation filter is applied, those pixels which are misclassified have a lower average alpha value than a rendering using an undilated importance map. Increasing the size of the dilation filter requires more calculations to generate the importance map and also increases the rendering time of the volume data due to the larger area of the image that is classified as important. However, it simultaneously reduces the

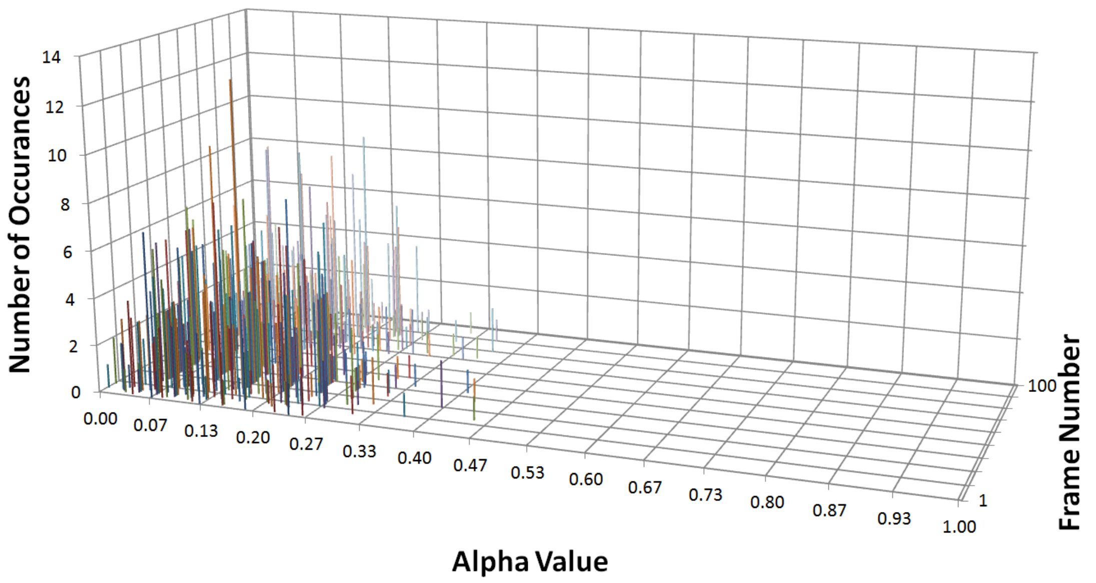


Figure 4.17: Histogram of the alpha values of the incorrectly classified pixels for the Carp dataset.

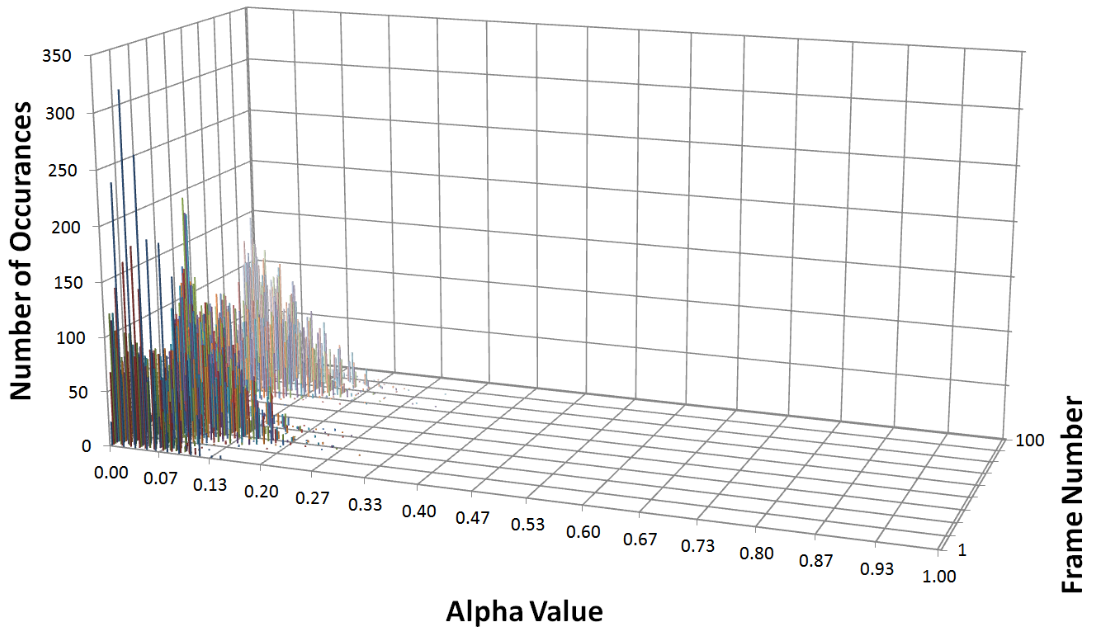


Figure 4.18: Histogram of the alpha values of the incorrectly classified pixels for the Porsche dataset.

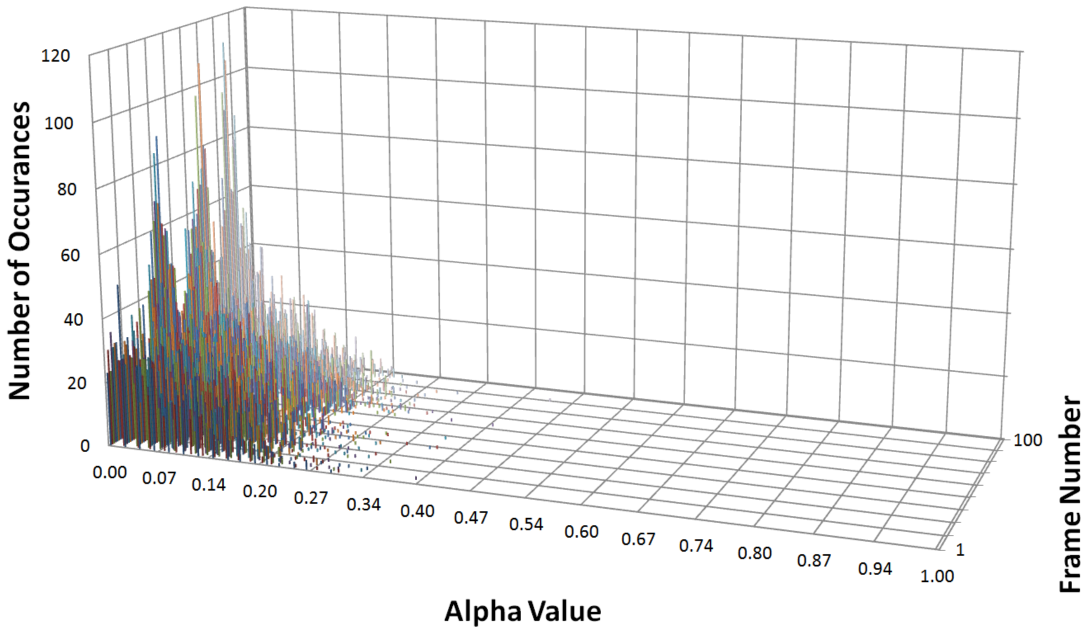


Figure 4.19: Histogram of the alpha values of the incorrectly classified pixels for the Skull dataset.

Table 4.7: Average sampling rate increase for the Porsche dataset with the dilation filter applied.

Filter Size	Sampling Rate Increase
No Dilation	19.9%
3x3	20.9%
5x5	20.2%
7x7	18.9%

number of sampling corrections required which improves rendering speed. By careful selection of the dilation filter size, it is possible to both increase sampling rate and reduce misclassifications simultaneously.

Table 4.8: Classification results for the Porsche dataset with the dilation filter applied.

Filter Size	Detected	Corrected	Missed	Alpha of Missed Pixels
No Dilation	85.6%	11.4%	3.0%	0.05
3x3	94.8%	3.8%	1.5%	0.04
5x5	98.0%	0.9%	1.0%	0.02
7x7	99.1%	0.1%	0.8%	0.01

4.4 Discussion

We have presented a system which allows for dynamic variation of the sampling rate of ray cast direct volume rendering in order to maintain a stable frame rate. Our system is robust and quick to respond to both user input and significant computational loads and rapidly converges on the target frame rate. We have tested our technique with a wide variety of datasets, and it has succeeded in maintaining the target frame rate on all datasets at a wide variety of target frame rates. Our technique can be integrated into existing ray casting optimisation techniques and contains no significant overhead.

By combining this system with other novel techniques, we developed image space adaptive sampling for direct volume rendering to allow for interactive navigation of a focus and context volume dataset. Our method utilises a 2D importance map which dramatically reduces the memory requirements of adaptive sampling when compared to a traditional 3D map. We exploit frame-to-frame coherency in order to enable us to generate our importance map on the fly for each frame at interactive frame rates. This allows us to quickly adapt to changes in transfer function and camera position. We also provide two techniques: importance map dilation and shader sampling rate correction, which reduce the number of incorrectly classified pixels in our importance map, improve the quality of the final image, and provide performance improvements. Finally, we tested our technique against a standard direct volume rendering system and showed that we can improve the rendering quality by an average of 20% to 60% while misclassifying only a small percentage of pixels, most of which are highly transparent and contribute little to the final image.

Chapter 5

Lighting and Shadowing

5.1 Introduction

The addition of advanced lighting and shadowing techniques has been shown to increase the perception and understanding of complex volume datasets [RDRS10, LR11]. Due to the highly complex nature of volume data, advanced illumination calculations are computationally expensive and difficult to render at interactive frame rates.

Researchers have developed a variety of techniques to improve the rendering speed of illumination calculations for volume data, but these techniques are either not applicable to ray cast renderings, require a reduction in quality, an increase in memory requirements or a pre-processing step in order to function. The large datasets that are being captured in recent times can no longer fit in memory, and thus techniques which increase memory requirements are no longer attractive. Additionally, the large body of research that has been directed towards optimising ray cast rendering systems means that texture slicing approaches to volume rendering are no longer competitive in terms of computational speed, and illumination techniques which are compatible with ray cast solutions are favoured.

We propose enhancing focus and context volume rendering by integrating real-time illumination techniques. The technique we propose utilises this paradigm by allowing the user to specify a target isosurface within the volume data. This target surface is rendered with high quality, ray cast shadows. The surrounding regions are illuminated in a less pronounced manner with a computationally efficient ambient occlusion calculation which is able to highlight the important structures of the surrounding regions for negligible cost. By splitting a rendering into two levels, it is possible to provide high quality rendering techniques for the target surface while simultaneously enhancing the surrounding context regions with a more computationally efficient calculation which still enhances the final image but for a much lower computational cost. This technique allows for real-time illumination of the entire dataset, with no extra memory requirements, by trading computational speed for a less detailed illumination solution in the surrounding regions.

We analyse the effectiveness of our technique by performing a series of comprehensive user experiments to test how a users perception of shape, depth, and visual information is affected by our solution. In addition, we perform a detailed investigation into the performance of our technique and how it can be integrated into existing volume rendering pipelines. Through careful analysis of the results of our user experiments, we are able to recommend optimum lighting strategies for enhancing focus and context volume rendering.

5.2 Ambient Occlusion

Ambient occlusion is an illumination technique which uses an unconventional approach to shading calculation. Instead of tracing light rays as is usually done in illumination algorithms, ambient occlusion instead calculates how easy it is for light to access a point i.e., how occluded a point is by the surrounding material. A popular approach that is used to calculate ambient occlusion for volume rendering is to analyse a local neighbourhood around each point and calculate the ratio of opaque to transparent voxels. The more a voxel is surrounded by opaque voxels the less likely it is that light can reach it.

Our efficient ambient occlusion calculation is a modification of the technique used by Ropinski et al. [RMD+08] (See page 30 for more details). Instead of precomputing and storing histograms for each voxel, we compute the ambient occlusion contribution on the fly during the rendering process, therefore, eliminating the memory requirement incurred by Ropinski et al. In order to ensure that the ambient occlusion calculation is performed in real-time, we are required to reduce the number of samples used by the ambient occlusion calculation to the surrounding six voxels. One would expect that this reduction in sampling size would result in a dramatic decrease in visual quality, but as we show in Section 5.5.2.1 this method still provides an increase in visual information. In addition, we perform this fast method of ambient occlusion only in the area of the volume classified as peripheral context information, and therefore a trade-off between visual quality and computational speed is acceptable. Sample code showing our implementation is show in Listing B.7.

We perform this fast ambient occlusion calculation by utilising the information

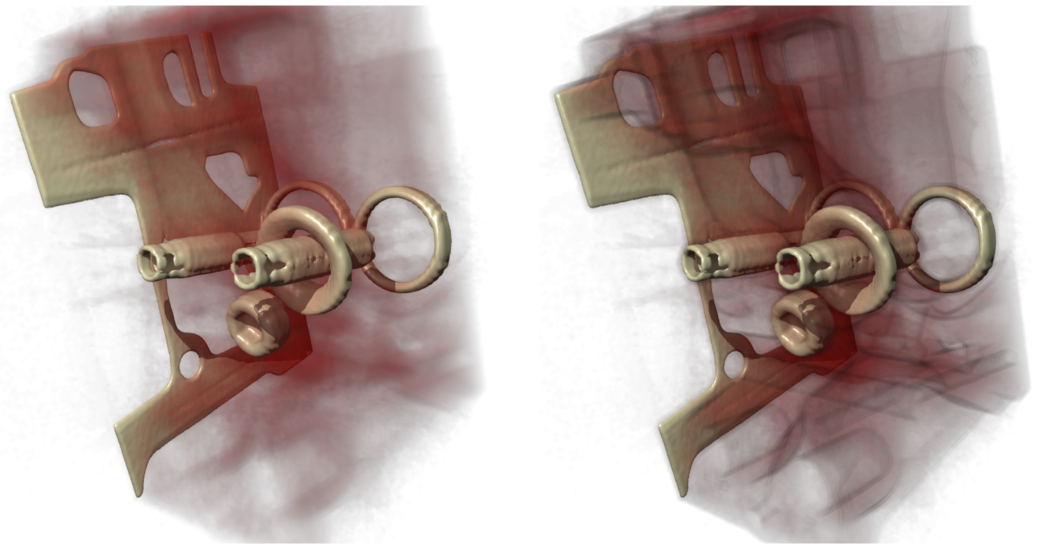

(a) Standard rendering. (b) Ambient occlusion enabled.

Figure 5.1: How our fast ambient occlusion technique can highlight context structures.

that would normally be accessed during a traditional per-voxel Phong illumination approach. In order to calculate the gradient information for Phong illumination, most techniques generally use on the fly gradient calculation using finite differences. This gradient calculation method samples the volume data once in each of the six directions surrounding each sampling point in order to generate gradient information. By limiting our ambient occlusion calculation to only the surrounding six directions, we utilise the same texture accesses that are required by the gradient calculation, thus reducing computation time. Using this gradient information, we perform a dependant texture lookup into the transfer function in order to determine the opacity of the surrounding region. If any of the surrounding area is of a higher opacity than the current voxel, we assume that the current voxel is occluded and modify its luminance based on how occluded the voxel is (Figure 5.1).

5.3 Ray Cast Shadows

Shadow ray casting consists of casting a ray from every surface towards the light source. If the ray encounters another surface before it reaches the light, then shadowing occurs. For volume rendering, this technique can be modified slightly to allow early shadow ray termination and therefore reduced computation time.

Standard shadow ray casting techniques terminate the ray when it reaches the light source. This technique works well for typical ray casting scenes where the light source is positioned within the scene. However, for the vast majority of cases in volume rendering, the light source is positioned outside the bounding box of the volume data which results in unneeded shadow ray calculations being performed in areas outside the volume where no occlusion can possibly occur. Using this knowledge, it is possible to perform early shadow ray termination by calculating the distance from each voxel to the maximum extents of the volume bounding box and terminating the shadow ray when it exceeds this distance. By taking the minimum of the distance from the voxel to the light source and the distance from the voxel to the extents of the bounding box we can ensure that no extra ray iterations are performed for all possible light source positions both inside and outside the volume.

5.3.1 Automatic Adaptive Sampling

With traditional volume rendering, the sampling rate of each ray directly impacts the image quality and too low a sampling rate can result in distracting artifacts. With shadow ray casting, sampling rate frequency has an even larger bearing on image quality due to the binary nature of hard shadows which are either present or not present. This binary on/off shadow condition results in easily noticeable and visually distracting image artifacts at low sampling rates. Figure 5.2a shows an example of this problem: as a result of a low sampling rate, shadow rays skip over potential occluders resulting in discontinuous and incorrect shadowing.

We have developed a technique for automatic, adaptive shadow ray sampling which increases the quality of ray cast shadows for minimal cost. Our technique is based on a similar method for adaptive sampling of view rays utilised in isosurface rendering [HSS+05]. We modify the shadow ray casting loop to allow for an increased sampling rate as the ray approaches a potential occluder. At every sample along the shadow ray, the volume is checked to see if the focus isosurface is present, and therefore occluding the shadow ray and causing a shadow. We dynamically increase the sampling rate of the ray if the isovalue of the current point along the ray is close to that of the isovalue of the target surface. This technique mimics a higher sampling rate but without the associated cost. Figure 5.2 shows how this technique can increase the quality and smoothness of shadows while simultaneously reducing rendering artifacts and displaying small shadowed regions that are missed at low sampling rates.

Table 5.1 shows the computational cost of this technique for a 360° rotation around the Skull dataset with varying levels of adaptive sampling quality. The quality level represents, on a scale from 0–1, how close a shadow ray needs to be to the target isovalue for its sampling rate to be increased. A quality of 0 means that a rays sampling rate is

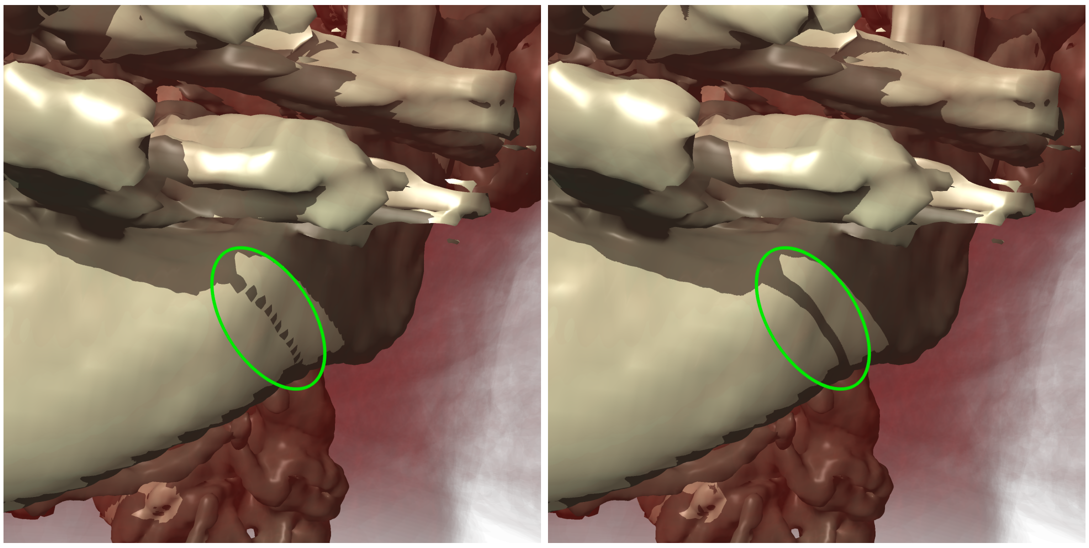

(a) Standard rendering. (b) Adaptive shadow ray sampling.

Figure 5.2: How adaptive shadow ray sampling can affect shadow quality.

never increased, and a quality of 1 means that every shadow ray is constantly sampling at a higher sampling rate (the higher sampling rate was set to eight times that of the user specified global sampling rate as this value was used by previous researchers [HSS+05]). As the quality level is increased, the frame rate converges on that of a rendering sampled at a constant high sampling rate. However, for low quality settings, shadow quality can be dramatically increased for only a small performance cost.

5.3.2 Jitter

In order to further increase the quality of our ray cast shadows we integrated stochastic jittering [HKRs+06] into our shadow ray casting pipeline. Similar to adaptive sampling, we adapted this technique from an implementation used for DVR view rays and applied it to shadow rays.

One of the problems of using low sampling rates for view rays is the appearance of wood-grain artifacts. These artifacts are caused by a large difference in depth between adjacent fragments which belong to the same isosurface and are most apparent when rendering large changes in opacity such as at a boundary between an isosurface and a transparent medium. Stochastic jittering randomly varies the start positions of each ray so that the sampling locations along the ray are randomised, therefore, masking the wood-grain patterns but introducing additional noise.

We have applied this stochastic jittering technique to shadow rays in order to randomly vary the sampling position along each ray and the results can be seen in Figure 5.3.

Table 5.1: Adaptive shadow ray sampling.

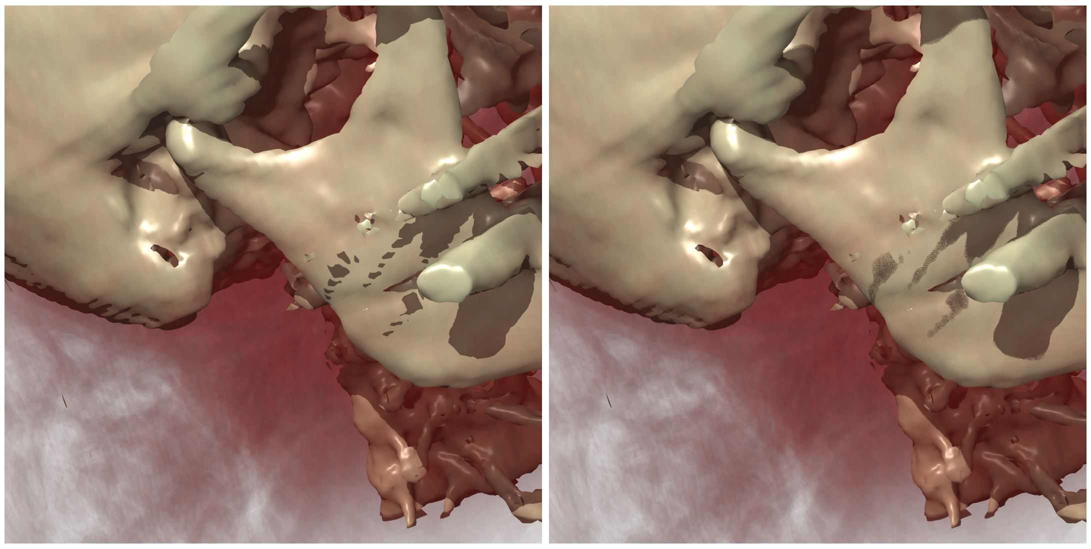
Quality	Frames per Second
0	26.36
0.01	26.17
0.02	25.74
0.03	25.39
0.04	22.86
0.05	19.87
0.06	19.66
0.07	19.46
0.08	19.46
0.09	19.4
0.1	19.43
1.0	19.48

This technique can increase image quality for a negligible performance cost (26.71 fps for a standard rendering and 26.51 fps for a jittered rendering).

5.3.3 Smooth Shadow Transitions

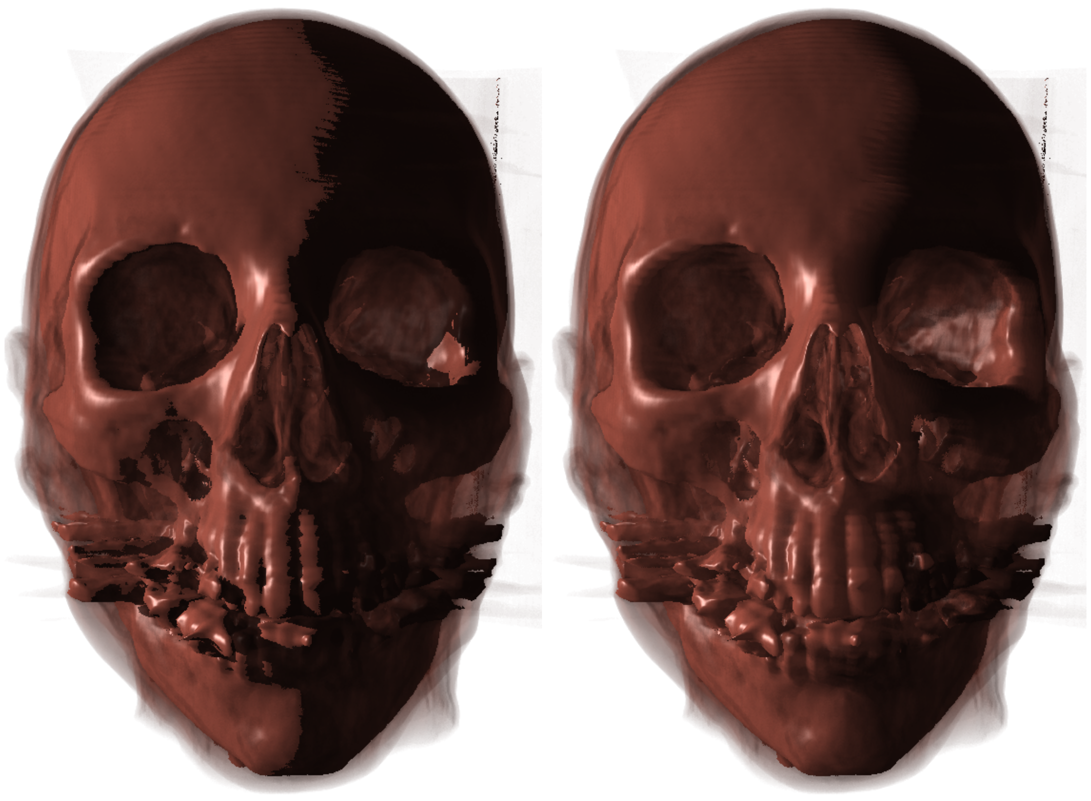
We have further modified our shadow algorithm in order to generate a smooth transition between lit and unlit areas. This technique provides an approximation of soft shadows (Figure 5.4) but is not based on a real world model of area light sources which are the lighting method usually used to generate soft shadows. Instead of simulating a dispersion of light from an area light source, we use a point light source and vary the shadowing based on occluder thickness.

Our initial hard shadows implementation stops propagating a shadow ray as soon as it encounters an occluding isosurface. We modify this technique to allow the shadow ray to continue propagating after encountering an occluder. Our algorithm keeps count of the number of times a shadow ray encounters the target isosurface, the more times the shadow ray encounters the isosurface the more its luminance is reduced. This technique causes thin occluders to cast softer shadows than thicker occluders. The user specifies the number of times the isosurface needs to be encountered for a voxel to be completely occluded. This determines the severity of the transition between fully occluded and fully lit areas of the volume, thus approximating soft shadows at very little computational cost. We analysed the performance of our shadow transition technique, and as can be seen from Table 5.2, even for shadows with large transitions between fully lit and fully shadowed areas, the extra computational cost was negligible.



(a) Standard rendering. (b) Stochastic jittering.

Figure 5.3: How stochastic jittering can affect shadow quality by reducing the appearance of sampling artifacts.



(a) Standard rendering. (b) Shadow transition range of 20.

Figure 5.4: Smooth shadow transition comparison.

Table 5.2: Average frames per second for a 360° rotation around the Skull dataset with varying shadow transitions.

Transition Range	Frames per Second
1	24.49
5	24.37
10	24.42
20	24.03

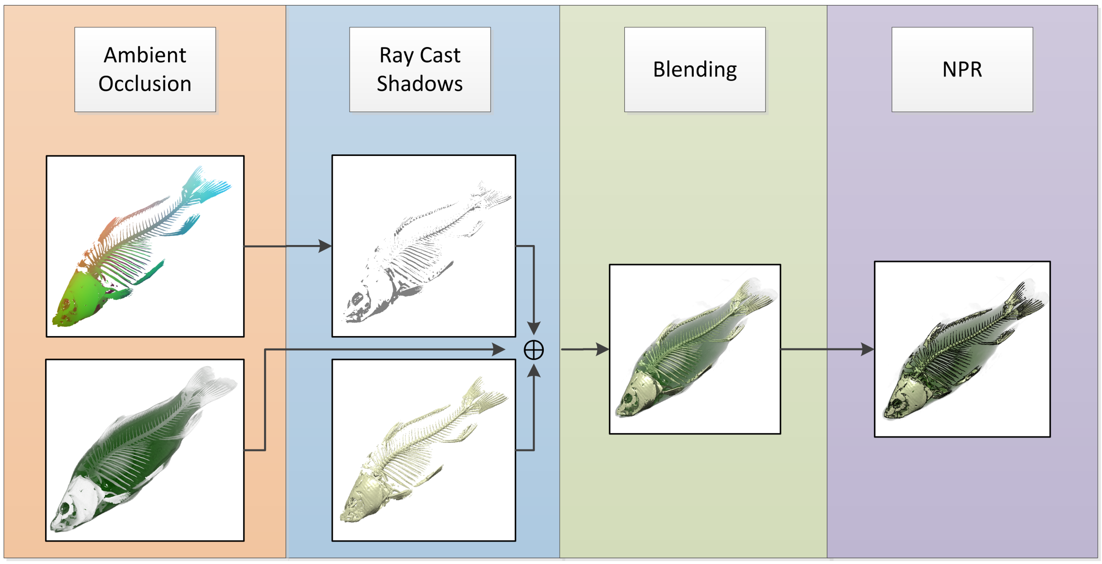


Figure 5.5: The rendering pipeline.

5.4 Framework

We have combined our ambient occlusion and shadow ray casting methods into a framework for generating illuminated focus and context volume renderings. Our technique works by calculating high quality ray cast shadows for the focus region of the rendering and a lower quality ambient occlusion shading solution for the context region. By combining the two lighting approaches together we have developed a framework which allows for real-time focus and context volume rendering, incorporating high quality shadowing and ambient occlusion. In addition, we have integrated NPR line drawings into the system, thus allowing for a combined lighting and line enhancement DVR rendering solution.

5.4.1 Pipeline Overview

We have implemented this technique using a two pass graphics processing unit (GPU) shader algorithm. By separating the ambient occlusion and the shadow ray calculation into separate passes we avoid expensive conditional checks in the GPU shader which can dramatically increase computation time.

In the first pass, we iterate along each ray computing the local ambient occlusion for every sample until the isovalue of the target surface is reached. At this point, we perform an iterative bisection procedure [HSS+05] in order to accurately determine the location of the isosurface and avoid artifacts due to low sampling rates. We then output the accumulated value of the ambient occlusion ray and the location of the isosurface for use in the second pass (Figure 5.5).

In the second pass, the ray cast shadow contribution is calculated by casting shadow rays from the isosurface location towards the light source utilising adaptive sampling in order to refine the shadow locations. The shadow ray information and the ambient occlusion rendering are blended together to generate a shaded image. If the user has indicated that they want to apply NPR enhancement to the focus layer, an additional image is output which contains an ambient and diffusely shaded rendering of the target isosurface.

In an optional third step, NPR lines are extracted from the isosurface image and blended with the complete ambient occlusion and ray cast image in order to generate the final rendering.

5.4.2 First Pass

In the first pass, we combine our ambient occlusion calculation, as described in Section 5.2, with a calculation that locates the start point of potential shadow rays. Our system casts a ray for every pixel in the viewport and calculates the ambient occlusion at each sampling point along the ray. Once the ray encounters the target isosurface, the rays location is refined through an iterative bisection procedure (Listing 5.1). The output from this stage of the rendering process is two images, one containing the integrated ambient occlusion calculations and the other containing the position of the target isosurface.

5.4.3 Second Pass

During the second pass, we use the isosurface location image generated in the first step to both calculate the Phong lighting for the isosurface and as a start point for shadow rays. The isosurface lighting for each pixel is calculated using the standard

Listing 5.1: Iterative bisection GLSL code.

```
if((voxel.a >= ISOVALUE) && (iterativeBisection_)){
    vec3 start = samplePos - tIncr*rayDirection;
    vec3 end = samplePos;

    for(int b = 0; b < 20; ++b) {
        vec3 testPos = (start+end)/2.0;
        if (getVoxel(volume_, volumeParameters_, testPos).a < ISOVALUE){
            start = testPos;
        } else {
            end = testPos;
        }
    }

    samplePos = (start+end)/2.0;
}
```

Phong lighting model and then the luminance is reduced if the region is shadowed. Our framework integrates jitter, adaptive sampling, and smooth shadow transitions to create an advanced shadow ray casting solution which allows the user numerous options to increase image quality and adjust rendering times. This stage of the pipeline blends the ambient occlusions calculations which were computed in the first pass with the isosurface lighting and shadowing.

5.4.4 NPR Additions

In addition to the lighting and shading calculations we have also integrated NPR line drawings into our DVR framework (Figure 5.6). We modify the second pass of our system to output an ambient and diffusely lit image of the isosurface as specular lighting has been shown to interfere with image space line extraction techniques. The overhead of generating this image is negligible as the lighting calculations are already required for the final rendered image. We then perform either Difference of Gaussian or Canny line extraction on this image. The extracted lines are then blended with the illuminated image to create an image which combines ray cast shadows, ambient occlusion, and NPR enhancement.

We have integrated Canny line extraction (Section 3.2.2.3) and Difference of Gaussian line extraction (Section 3.2.2.1) into our rendering system as both these techniques were shown to improve shape understanding (Section 3.4.2.2). Suggestive contour lines were not integrated into the system, despite their successful track record at improving shape perception, due to the significant increase in computation time required to extract these lines, which would have prevented our system from achieving real-time performance.

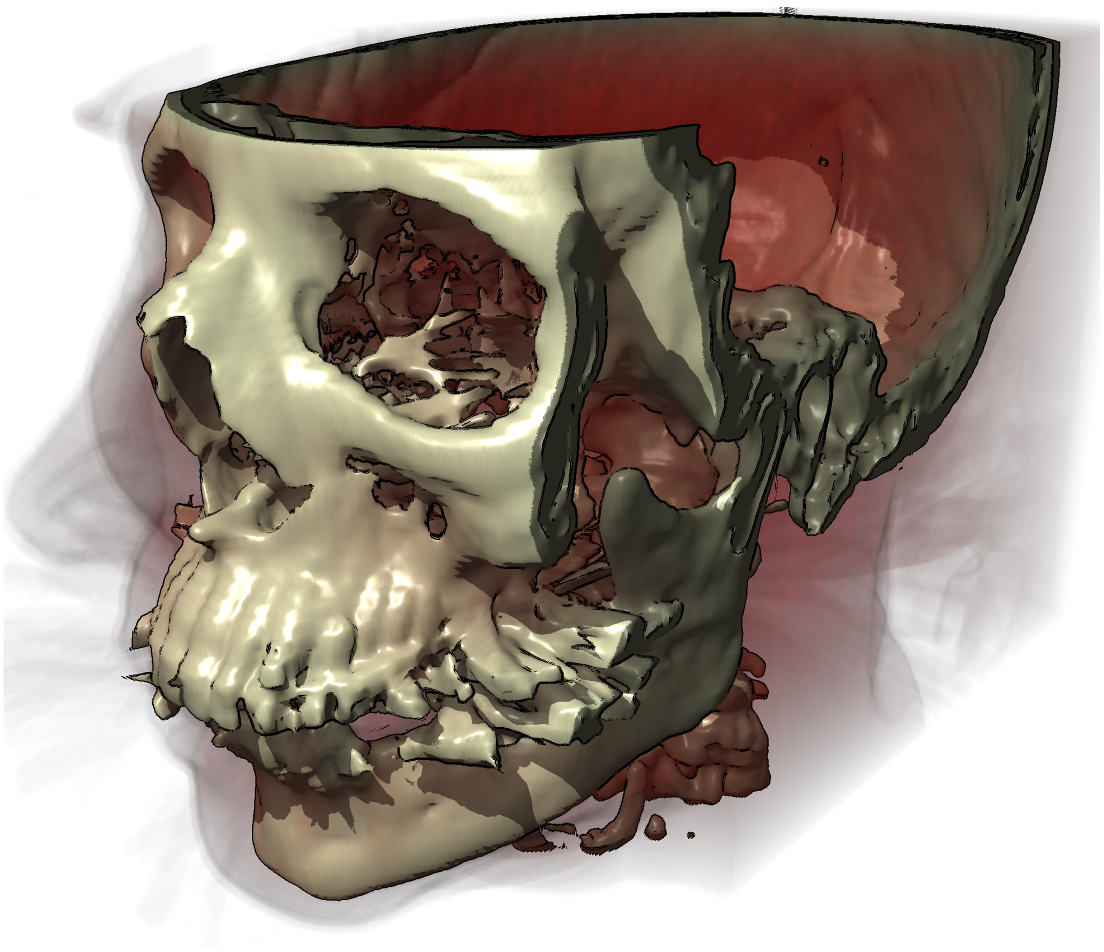


Figure 5.6: Difference of Gaussian line enhancement combined with ambient occlusion and ray cast shadows.

Table 5.3: Average frames per second for a 360° rotation around each dataset.

Dataset	Size	Phong (fps)	Our Method (fps)
Skull	256 x 256 x 113	34.26	24.49
Engine	256 x 256 x 256	19.63	15.75
Carp	256 x 256 x 512	14.09	17.16

5.5 Evaluation

5.5.1 Rendering Speed

We compared the rendering time of our method to a simple per-voxel Phong illumination model by calculating the average frames per second (fps) over 100 frames of a 360° rotation around three datasets. As can be seen from Table 5.3, our advanced illumination technique is approximately 28% slower for the Skull dataset, 19% slower for the Engine dataset and 21% faster for the Carp dataset (we are unsure as to why this speed up occurs but believe it may be due to the highly rectangular nature of the Carp dataset). Our advanced lighting technique is slightly slower than the same scene when lit using the Phong reflectance model, but this is expected due to the extra lighting and illumination calculations we perform which result in a higher quality final image.

We devised a test to determine how our illumination technique performs when compared to a modern state of the art lighting algorithm. Sundén et al. [SYR11] have proposed Image Plane Sweep Volume Illumination (IPSVI), a state of the art technique for global volume illumination which calculates ray cast lighting for the entire dataset. They compare their IPSVI technique to standard Phong illumination and show that their technique experiences an 85% reduction in frame rate. This compares favourably to our results which show an average reduction of 9% versus Phong lighting over the three datasets tested. Assuming both Phong implementations by Sundén et al. and ourselves incur a similar computational cost, we can compare the computationally efficiency of our technique to IPSVI by comparing the relative reduction in speed versus Phong illumination. Using this comparison, our local lighting technique is approximately five times faster than the global lighting solution of IPSVI.

5.5.2 User Experiments

In order to further test our technique, we devised a number of user experiments to gauge how a user's perception of shape, depth, and visual information changed with the addition of ray cast shadows and ambient occlusion. The experiments were conducted in two separate parts with different participants for each part. The first series of experiments consisted of the subjective evaluation, shape perception, and relative depth perception evaluation. Following this, we conducted a subsequent experiment to test absolute depth perception. All experiments were conducted on a 24 inch monitor in a well lit office environment.

For the first series of experiments, three user studies were devised. In two studies, we tested how the addition of ray cast shadows affect a user's perception of both shape and relative depth and in the other study, we tested whether the addition of ambient occlusion affects a user's perception of the amount of visual information in the scene. For all trials, the light source was positioned at 16.7° to the left of vertical and at 26.6° above horizontal as this position has been shown by other researchers to be the best location for perception of shape [OBA08, SP98, LB00]. Three diverse datasets were used in the experiment: scans of a male head, a brain, and a set of knees. Each image was a focus and context rendering which displayed a target isosurface and a semi-transparent context. Eighteen naive participants took part in each experiment (12M-6F). The participants ranged in age from 21 to 31 and came from a wide variety of backgrounds and occupations. Twelve participants rated themselves as having a novice knowledge of computer graphics, two participants rated themselves as experienced and four participants rated themselves as experts.

In the subsequent absolute depth perception experiment, users were shown an image with a single point highlighted and asked to estimate the absolute depth of the point using a slider [LR11]. In this experiment, we were only interested in testing the effect of ray cast shadows on the focus surface, and as such, we disabled the rendering of the context layer and only rendered the focus isosurface. This experiment had 14 participants (11M-3F), ranging in age from 16 to 62. Five participants rated themselves as having a novice knowledge of computer graphics, three participants rated themselves as experienced and six participants rated themselves as experts. In this experiment, five datasets were shown to participants: a skull, heart, clouds, bonsai tree, and a randomly generated sine wave dataset (Figure 5.11). The light source was placed in the same upper left position as in the first set of trials and the experiment was conducted on the same computer monitor and in the same environment.

Results from all the user tests were collected and an ANalysis Of VAriance (ANOVA) was performed. If a confidence level of $p < 0.05$ was reached, post-hoc Newman-Keuls analysis was performed at the same significance level. For the subjective evaluation, a single sample t-test was performed to compare the results to chance.

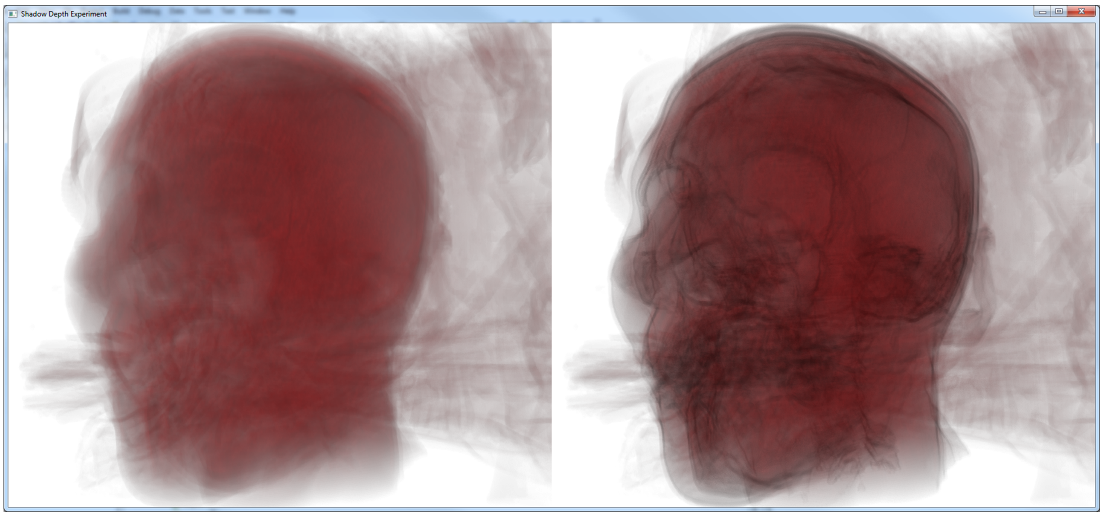


Figure 5.7: The evaluation program used by participants in the ambient occlusion user study. The participants were asked to select which of the two images they felt contained the most visual information.

5.5.2.1 Subjective Evaluation

Experiment Setup In this experiment we wanted to test if our ambient occlusion technique was preferred over a standard rendering solution. Participants were shown two side by side images of the same dataset, from the same camera position, with Phong illumination enabled. The only difference between the two images was that one image had our ambient occlusion calculation enabled and the other did not. Participants were asked to select which image they felt had the most visual information. The users used the mouse to select an image by pressing the left mouse button to select the left image and the right mouse button to select the right image.

For each dataset, 2 pairs of images were generated, all from the same viewpoint. One pair of images contained no target isosurface and purely ambient occlusion calculations, the other set of images contained a target isosurface combined with ray cast shadows and our ambient occlusion technique. Each pair of images was tested 3 times, for a total of 27 trials. The order in which the sets of images were shown and the left/right ordering of each individual pair of images was randomised to eliminate any bias.

Results In order to evaluate if participants were answering significantly above chance level (i.e., significantly different from 50%), we averaged the results for each isovalue and performed a single sample t-test. We found that participants preferred our ambient occlusion method both when an isosurface was present ($t(17) = 3.387, p = 0.004$) and significantly more so when no isosurface was rendered ($t(17) = 15.912, p < 0.0001$).

For each trial, we also averaged the users selection for each of the three repetitions and then performed an ANalysis Of VAriance (ANOVA) on the results where the conditions were *Dataset*(3) and *Isovalue*(2). It was found that there was a main effect of isovalue ($F(1,17) = 25.576, p = 0.0001$) but no effect of dataset ($F(2,34) = 0.979, p = 0.386$).

Discussion The results show that participants selected our ambient occlusion technique significantly above chance, thus showing that our computationally efficient ambient occlusion method can improve the visual information contained within a dataset for very little extra computational cost.

In addition to testing our technique on its own, we tested it with a large amount of the dataset covered by a target isosurface on which ray cast shadows were displayed. By setting up the images in this way, only a very small amount of the dataset receives our ambient occlusion technique, therefore, making our technique far less noticeable. We wanted to test if this worst case scenario for our ambient occlusion method resulted in a statistically significant result. Analysis of the results show, that for this scenario, our ambient occlusion technique is also favoured above chance.

The ANOVA analysis shows that there is a main effect of isovalue, therefore, indicating that there is a statistically significant difference between how well our ambient occlusion technique performs in this worse case scenario when compared to a more plausible use case. In general, our ambient occlusion technique was shown to significantly increase the amount of visual information contained within an image, with a larger effect reported when no isosurface was rendered. Furthermore, our ambient occlusion technique highlights differences in density in the context region which enhances internal structures and makes them easier to discern. Figure 5.1 shows how our ambient occlusion technique enhances the context region and makes the internal valving of the engine block (which was hidden in the standard rendering) visible.

5.5.2.2 Shape Perception

Experiment Setup In this experiment, participants were asked to align a gauge which had been placed on a static image, this protocol has been described in previous experiments as a method for testing perception of shape [CSD+09, CRD10, ŠPV11, OBA08]. Participants were shown a series of images from the system and asked to rotate gauges which overlaid the image to match the surface normal of the isosurface at that point. Participants had no control of the gauge position, only its orientation. The alignment of the gauge was controlled using the mouse and the space bar was used to indicate that the participant was happy with the gauge alignment and ready to move on to the next

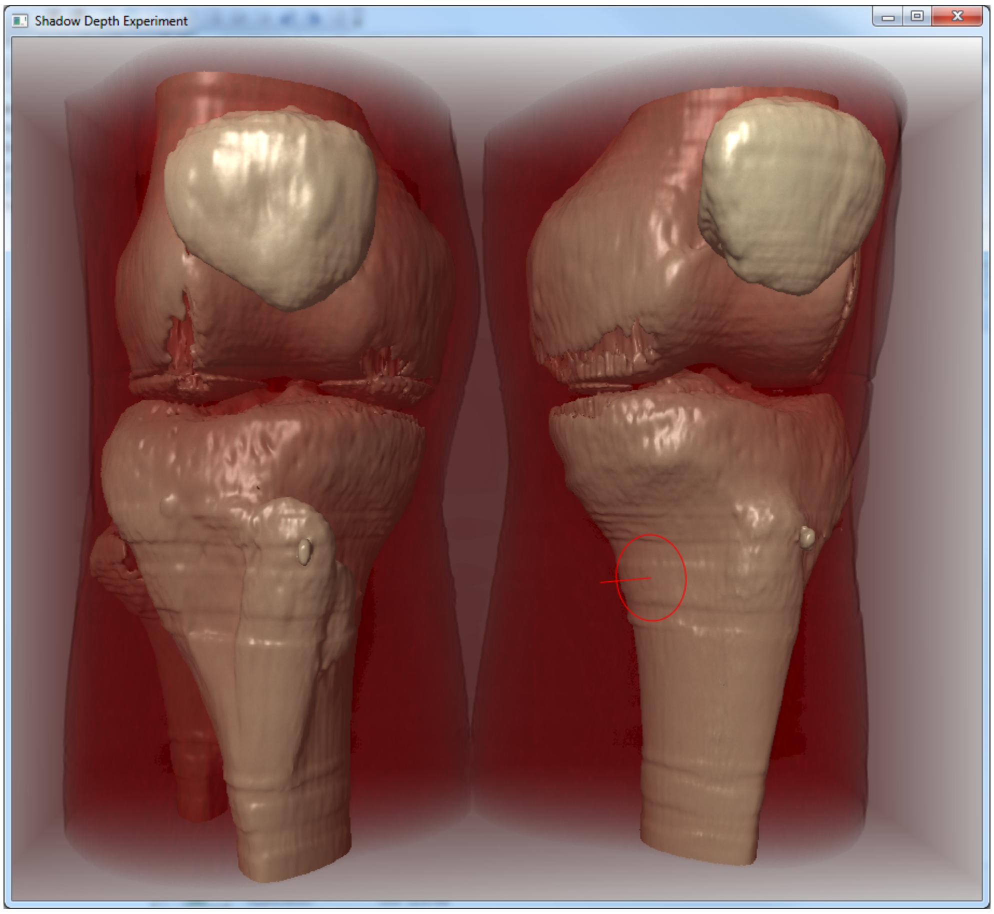


Figure 5.8: The evaluation program used for the shape perception experiment.

image. Each gauge was drawn as an ellipse and a line as seen in Figure 5.8.

Two images were generated for each dataset, one with and one without ray cast shadows enabled. Both images had ambient occlusion enhancement and Phong illumination enabled. Each image was shown 8 times, with different gauge placements, resulting in a total of 48 trials per participant. Half of the gauges were placed in shadowed regions and half in lit regions. The placement of the gauges for the shadowed and unshadowed images of each dataset was kept constant. The order of the images shown to each participant was randomised to prevent any ordering bias.

Results For each trial, the angle between the correct surface normal and the user estimated surface normal was calculated. The average angle was then calculated for each dataset, with and without ray cast shadows enabled, and an ANalysis Of VAriance (ANOVA) was then performed on the results where the conditions were *Dataset*(3), *Ray Cast Shadows*(2) and *In Shadow*(2). It was found that there was a main effect of *Dataset* ($F(2,34) = 4.846, p = 0.014$) and *In Shadow* ($F(1,17) = 81.669, p < 0.0001$)

but no effect of *Ray Cast Shadows* ($F(1,17) = 0.399, p = 0.536$). Post-hoc analysis was then performed using a standard Newman-Keuls test for pairwise comparison between means. Participant error was shown to be significantly higher for the brain dataset when compared to the other datasets ($p < 0.015$). In addition, participant error was higher in shadowed regions ($p < 0.012$) when compared to all other regions.

Discussion Previous work [ŠPV11] has shown that chromatic shadows can affect shape perception when compared to standard shadows, but to our knowledge no work has investigated the effect that shadows have on shape perception when compared to Phong lighting. We postulated that the addition of ray cast shadows would cause participants to experience a reduction in shape perception due to the addition of shadowed regions causing distractions and also due to the reduction in luminance that shadowed areas receive. In contrast, our experiment shows that the addition of ray cast shadows has no statistically significant effect on the participants shape perception. This shows that we can add ray cast shadow calculations to a volume rendering scene without reducing a user's understanding of the shape of the dataset.

Post-hoc analysis shows that there is a statistically significant difference between the accuracy of participants for gauges placed in shadowed areas. Interestingly, this effect remains when shadows are disabled and gauges are placed in regions where shadows would be cast if the shadows were rendered. This shows that the addition of shadows is not the sole reason for less accurate results for gauges placed in shadowed regions. These regions receive statistically significant less accurate results even when no shadows are displayed.

Post-hoc analysis also shows that there is a significant reduction in accuracy for the brain dataset. A number of participants commented that they recognised the knee and skull datasets and could use prior knowledge in order to assist in gauge alignment. In contrast, they found that they had no prior knowledge of the brain dataset and were relying solely on the images presented to them in order to estimate gauge orientation.

5.5.2.3 Relative Depth Perception

Experiment Setup The primary aim of volume visualisation and the addition of ray cast shadows is to improve user understanding of the dataset. Other researchers have shown that the addition of certain types of shadows can enhance depth perception in an image [Pue89, LR11]. In this relative depth experiment, we wanted to investigate whether this hypothesis held true for focus and context volume rendering. For this task, we designed an experiment where two points were highlighted on a static image and

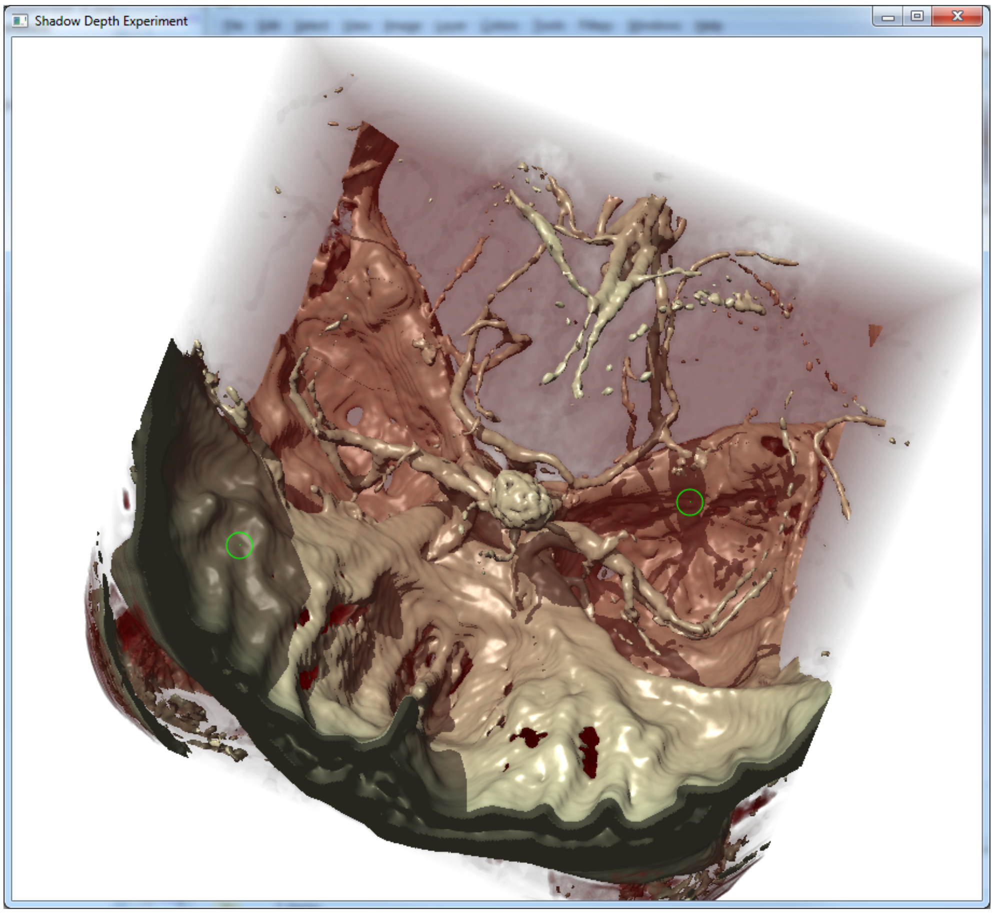


Figure 5.9: The evaluation program used for the relative depth perception experiment.

the participant was asked to select which point they felt was closest to the screen [LR11, ŠPV11]. The selection of a point was controlled by the mouse and the space bar was used to indicate that the participant was happy with the selection and ready to move on to the next trial. Each point was drawn as a single pixel with a surrounding larger circle in order to make it easier for the user to locate the pixel.

For this experiment, we used the same images as were used in the shape perception experiment (Section 5.5.2.2). Each image was shown 6 times, with different point placements, resulting in a total of 36 trials per participant. Two of the trials had both points placed in lit regions, two of the trials had both points placed in shadowed regions and the remaining two trials had one point in a lit region and one point in a shadowed region. The placement of the points for images with ray cast shadows enabled and disabled was kept constant. The order of the images shown to each participant was randomised to prevent any ordering bias.

Results For each trial, the correctness of the users selection was averaged for each dataset, with and without ray cast shadows enabled, and an ANalysis Of VAriance (ANOVA) was then performed on the results where the conditions were *Dataset*(3) and *Ray Cast Shadows*(2). A main effect of *Dataset* was found ($F(2, 34) = 5.982, p = 0.006$) but no main effect of *Ray Cast Shadows* ($F(1, 17) = 0.265, p = 0.613$).

Discussion Interestingly, this experiment showed a statistically significant decrease in accuracy for the head dataset which contrasts with the previous experiments where participants had lower accuracy when analysing the brain dataset and no significant change when analysing the head dataset. We believe that this reduction in accuracy for the head dataset is due to the smoothness of the target surface which results in smaller changes in depth between points.

Furthermore, this experiment did not show any statistical significant increase in depth perception due to the addition of ray cast shadows. A number of users mentioned that they found the depth experiment relatively easy. Analysis of the results show that the average accuracy (89%) for this experiment was very high which may have contributed to the non significant result. Some participants received perfect scores for some images for both ray cast images and standard renderings which, due to the binary nature of the experiment, may have masked a possible difference between the two renderings.

5.5.2.4 Absolute Depth Perception

Experiment Setup In our previous relative depth experiment, we were unable to show any statistically significant increase in depth perception due to the addition of ray cast shadows. In discussions with our participants, we were informed that a large number of users used prior knowledge of basic anatomy in order to determine depth information for our datasets. We believed that the reason other researchers have shown an increase in depth perception with the addition of shadows is due to their use of unfamiliar datasets which did not allow participants to utilise prior knowledge [Pue89, LR11, ŠPV11]. Therefore, we designed a new experiment to test absolute depth perception using a set of unfamiliar datasets.

In this experiment, users were shown an image with a single point highlighted and asked to estimate the absolute depth of the point using a slider. Participants were told to estimate the nearest and furthest points in the image and position the slider relative to these points [LR11]. The point was drawn as a single pixel with two surrounding larger circles to make it easier for the user to locate the pixel. The user manipulated the slider using the mouse, and the space bar was used to indicate that they were happy with the

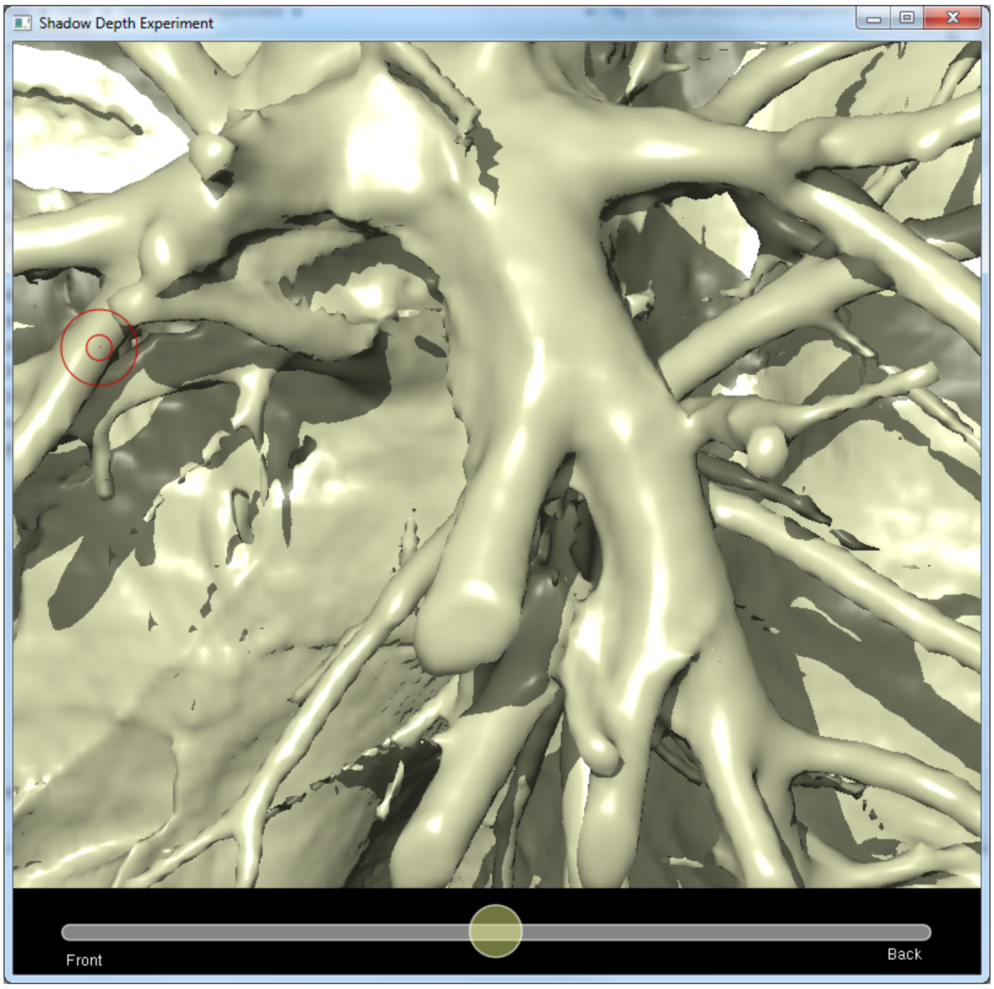
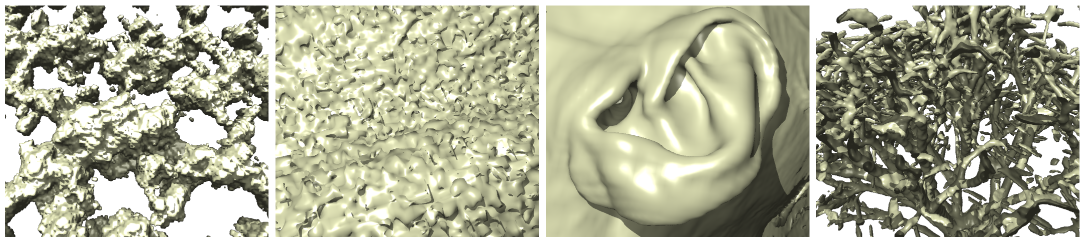


Figure 5.10: The evaluation program used for the absolute depth perception experiment showing the heart dataset.

slider's position and ready to move on to the next image. In this experiment, five datasets were shown to participants: a skull, heart, clouds, bonsai tree, and a randomly generated sine wave dataset. Each image was shown 25 times with different point positions and then the same scene was displayed with shadows enabled and using the same point positions resulting in a total of 250 trials per participant. The order of the images shown to each participant was randomised to prevent any ordering bias. As we were only interested in testing the effect of ray cast shadows, we disabled ambient occlusion calculations for these images and only displayed the target isosurface which was rendered with Phong illumination enabled.

Results For each trial, the error between the user selected depth and the actual depth was calculated. This was averaged across each image, both shadowed and unshadowed, and an ANalysis Of VAriance (ANOVA) was then performed on the results where the



(a) Clouds. (b) Random sine wave. (c) Skull. (d) Bonsai tree.

Figure 5.11: Some of the datasets shown to participants in the absolute depth perception experiment.

conditions were *Dataset*(5) and *Ray Cast Shadows*(2). There was a main effect of *Ray Cast Shadows* ($F(1, 13) = 5.246, p = 0.039$) with the addition of shadows reducing depth perception and no main effect of *Dataset* ($F(4, 52) = 2.486, p = 0.055$).

Discussion Other researchers [Pue89] have shown that the addition of shadows can produce the sensation of depth, but our results show that this perception of depth does not result in increased accuracy at estimating depth for focus and context rendering.

5.6 Discussion

We have presented a technique for focus and context volume rendering which produces high quality, ray cast shadows for a target isosurface within a volume dataset while simultaneously providing computationally efficient ambient occlusion shading for the surrounding volume. Our technique requires no pre-processing, does not significantly increase memory requirements, is compatible with ray cast rendering systems, and achieves real-time frame rates. In addition, we extended our technique to incorporate high quality smooth shadow transitions, adaptive shadow ray sampling, shadow ray jitter and demonstrate how to perform early ray termination for shadow rays. We also combine our focus and context illumination approach with NPR line drawings in order to further enhance the focus isosurface.

Through a series of user experiments, we show that our technique can significantly increase the visual information in an image without reducing shape or relative depth perception. This increase in visual information improves the users ability to perceive internal structures in the final image, thus providing more context information, allowing for easier interaction with the dataset. Our results show that focus and context volume rendering can be enhanced by the addition of our computationally efficient ambient occlusion enhancement on the context area of the rendering and that our ray cast shadows do not reduce shape perception of the focus surface. However, the results of our user study indicate that the addition of ray cast shadows has no effect on relative depth perception and worsens absolute depth perception. Therefore, the addition of ray cast shadows, despite their usefulness in other techniques is not strongly recommended for enhancing user understanding of focus and context volume rendering. Interestingly, numerous participants commented that they preferred the renderings which had ray cast shadows enabled. While the results of our experiment indicate that shadows are not recommended for enhancing user understanding, they seem to be visually pleasing and preferred by users from a purely aesthetic point of view.

Chapter 6

Conclusion

6.1 Summary of Contributions

The goal of this thesis was to provide a focus and context volume rendering framework which allows for enhanced renderings. Through analysis of the results of multiple user experiments and the creation of novel rendering techniques we have developed such a framework.

In Chapter 3, we investigate possible NPR stylisations to enhance focus and context renderings. We test a number of different NPR styles ranging from line drawings to stylised shading and perform a detailed user experiment on how stylisation affects shape perception. Our results show that NPR stylisations can significantly enhance user shape perception and allow us to determine which styles provide the greatest benefit. In addition, we show that NPR stylisations applied to the context region have no effect on shape perception of the focus surface.

In Chapter 4, we describe a technique we developed for improving the rendering speed of focus and context volume rendering by taking advantage of the principle of temporal coherency. Our technique is able to dramatically decrease the computation time for focus and context rendering and can be easily integrated into existing volume rendering frameworks. In addition, we suggest a series of improvements to our technique which allow for increased rendering quality. We verify our approach by performing an exhaustive comparison to a standard rendering system.

Chapter 5 describes a technique we developed which allows for real-time illumination of volume rendering by performing adaptive illumination based on the importance of each area in the volume dataset. This approach combines both ambient occlusion and ray cast shadows into an integrated rendering system. Through an in-depth series of user experiments, we prove that our ambient occlusion technique can increase the amount of information in the image, thus enhancing internal structures. In addition, our experiments show that despite the obvious visual appeal of ray cast shadows they do not improve functional performance in tasks such as depth estimation.

The techniques developed in this thesis and the insights gained from our user experiments have allowed us to create a real-time focus and context volume rendering solution which combines NPR stylisation and adaptive illumination calculations. Our system allows the user to control multiple parameters to adjust both style and quality in order to generate a final enhanced rendering. The work in this thesis allows us to recommend the use of NPR line drawings and ambient occlusion lighting calculations for the enhancement of focus and context volume rendering. In addition, despite their use in other rendering styles we do not recommend ray cast shadows for enhancing focus and context volume renderings despite their aesthetic appeal. Finally, we can recommend image space adaptive rendering as a novel technique for improving the rendering speed of focus and context rendering.

6.2 Future Work

6.2.1 Image Space Adaptive Rendering Enhancements

In Chapter 4, we describe a technique for improving rendering speed by generating an importance map which utilises information from the previously rendered frame. This importance map will not exactly correspond to the current frame due to user interaction causing rotations and translations, and therefore we dilate the map in order to ensure that no regions are undersampled. This dilation approach reduces undersampling but comes with an associated performance cost as it increases the number of regions that are inadvertently sampled at high quality.

There is scope to investigate an adaptive dilation approach which modifies the dilation filter based on the type of interaction the user has with the dataset. For example, when the user is zooming out from the dataset we know that the important region is getting smaller in every subsequent frame, therefore, there is no need to perform dilation and in fact applying a dilation filter in this situation reduces performance without improving image quality. When the user is rotating around the dataset, new features will be more likely to appear along the axis of rotation, therefore, dynamically orientating the filter to the rotation axis could improve performance.

In addition, we performed an informal test on the effectiveness of the different importance measures that could be used to generate the importance map for our image space adaptive rendering technique. A detailed study on their effectiveness versus their computational cost, in order to determine the optimum technique for importance map generation, would be useful.

6.2.2 User Experimentation

More exhaustive user experiments could be performed to verify the effectiveness of the techniques developed in this thesis. We limited the scope of our evaluation to analysing perceptual changes, however, task based techniques or eye tracking experiments could be used to verify our results. In addition, we wish to perform a number of user experiments involving domain experts to test if the effectiveness of the techniques we have developed varies based on user knowledge.

In Chapter 3, we investigate NPR enhancement for the focus layer of two-level volume rendering and show that NPR line enhancement can improve shape perception of the focus layer. In Chapter 5, we show that our ambient occlusion technique can enhance the visibility of internal structures in the context layer. Our system combines these two techniques together and allows for a rendering which applies line stylisations to the focus layer and ambient occlusion calculation to the context layer. Through user studies, we have proven that on their own, these techniques can enhance focus and context rendering, but we have not investigated if this improvement is maintained when the two techniques are combined. It could be assumed that since the two techniques are applied to separate regions of the volume dataset that their effects are maintained even when both techniques are enabled at the same time, but due to the overlapping nature of volume data we can not state this explicitly and a user experiment to investigate this would be beneficial.

6.2.3 Domain Specific Use

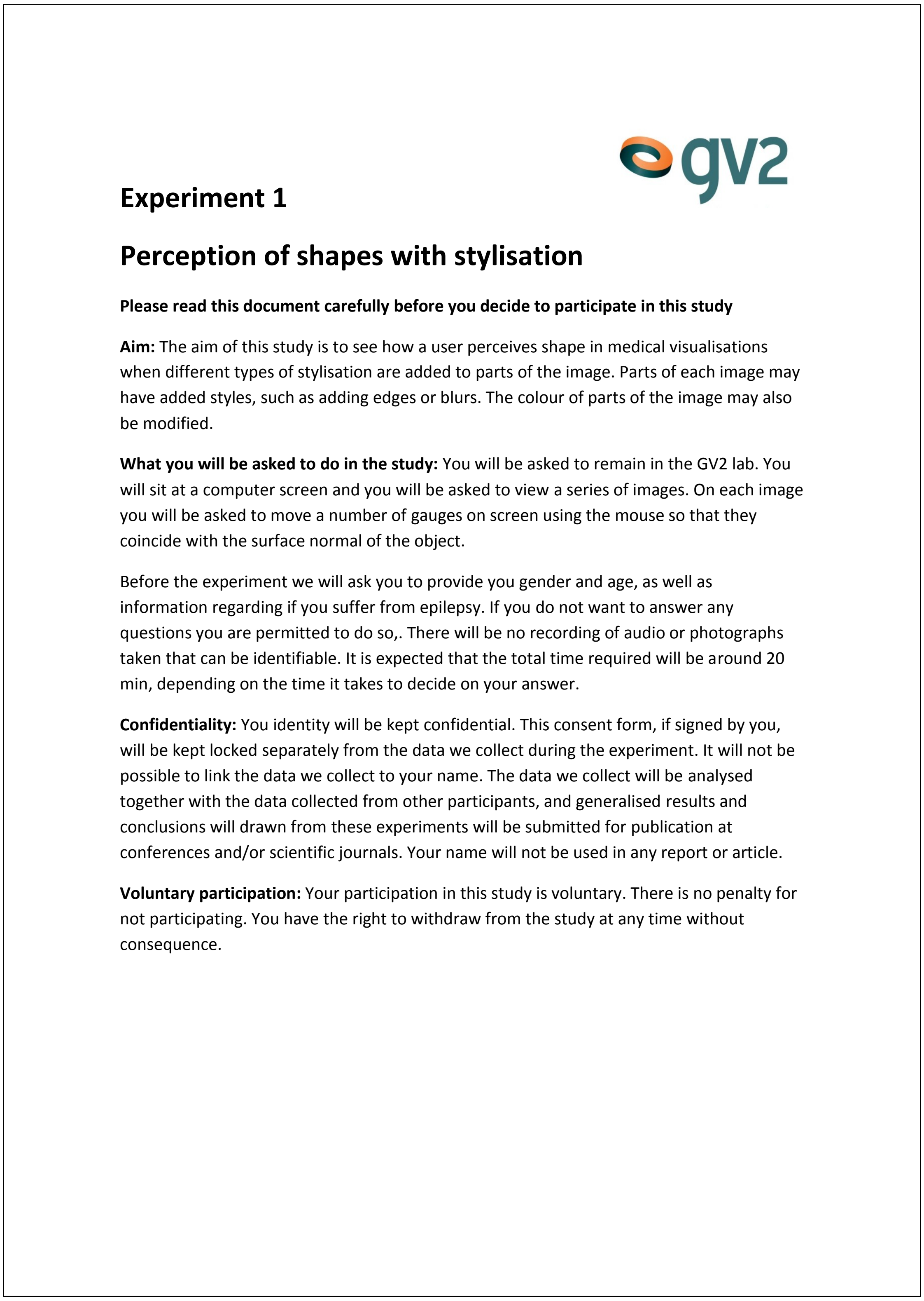
Studies have shown that (in a hospital setting) increasing patient understanding of their care can improve clinical outcomes [Ada10]. Unfortunately, the vast majority of patients do not understand their plan of care [OKL+10]. Given that the techniques developed in this thesis have been proven to enhance understanding of volume renderings for novice users, we believe that their is scope to develop a system which uses our techniques to enhance the rendering of MRI and CT patient data. These focus and context NPR renderings could be used in a clinical setting in order to allow doctors to more easily explain the procedures that their patients are about to undergo.

Medical textbooks have used focus and context illustrations in anatomy education for many years [Net97]. These diagrams typically rely on hand drawn, artist illustrations that are both time consuming and expensive to produce. Therefore, there is scope to develop an automatic, focus and context, illustrative rendering system for anatomy education.

Appendix A

Experiment Information Sheets

This section lists the experiment information sheets that were given to participants in our user experiments.



Experiment 1

Perception of shapes with stylisation

Please read this document carefully before you decide to participate in this study

Aim: The aim of this study is to see how a user perceives shape in medical visualisations when different types of stylisation are added to parts of the image. Parts of each image may have added styles, such as adding edges or blurs. The colour of parts of the image may also be modified.

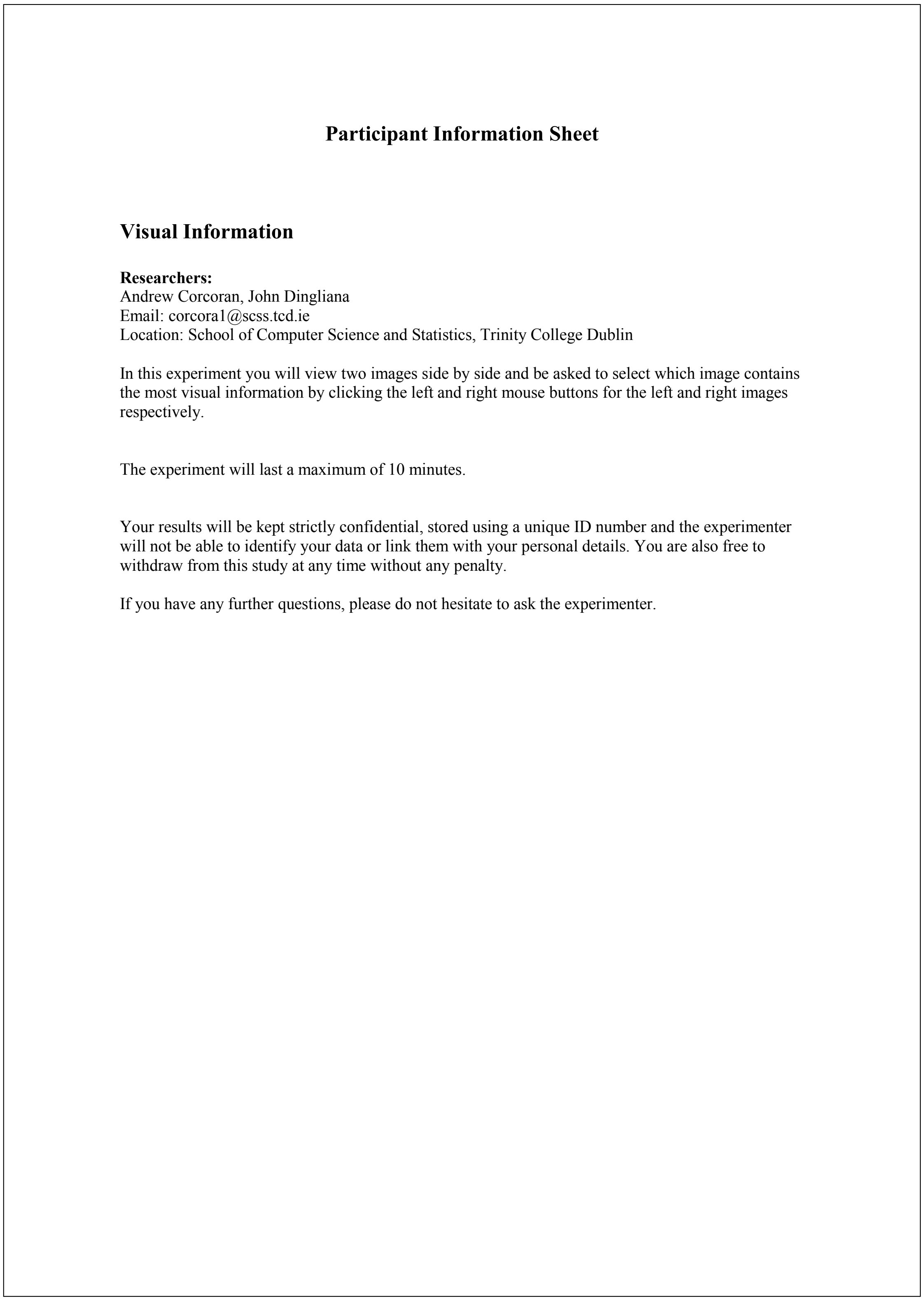
What you will be asked to do in the study: You will be asked to remain in the GV2 lab. You will sit at a computer screen and you will be asked to view a series of images. On each image you will be asked to move a number of gauges on screen using the mouse so that they coincide with the surface normal of the object.

Before the experiment we will ask you to provide you gender and age, as well as information regarding if you suffer from epilepsy. If you do not want to answer any questions you are permitted to do so,. There will be no recording of audio or photographs taken that can be identifiable. It is expected that the total time required will be around 20 min, depending on the time it takes to decide on your answer.

Confidentiality: You identity will be kept confidential. This consent form, if signed by you, will be kept locked separately from the data we collect during the experiment. It will not be possible to link the data we collect to your name. The data we collect will be analysed together with the data collected from other participants, and generalised results and conclusions will drawn from these experiments will be submitted for publication at conferences and/or scientific journals. Your name will not be used in any report or article.

Voluntary participation: Your participation in this study is voluntary. There is no penalty for not participating. You have the right to withdraw from the study at any time without consequence.

Figure A.1: Information sheet given to participants in the NPR shape perception experiment.



Participant Information Sheet

Visual Information

Researchers:
Andrew Corcoran, John Dingliana
Email: corcora1@scss.tcd.ie
Location: School of Computer Science and Statistics, Trinity College Dublin

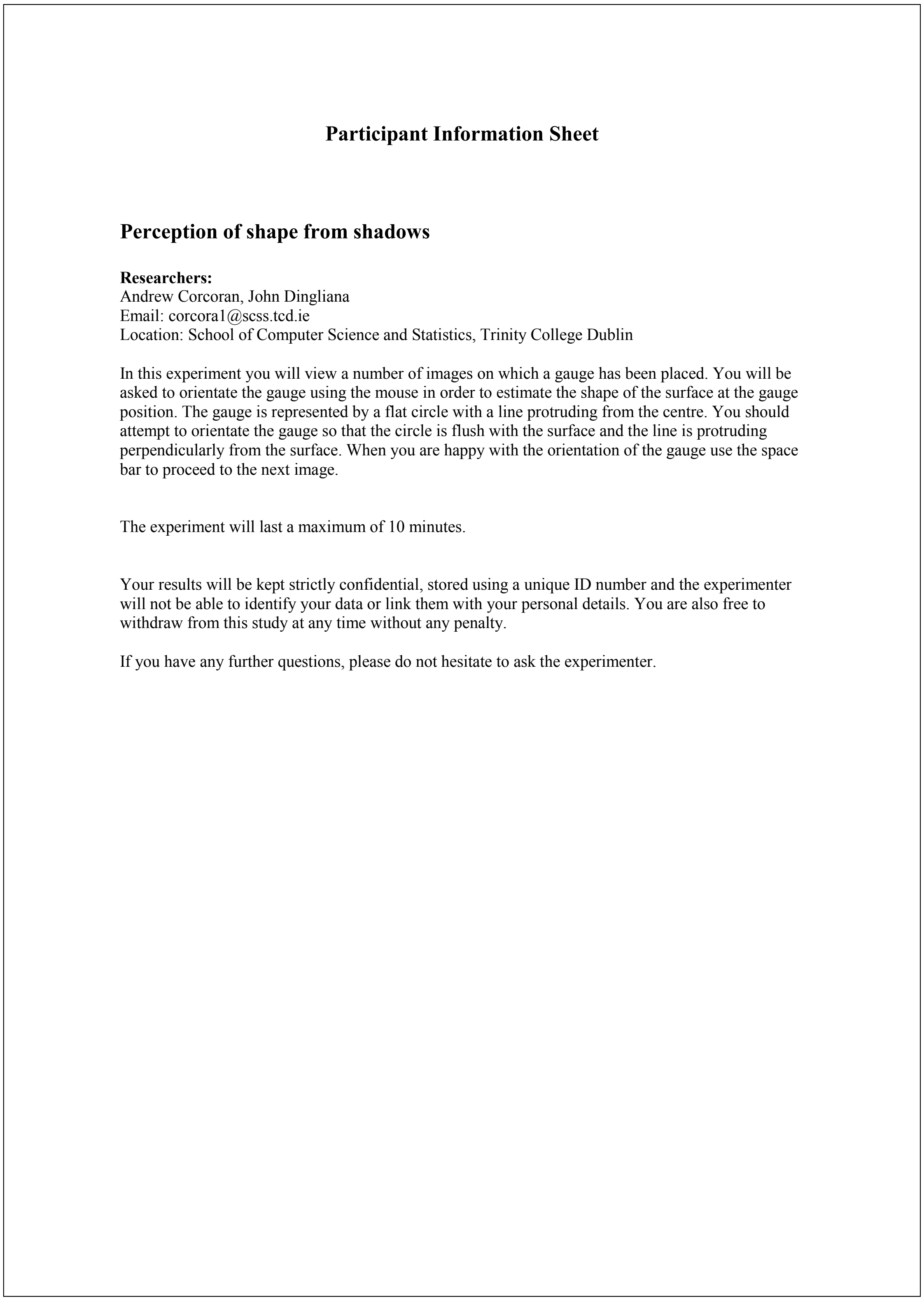
In this experiment you will view two images side by side and be asked to select which image contains the most visual information by clicking the left and right mouse buttons for the left and right images respectively.

The experiment will last a maximum of 10 minutes.

Your results will be kept strictly confidential, stored using a unique ID number and the experimenter will not be able to identify your data or link them with your personal details. You are also free to withdraw from this study at any time without any penalty.

If you have any further questions, please do not hesitate to ask the experimenter.

Figure A.2: Information sheet given to participants in the ambient occlusion subjective evaluation experiment.



Participant Information Sheet

Perception of shape from shadows

Researchers:
Andrew Corcoran, John Dingliana
Email: corcora1@scss.tcd.ie
Location: School of Computer Science and Statistics, Trinity College Dublin

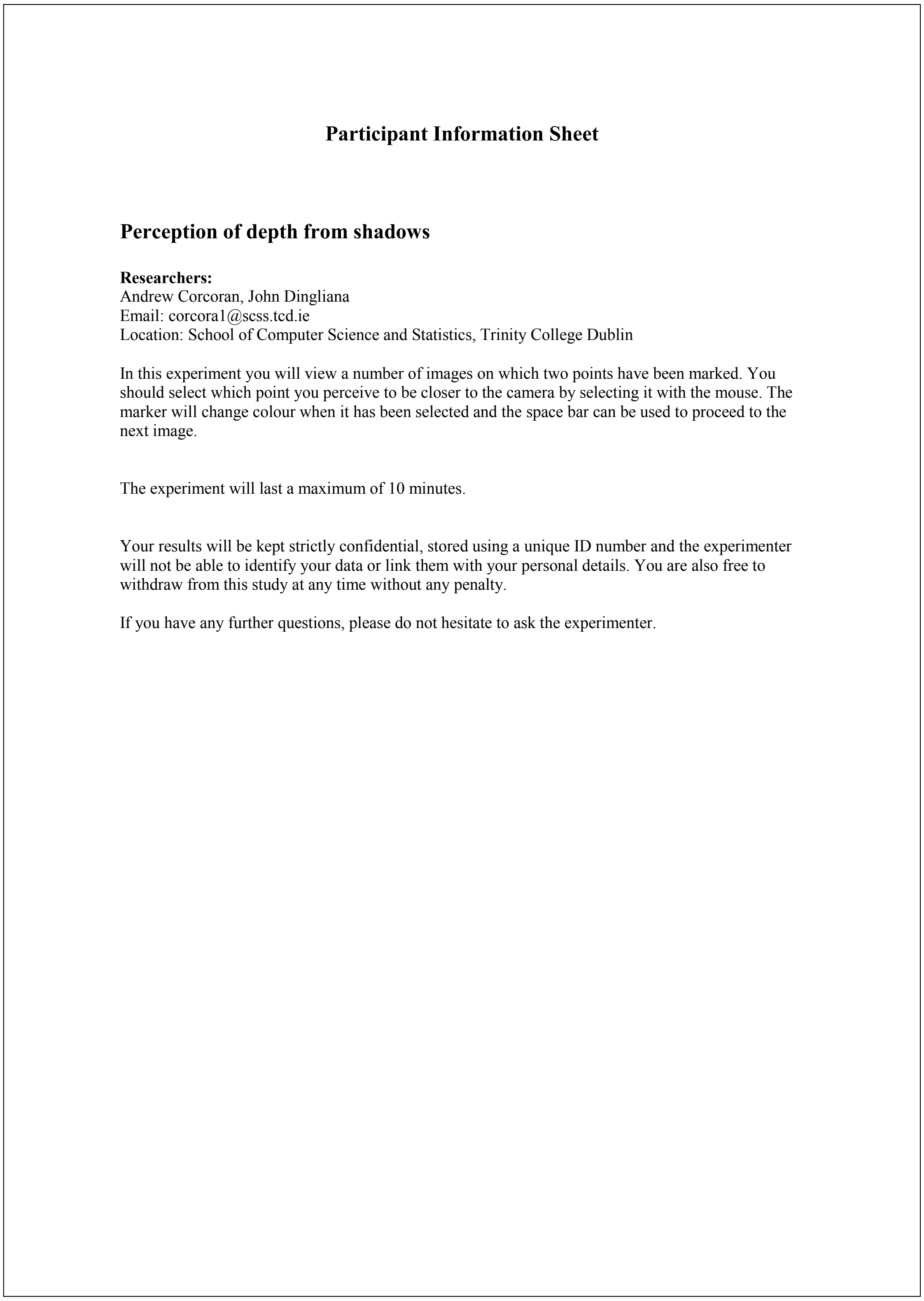
In this experiment you will view a number of images on which a gauge has been placed. You will be asked to orientate the gauge using the mouse in order to estimate the shape of the surface at the gauge position. The gauge is represented by a flat circle with a line protruding from the centre. You should attempt to orientate the gauge so that the circle is flush with the surface and the line is protruding perpendicularly from the surface. When you are happy with the orientation of the gauge use the space bar to proceed to the next image.

The experiment will last a maximum of 10 minutes.

Your results will be kept strictly confidential, stored using a unique ID number and the experimenter will not be able to identify your data or link them with your personal details. You are also free to withdraw from this study at any time without any penalty.

If you have any further questions, please do not hesitate to ask the experimenter.

Figure A.3: Information sheet given to participants in the shadows shape perception experiment.



Participant Information Sheet

Perception of depth from shadows

Researchers:
Andrew Corcoran, John Dingliana
Email: corcora1@scss.tcd.ie
Location: School of Computer Science and Statistics, Trinity College Dublin

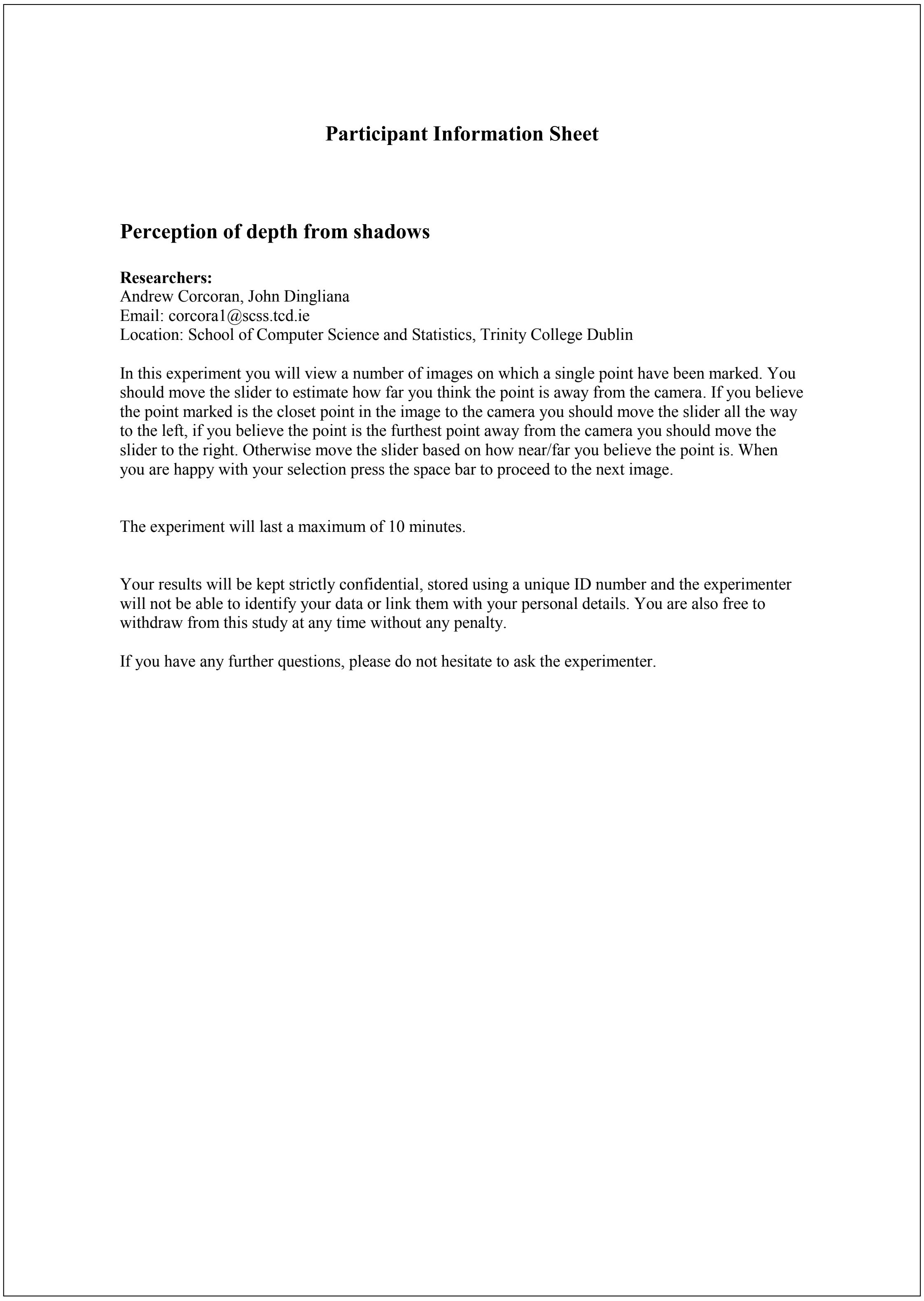
In this experiment you will view a number of images on which two points have been marked. You should select which point you perceive to be closer to the camera by selecting it with the mouse. The marker will change colour when it has been selected and the space bar can be used to proceed to the next image.

The experiment will last a maximum of 10 minutes.

Your results will be kept strictly confidential, stored using a unique ID number and the experimenter will not be able to identify your data or link them with your personal details. You are also free to withdraw from this study at any time without any penalty.

If you have any further questions, please do not hesitate to ask the experimenter.

Figure A.4: Information sheet given to participants in the shadows relative depth perception experiment.



Participant Information Sheet

Perception of depth from shadows

Researchers:
Andrew Corcoran, John Dingliana
Email: corcora1@scss.tcd.ie
Location: School of Computer Science and Statistics, Trinity College Dublin

In this experiment you will view a number of images on which a single point have been marked. You should move the slider to estimate how far you think the point is away from the camera. If you believe the point marked is the closet point in the image to the camera you should move the slider all the way to the left, if you believe the point is the furthest point away from the camera you should move the slider to the right. Otherwise move the slider based on how near/far you believe the point is. When you are happy with your selection press the space bar to proceed to the next image.

The experiment will last a maximum of 10 minutes.

Your results will be kept strictly confidential, stored using a unique ID number and the experimenter will not be able to identify your data or link them with your personal details. You are also free to withdraw from this study at any time without any penalty.

If you have any further questions, please do not hesitate to ask the experimenter.

Figure A.5: Information sheet given to participants in the shadows absolute depth perception experiment.

Appendix B

Code Listings

Listing B.1: Difference of Gaussian CG code.

```
float4 DoG_ps (half2 texCoord : TEX0,
      uniform sampler2D Image,
      uniform float threshold) : COLOR
{
  half THRESHOLD;
  THRESHOLD = threshold;//0.0023;
  float4 OUT;
  float2 screensize;
  float pixelSize = 0.0031;
  screensize.x = pixelSize/7;
  screensize.y = pixelSize/4;
  half4 c  = tex2D(Image, texCoord);
  half4 bl = tex2D(Image, texCoord + float2(-1*screensize.x, -1*screensize.y));
  half4 l  = tex2D(Image, texCoord + float2(-1*screensize.x,0*screensize.y));
  half4 tl = tex2D(Image, texCoord + float2(-1*screensize.x,1*screensize.y));
  half4 t  = tex2D(Image, texCoord + float2(0*screensize.x,1*screensize.y));
  half4 ur = tex2D(Image, texCoord + float2( 1*screensize.x,1*screensize.y));
  half4 r  = tex2D(Image, texCoord + float2(1*screensize.x,0*screensize.y));
  half4 br = tex2D(Image, texCoord + float2(1*screensize.x,-1*screensize.y));
  half4 b  = tex2D(Image, texCoord + float2(0*screensize.x,-1*screensize.y));

  half4 luu = tex2D(Image, texCoord + float2(-2*screensize.x,-2*screensize.y));
  half4 lu  = tex2D(Image, texCoord + float2(-2*screensize.x,-1*screensize.y));
  half4 lm  = tex2D(Image, texCoord + float2(-2*screensize.x,0*screensize.y));
  half4 ld  = tex2D(Image, texCoord + float2(-2*screensize.x,1*screensize.y));
  half4 ldd = tex2D(Image, texCoord + float2(-2*screensize.x,2*screensize.y));

  half4 ruu = tex2D(Image, texCoord + float2(2*screensize.x,-2*screensize.y));
  half4 ru  = tex2D(Image, texCoord + float2(2*screensize.x,-1*screensize.y));
  half4 rm  = tex2D(Image, texCoord + float2(2*screensize.x,0*screensize.y));
  half4 rd  = tex2D(Image, texCoord + float2(2*screensize.x,1*screensize.y));
  half4 rdd = tex2D(Image, texCoord + float2(2*screensize.x,2*screensize.y));

  half4 ult = tex2D(Image, texCoord + float2(-1*screensize.x,-2*screensize.y));
  half4 uu  = tex2D(Image, texCoord + float2(0*screensize.x,-2*screensize.y));
  half4 urt = tex2D(Image, texCoord + float2(1*screensize.x,-2*screensize.y));

  half4 blt = tex2D(Image, texCoord + float2(-1*screensize.x,2*screensize.y));
  half4 bb  = tex2D(Image, texCoord + float2(0*screensize.x,2*screensize.y));
  half4 brt = tex2D(Image, texCoord + float2(1*screensize.x,2*screensize.y));

  //Gaussian Blurs
  //sigma=1;
  half4 firstfilter = ((luu + ldd + ruu + rdd) + (4*(ult+urt+blt+brt+lu+ld+ru+rd)) +↩
       (7*(bb+uu+lm+rm)) + (16*(tl+bl+br+ur)) + (26*(l+r+t+b)) + (41*c))/273;
  //sigma = 1.6;
  half4 secondfilter = ((13*(luu + ldd + ruu + rdd)) + (23*(ult+urt+blt+brt+lu+ld+ru↩
      +rd)) + (28*(bb+uu+lm+rm)) + (42*(tl+bl+br+ur)) + (51*(l+r+t+b)) + (62*c))↩
      /782;

  //Find the difference between the gaussian blurs
  half first = ((0.3*firstfilter.x) + (0.59*firstfilter.y) + (0.11*firstfilter.z));
  half second = ((0.3*secondfilter.x) + (0.59*secondfilter.y) + (0.11*secondfilter.z↩
      ));
  half diffo;
  if(first>second){
    diffo = first - second;
  }
  else{
    diffo = second - first;
  }

    if(diffo<THRESHOLD){
      OUT=half4(0.0);
    }
    else{
      OUT.x=0.0;OUT.y=0.0;OUT.z=0.0;OUT.w=c.w;
    }


  return OUT;
}
```

Listing B.2: Sobel CG code.

```
float4 Sobel (half2 texCoord : TEX0,
      uniform sampler2D Image,
      uniform float threshold) : COLOR
{
  //Input level changes the threshold level of edges found, the greater the number ↩
      the more edges that are displayed (0-9.9)
    float INPUTLEVEL = threshold;

  float pixelSize = 0.0031;
    half4 OUT;
  half4 centre = tex2D(Image, texCoord);
  //Grab 3x3 matrix around current pixel
    half4 bl = tex2D(Image, texCoord + float2(-1, -1)*pixelSize);
    half4 l  = tex2D(Image, texCoord + float2(-1,0)*pixelSize);
    half4 tl = tex2D(Image, texCoord + float2(-1,1)*pixelSize);
    half4 t  = tex2D(Image, texCoord + float2(0,1)*pixelSize);
    half4 ur = tex2D(Image, texCoord + float2( 1,1)*pixelSize);
    half4 r  = tex2D(Image, texCoord + float2(1,0)*pixelSize);
    half4 br = tex2D(Image, texCoord + float2(1,-1)*pixelSize);
    half4 b  = tex2D(Image, texCoord + float2(0,-1)*pixelSize);
    half4 c  = tex2D(Image, texCoord);

    //get grayscale
  c.x = (c.x+c.y+c.z)/3;
  bl.x = (bl.x+bl.y+bl.z)/3;
  l.x = (l.x+l.y+l.z)/3;
  tl.x = (tl.x+tl.y+tl.z)/3;
  t.x = (t.x+t.y+t.z)/3;
  ur.x = (ur.x+ur.y+ur.z)/3;
  r.x = (r.x+r.y+r.z)/3;
  br.x = (br.x+br.y+br.z)/3;
  b.x = (b.x+b.y+b.z)/3;

  //Set threshold
  float internalthreshold;
  internalthreshold=(1.0-(INPUTLEVEL/10));

  //Sobel Operator
  float sumx = (tl.x*-1)+(l.x*-2)+(bl.x*-1)+(ur.x)+(r.x*2)+(br.x);
  float sumy = (tl.x*-1)+(t.x*-2)+(ur.x*-1)+(bl.x)+(b.x*2)+(br.x);

  OUT.x = sqrt((sumx*sumx)+(sumy*sumy));

  //Return black if above threshold, otherwise return input pixel
  if(OUT.x>internalthreshold){
    OUT.x = 0.0;OUT.y = 0.0;OUT.z = 0.0;
    OUT.w=centre.w;
  }
  else{
    OUT = half4(0.0);
  }

  return OUT;
}
```

Listing B.3: Canny CG code.

```
float4 Canny3_ps (half2 texCoord : TEX0,
      uniform sampler2D Image, uniform sampler2D Image2, uniform sampler2D Image3,
      uniform float threshold) : COLOR
{
  float CANNYTHRESHOLD = threshold;
  float pixelSize = 0.0031;
```

```
half4 c  = tex2D(Image2, texCoord);
half4 bl = tex2D(Image2, texCoord + float2(-1, -1)*pixelSize);
half4 l  = tex2D(Image2, texCoord + float2(-1,0)*pixelSize);
half4 tl = tex2D(Image2, texCoord + float2(-1,1)*pixelSize);
half4 t  = tex2D(Image2, texCoord + float2(0,1)*pixelSize);
half4 ur = tex2D(Image2, texCoord + float2( 1,1)*pixelSize);
half4 r  = tex2D(Image2, texCoord + float2(1,0)*pixelSize);
half4 br = tex2D(Image2, texCoord + float2(1,-1)*pixelSize);
half4 b  = tex2D(Image2, texCoord + float2(0,-1)*pixelSize);

c.x = (c.x+c.y+c.z)/3;
bl.x = (bl.x+bl.y+bl.z)/3;
l.x = (l.x+l.y+l.z)/3;
tl.x = (tl.x+tl.y+tl.z)/3;
t.x = (t.x+t.y+t.z)/3;
ur.x = (ur.x+ur.y+ur.z)/3;
r.x = (r.x+r.y+r.z)/3;
br.x = (br.x+br.y+br.z)/3;
b.x = (b.x+b.y+b.z)/3;

float sumx = (tl.x*-1)+(l.x*-2)+(bl.x*-1)+(ur.x)+(r.x*2)+(br.x);
float sumy = (tl.x*-1)+(t.x*-2)+(ur.x*-1)+(bl.x)+(b.x*2)+(br.x);

float orienting = atan2(sumy,sumx);

half LEVEL = sqrt((sumx*sumx)+(sumy*sumy));

half4 OUT;
c = tex2D(Image3, texCoord);
half4 c2  = tex2D(Image2, texCoord);
half4 c3  = tex2D(Image, texCoord);

bl  = tex2D(Image3, texCoord + float2(-1, -1)*pixelSize);
l   = tex2D(Image3, texCoord + float2(-1,0)*pixelSize);
tl  = tex2D(Image3, texCoord + float2(-1,1)*pixelSize);
ur  = tex2D(Image3, texCoord + float2( 1,1)*pixelSize);
t   = tex2D(Image3, texCoord + float2(0,1)*pixelSize);
r   = tex2D(Image3, texCoord + float2(1,0)*pixelSize);
br  = tex2D(Image3, texCoord + float2(1,-1)*pixelSize);
b   = tex2D(Image3, texCoord + float2(0,-1)*pixelSize);

float orient = orienting * 180 / 3.14159265;
OUT.x=1.0;OUT.y=1.0;OUT.z=1.0;
OUT.w=1.0;
OUT = c3;

if (orient > 180){
  orient-=180.0;
}

if((orient<=22.5)||(orient>157.5)){
  orient=180;
  if((c.x>l.x)&&(c.x>r.x)){
    OUT.x=0.0;OUT.y=0.0;OUT.z=0.0;
  }
```

```
    else{
      OUT = half4(1,0,0,0);
    }
  }
  else if((orient>22.5)&&(orient<=67.5)){
    orient=45;
    if((c.x>br.x)&&(c.x>tl.x)){
      OUT.x=0.0;OUT.y=0.0;OUT.z=0.0;
    }
    else{
      OUT = half4(1,0,0,0);
    }
  }
  else if((orient>67.5)&&(orient<=112.5)){
    orient=90;
    if((c.x>b.x)&&(c.x>t.x)){
      OUT.x=0.0;OUT.y=0.0;OUT.z=0.0;
    }
    else{
      OUT = half4(1,0,0,0);
    }
  }
  else if((orient>112.5)&&(orient<=157.5)){
    orient=135;
    if((c.x>ur.x)&&(c.x>bl.x)){
      OUT.x=0.0;OUT.y=0.0;OUT.z=0.0;
    }
    else{
      OUT = half4(1,0,0,0);
    }
  }
  if(LEVEL < CANNYTHRESHOLD){
    OUT = half4(1,0,0,0);
  }

  return OUT;
}
```

Listing B.4: Kuwahara CG code.

```
half4 variance (half3 surroundingPixels[25], half2 centre){

   int index;
   half3 average = half3(0,0,0);
   half3 minVal=1, maxVal=0;

   for(int x=1;x<=3;x++){
      for(int y=1;y<=3;y++){
         index = (y + centre.y)*5 + (x + centre.x);
         minVal = min(surroundingPixels[index], minVal);
         maxVal = max(surroundingPixels[index], maxVal);
         average = average + surroundingPixels[index];
      }
   }
   half3 var = maxVal-minVal;
   half averageVar = (var.r+var.g+var.b)/3.0;
   return half4(average/9.0, averageVar);
}


float4 Kuwa_ps (half2 texCoord : TEX0,
      uniform sampler2D Image,
      uniform float threshold) : COLOR
{

   float4 OUT;
   OUT = tex2D(Image, texCoord);

   half3 surroundingPixels[25];

   int index;
   half2 texIndex;
   for(int x=0;x<5;x++){
      for(int y=0;y<5;y++){
         index = y*5+x;
         texIndex = texCoord + half2((x-2)*threshold,(y-2)*threshold);
         surroundingPixels[index] = tex2D(Image, texIndex).rgb;
      }
   }

   //Loop through each area checking if alpha channel which stores variance is less
   half4 tempVar;
   half4 minVar;
   half2 centre[4] = {half2(1,1),half2(-1,1),half2(1,-1),half2(-1,-1)};

   minVar = variance(surroundingPixels, mask[0]);
   for(int i =1; i<4; i++){
      tempVar = variance(surroundingPixels, centre[i]);
      if (tempVar.a < minVar.a){
         minVar=tempVar;
      }
   }

   OUT.rgb=minVar.rgb;
  return OUT;
}
```

Listing B.5: Cel shading GLSL code.

```
vec3 celShading(in vec3 gradient, in vec3 vposTex, in VOLUME_PARAMETERS volumeParams↩
    , in vec3 ka, in vec3 kd, in vec3 ks) {
    // transform voxel position to the volume's object space
    vec3 vpos = (vposTex-0.5)*volumeParams.volumeCubeSize_;
    vec3 N = normalize(gradient);
    vec3 L = normalize(volumeParams.lightPositionOBJ_ - vpos);            // using↩
         light position in volume object space
    vec3 V = normalize(volumeParams.cameraPositionOBJ_ - vpos);  // using camera ↩
        position in volume object space

    vec3 ka1 = ka * lightSource_.ambientColor_;
    vec3 kd1 = kd * lightSource_.diffuseColor_;
    vec3 ks1 = ks * lightSource_.specularColor_;

    vec3 shadedColor = vec3(0.0);
    if (max(dot(N, V), 0.0) < 0.2){
        return vec3(0);
    }
    float ndotl = floor(max(dot(N, L), 0.0) * 2.0) / 2.0;
    float ndoth = floor(pow(max(dot(N, normalize(V + L)), 0.0), shininess_)*2.0) / ↩
        2.0;
    shadedColor =  ka1 + kd1 * ndotl + ks1*ndoth;


    return shadedColor;
}
```

Listing B.6: Ridge and valley GLSL code.

```
//compute min and max principle curvature
vec2 curvature(in sampler3D volume, in vec3 samplePos, in VOLUME_PARAMETERS ↩
    volumeParameters) {

    //compute hessian
    vec3 offset = volumeParameters_.datasetDimensionsRCP_;
    vec3 x = calcGradient(volume, volumeParameters, samplePos - vec3(offset.x, 0, 0)↩
        )
        - calcGradient(volume, volumeParameters, samplePos + vec3(offset.x, 0.0, ↩
            0.0))/2.0;
    vec3 y = calcGradient(volume, volumeParameters, samplePos - vec3(0, offset.y, 0)↩
        )
        - calcGradient(volume, volumeParameters, samplePos + vec3(0, offset.y, 0))↩
            /2.0;
    vec3 z = calcGradient(volume, volumeParameters, samplePos - vec3(0, 0, offset.z)↩
        )
        - calcGradient(volume, volumeParameters, samplePos + vec3(0, 0, offset.z))↩
            /2.0;
    mat3 H = mat3(  x.x, y.x, z.x, y.x, y.y, z.y, z.x, z.y, z.z   );

    //compute gradient tensor
    vec3 g = calcGradient(volume, volumeParameters, samplePos);
    vec3 n = normalize(-g);
    mat3 nnt = mat3(    n.x*n.x, n.x*n.y, n.x*n.z,
                        n.y*n.x, n.y*n.y, n.y*n.z,
                        n.z*n.x, n.z*n.y, n.z*n.z   );
    mat3 I = mat3(1.0);
    mat3 P = I - nnt;
    mat3 G = (-1.0/length(g))*P*H*P;

    //trace
    float T = G[0][0]+G[1][1]+G[2][2];

    //Frobenius norm
    mat3 temp = G * transpose(G);
    float F = sqrt(temp[0][0]+temp[1][1]+temp[2][2]);

    //compute curvatures
    float temp2 = sqrt(2*F*F-T*T);
    float k1 = (T+temp2)/2.0;
    float k2 = (T-temp2)/2.0;
    return vec2(k1,k2);
}

vec4 ridgeValley(in sampler3D volume, in vec3 samplePos, in VOLUME_PARAMETERS ↩
    volumeParameters){

    vec2 k = curvature(volume, samplePos, volumeParameters);

    const float SIZE = 0.3;
    if (k.x > (1.0 - SIZE) && (abs(k.y) < SIZE)){
        return vec4(vec3(1),1);
    }
    if (k.y < (-1.0 + SIZE) && (abs(k.x) < SIZE)){
        return vec4(vec3(0),1);
    }
    return vec4(0);
}
```

Listing B.7: Ambient occlusion calculation.

```
//get density values
vec3 offset = volumeParameters_.datasetDimensionsRCP_;
v0 = textureLookup3DUnnormalized(volume_, volumeParameters_, samplePos + vec3(offset↩
    .x, 0.0, 0.0)).a;
v1 = textureLookup3DUnnormalized(volume_, volumeParameters_, samplePos + vec3(0, ↩
    offset.y, 0)).a;
v2 = textureLookup3DUnnormalized(volume_, volumeParameters_, samplePos + vec3(0, 0, ↩
    offset.z)).a;
v3 = textureLookup3DUnnormalized(volume_, volumeParameters_, samplePos + vec3(-↩
    offset.x, 0, 0)).a;
v4 = textureLookup3DUnnormalized(volume_, volumeParameters_, samplePos + vec3(0, -↩
    offset.y, 0)).a;
v5 = textureLookup3DUnnormalized(volume_, volumeParameters_, samplePos + vec3(0, 0, ↩
    -offset.z)).a;

//get tf alpha values and do ambient occlusion
amboc = max(amboc,applyTF(transferFunc_, v0* volumeParameters_.bitDepthScale_).a);
amboc = max(amboc,applyTF(transferFunc_, v1* volumeParameters_.bitDepthScale_).a);
amboc = max(amboc,applyTF(transferFunc_, v2* volumeParameters_.bitDepthScale_).a);
amboc = max(amboc,applyTF(transferFunc_, v3* volumeParameters_.bitDepthScale_).a);
amboc = max(amboc,applyTF(transferFunc_, v4* volumeParameters_.bitDepthScale_).a);
amboc = max(amboc,applyTF(transferFunc_, v5* volumeParameters_.bitDepthScale_).a);
```

```
// apply classification
vec4 color = RC_APPLY_CLASSIFICATION(transferFunc_, voxel);

//if enabled and occluded apply amboc
if((amboc > color.a) && (useAmbientOcclusion_)){
    color.rgb *= AMBOC_WEIGHT1*amboc;
    color.a = min(1.0, AMBOC_WEIGHT2*amboc);
}
```

Bibliography

[Ada10] Robert Adams. Improving health outcomes with better patient understanding and education. *Risk Management and Healthcare Policy*, 3(1):61–72, October 2010.

[Ake93] Kurt Akeley. Reality engine graphics. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, page 109–116, New York, NY, USA, 1993. ACM.

[ARC05] Alfie Abdul-Rahman and Min Chen. Spectral volume rendering based on the kubelka-munk theory. *Computer Graphics Forum*, 24(3):413–422, 2005.

[AWC10] M. Ament, D. Weiskopf, and H. Carr. Direct interval volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1505 –1514, December 2010.

[Bau11] Dennis Bautembach. *Animated Sparse Voxel Octrees*. Bachelor's thesis, University of Hamburg, March 2011.

[BB07] David C. Banks and Kevin Beason. Fast global illumination for visualizing isosurfaces with a 3D illumination grid. *Computing in Science and Engg.*, 9(1):48–54, January 2007.

[BFGS86] Larry Bergman, Henry Fuchs, Eric Grant, and Susan Spach. Image rendering by adaptive refinement. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, page 29–37, New York, NY, USA, 1986. ACM.

[BG05] Stefan Bruckner and M. Eduard Gröller. VolumeShop: an interactive system for direct volume illustration. In *Visualization Conference, IEEE*, page 85, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[BG06] S. Bruckner and M.E. Gröller. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1077 –1084, October 2006.

[BG07a] S. Bruckner and M. E. Gröller. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 26(3):715–724, September 2007.

[BG07b] S. Bruckner and M.E. Gröller. Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1344 –1351, December 2007.

[BGB$^+$06] Kevin Beason, Josh Grant, David Banks, Brad Futch, and Yousuff Hussaini. Pre-computed illumination for isosurfaces. In *VDA '94: Proceedings of the conference on Visualization and Data Analysis '06*, volume SPIE Vol. 6060, pages 1–11, 2006.

[BGKG05] Stefan Bruckner, Sören Grimm, Armin Kanitsar, and M. Eduard Gröller. Illustrative context-preserving volume rendering. *Proceedings of EuroVis 2005*, pages 69—76, 2005.

[BGM$^+$10] Stefan Bruckner, Meister Eduard Gröller, Klaus Mueller, Bernhard Preim, and Deborah Silver. Illustrative Focus+Context approaches in interactive volume visualization. *Scientific Visualization: Advanced Concepts*, 2010.

[BKR$^+$05] Michael Burns, Janek Klawe, Szymon Rusinkiewicz, Adam Finkelstein, and Doug DeCarlo. Line drawings from volume data. *ACM Trans. Graph.*, 24(3):512–518, 2005.

[BKW08] K. Burger, J. Kruger, and R. Westermann. Direct volume editing. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1388 –1395, December 2008.

[Bli77] James F. Blinn. Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph.*, 11(2):192–198, July 1977.

[BLM96] M.J. Bentum, B.B.A. Lichtenbelt, and T. Malzbender. Frequency analysis of gradient estimators in volume rendering. *Visualization and Computer Graphics, IEEE Transactions on*, 2(3):242–254, 1996.

[BR98] Uwe Behrens and Ralf Ratering. Adding shadows to a texture-based volume renderer. In *Proceedings of the 1998 IEEE symposium on Volume visualization*, VVS '98, page 39–46, New York, NY, USA, 1998. ACM.

[Bru04] Stefan Bruckner. *Efficient Volume Visualization of Large Medical Datasets*. Masters thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, May 2004.

[BW98] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, page 43–54, New York, NY, USA, 1998. ACM.

[BWC04] Praveen Bhaniramka, Rephael Wenger, and Roger Crawfis. Isosurface construction in any dimension using convex hulls. *IEEE Transactions on Visualization and Computer Graphics*, 10(2):130–141, 2004.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[ÇÇ10] Ömer Cengiz Çelebi and Ulus Çevik. Accelerating volume rendering by ray leaping with back steps. *Comput. Methods Prog. Biomed.*, 97(2):99–113, February 2010. ACM ID: 1716120.

[CCB11] Weifeng Chen, Wei Chen, and Hujun Bao. An efficient direct volume rendering approach for dichromats. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2144 –2152, December 2011.

[CCF94] Brian Cabral, Nancy Cam, and Jim Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of the 1994 symposium on Volume visualization*, pages 91–98, Tysons Corner, Virginia, United States, 1994. ACM.

[CCF97] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. Making distortions comprehensible. In *Proceedings of the 1997 IEEE Symposium on Visual Languages (VL '97)*, VL '97, page 36–, Washington, DC, USA, 1997. IEEE Computer Society.

[CD05] Yy. Cai and F. Dong. Surface hatching for medical volume data. In *Computer Graphics, Imaging and Vision: New Trends, 2005. International Conference on*, pages 232–238, 2005.

[CDF+06] Forrester Cole, Doug DeCarlo, Adam Finkelstein, Kenrick Kin, Keith Morley, and Anthony Santella. Directing gaze in 3D models with stylized focus. *Eurographics Symposium on Rendering*, page 377–387, June 2006.

[CF07] Franck Caniard and Roland W. Fleming. 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.

[CF09] Forrester Cole and Adam Finkelstein. Fast high-quality line visibility. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 115–120, Boston, Massachusetts, 2009. ACM.

[CF10] Forrester Cole and Adam Finkelstein. Two fast methods for high-quality line visibility. *IEEE Transactions on Visualization and Computer Graphics*, 16(5):707–717, 2010.

[CGL+08] Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz. Where do people draw lines? In *ACM SIGGRAPH 2008 papers*, pages 1–11, Los Angeles, California, 2008. ACM.

[CKY01] Min Chen, Arie Kaufman, and Roni Yagel, editors. *Volume Graphics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[CM08] C. Correa and Kwan-Liu Ma. Size-based transfer functions: A new volume exploration technique. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1380 –1387, December 2008.

[CM09] C. Correa and Kwan-Liu Ma. The occlusion spectrum for volume classification and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1465–1472, November 2009.

[CMH+01] Balázs Csébfalvi, Lukas Mroz, Helwig Hauser, Andreas König, and Eduard Gröller. Fast visualization of object contours by non-photorealistic volume rendering. *Computer Graphics Forum*, 20:452–460, 2001.

[CN94] Timothy J. Cullip and Ulrich Neumann. Accelerating volume reconstruction with 3D texture hardware. Technical report, University of North Carolina at Chapel Hill, 1994.

[CNLE09] Cyril Crassin, Fabrice Neyret, Sylvain Lefebvre, and Elmar Eisemann. GigaVoxels: ray-guided streaming for efficient and detailed voxel rendering. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 15–22, Boston, Massachusetts, 2009. ACM.

[CNS+11] Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum (Proceedings of Pacific Graphics 2011)*, 30(7), September 2011.

[CR08] J.J. Caban and P. Rheingans. Texture-based transfer functions for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1364 –1371, December 2008.

[CRBR74] E Catmull, R Rom, R Barnhill, and R Riesenfeld. A class of local interpolating splines. *Computer aided geometric design. Academic Press.*, pages 317–326, 1974.

[CRD10] Andrew Corcoran, Niall Redmond, and John Dingliana. Perceptual enhancement of two-level volume rendering. *Computers & Graphics*, 34(4):388–397, August 2010.

[CRZP04] Wei Chen, Liu Ren, Matthias Zwicker, and Hanspeter Pfister. Hardware-accelerated adaptive EWA volume splatting. In *Proceedings of the conference on Visualization '04*, VIS '04, page 67–74. IEEE Computer Society, 2004. ACM ID: 1034432.

[CSC06] C.D. Correa, D. Silver, and M. Chen. Feature aligned volume manipulation for illustration and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1069 –1076, October 2006.

[CSD+09] Forrester Cole, Kevin Sanik, Doug DeCarlo, Adam Finkelstein, Thomas Funkhouser, Szymon Rusinkiewicz, and Manish Singh. How well do line drawings depict shape? In *ACM SIGGRAPH 2009 papers*, pages 1–9, New Orleans, Louisiana, 2009. ACM.

[CSW+03] M. Chen, D. Silver, A. S. Winter, V. Singh, and N. Cornea. Spatial transfer functions: a unified approach to specifying deformation in volume modeling and animation. In *Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics*, VG '03, page 35–44, New York, NY, USA, 2003. ACM.

[CWM+09] Ming-Yuen Chan, Yingcai Wu, Wai-Ho Mak, Wei Chen, and Huamin Qu. Perception-based transparency optimization for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1283 –1290, December 2009.

[DaV78] Leonardo DaVinci. Dell'anatomia fogli a et b. quaderni d'anatomia. i-VI, 1478.

[DC05] Feng Dong and Gordon J. Clapworthy. Volumetric texture synthesis for non-photorealistic volume rendering of medical data. *The Visual Computer*, 21(7):463–473, 2005.

[DCLK03] Feng Dong, G.J. Clapworthy, Hai Lin, and M.A. Krokos. Nonphotorealistic rendering of medical volume data. *Computer Graphics and Applications, IEEE*, 23(4):44–52, 2003.

[DE07] Philippe Desgranges and Klaus Engel. Fast ambient occlusion for direct volume rendering, January 2007. US patent application 2007/0013696 A1.

[DEP05] Philippe Desgranges, Klaus Engel, and Gianluca Paladini. Gradient-free shading: A new method for realistic interactive volume rendering. In *Vision, Modelling and Visualization*, pages 209–216, Erlangen, Germany, 2005.

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

[DR07] Doug DeCarlo and Szymon Rusinkiewicz. Highlight lines for conveying shape. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 63–70, San Diego, California, 2007. ACM.

[DS02] Doug DeCarlo and Anthony Santella. Stylization and abstraction of photographs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, page 769–776, New York, NY, USA, 2002. ACM.

[DSS+09] C. A Dietrich, C. E Scheidegger, J. Schreiner, J. L.D Comba, L. P Nedel, and C. T Silva. Edge transformations for improving mesh quality of marching cubes. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):150–159, February 2009.

[DVND10] José Díaz, Pere-Pau Vázquez, Isabel Navazo, and Florent Duguet. Real-time ambient occlusion and halos with summed area tables. *Computers & Graphics*, 34(4):337–350, August 2010.

[EKE01] Klaus Engel, Martin Kraus, and Thomas Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 9–16, Los Angeles, California, United States, 2001. ACM.

[ER00] David Ebert and Penny Rheingans. Volume illustration: non-photorealistic rendering of volume models. In *Proceedings of the conference on Visualization '00*, pages 195–202, Salt Lake City, Utah, United States, 2000. IEEE Computer Society Press.

[FM07] N. Fout and Kwan-Liu Ma. Transform coding for hardware-accelerated volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1600 –1607, December 2007.

[FS93] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, page 247–254, New York, NY, USA, 1993. ACM.

[Fur86] G. W. Furnas. Generalized fisheye views. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '86, page 16–23, New York, NY, USA, 1986. ACM.

[GAMD10] A. Guetat, A. Ancel, S. Marchesin, and J.-M. Dischler. Pre-integrated volume rendering with non-linear gradient interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1487 –1494, December 2010.

[Gar90] Michael P. Garrity. Raytracing irregular volume data. In *Proceedings of the 1990 workshop on Volume visualization*, VVS '90, page 35–40, New York, NY, USA, 1990. ACM.

[GB99] Enrico Gobbetti and Eric Bouvier. Time-critical multiresolution scene rendering. In *Proceedings of the conference on Visualization ’99: celebrating ten years*, VIS ’99, page 123–130, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[GGSC98] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A non-photorealistic lighting model for automatic technical illustration. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 447–452. ACM, 1998.

[GK96] Allen Van Gelder and Kwansik Kim. Direct volume rendering with shading via three-dimensional textures. In *Proceedings of the 1996 symposium on Volume visualization*, pages 23–ff., San Francisco, California, United States, 1996. IEEE Press.

[GMY11] Hanqi Guo, Ningyu Mao, and Xiaoru Yuan. WYSIWYG (What you see is what you get) volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2106 –2114, December 2011.

[GNBP11] R. Gasteiger, M. Neugebauer, O. Beuing, and B. Preim. The FLOWLENS: a focus-and-context visualization approach for exploration of blood flow in cerebral aneurysms. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2183 –2192, December 2011.

[GTDS10] Stéphane Grabli, Emmanuel Turquin, Frédo Durand, and François X. Sillion. Programmable rendering of line drawing from 3D scenes. *ACM Trans. Graph.*, 29(2):18:1–18:20, April 2010.

[GWGS02] S. Guthe, M. Wand, J. Gonser, and W. Strasser. Interactive rendering of large volume data sets. In *IEEE Visualization, 2002. VIS 2002*, pages 53–60. IEEE, November 2002.

[GXY12] Hanqi Guo, He Xiao, and Xiaoru Yuan. Scalable multivariate volume visualization and analysis based on dimension projection and parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 18(9):1397 –1410, September 2012.

[HBC12] M. Howison, E.W. Bethel, and H. Childs. Hybrid parallelism for volume rendering on large-, multi-, and many-core systems. *IEEE Transactions on Visualization and Computer Graphics*, 18(1):17 –29, January 2012.

[HBH03] M. Hadwiger, C. Berger, and H. Hauser. High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *Visualization, 2003. VIS 2003. IEEE*, pages 301–308, 2003.

[HKG00] Jiří Hlad?vka, Andreas König, and Eduard Gröller. Curvature-based transfer functions for direct volume rendering. In *Spring Conference on Computer Graphics 2000*, page 58–65, 2000.

[HKRs+06] Markus Hadwiger, Joe M Kniss, Christof Rezk-salama, Daniel Weiskopf, and Klaus Engel. Stochastic jittering. In *Real-time Volume Graphics*, pages 220–223. A. K. Peters, Ltd., Natick, MA, USA, 2006.

[HKSB06] Markus Hadwiger, Andrea Kratz, Christian Sigg, and Katja Bühler. GPU-accelerated deep shadow maps for direct volume rendering. In *Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*, page 49–52, New York, NY, USA, 2006. ACM.

[HLY07] Frida Hernell, Patric Ljung, and Anders Ynnerman. Efficient ambient and emissive tissue illumination using local occlusion in multiresolution volume rendering. In *IEEE/EG Volume Graphics*, 2007.

[HLY10] F. Hernell, P. Ljung, and A. Ynnerman. Local ambient occlusion in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):548–559, August 2010.

[HMBG00] H. Hauser, L. Mroz, G.-I. Bischi, and M. E Groller. Two-level volume rendering - fusing MIP and DVR. In *Visualization 2000. Proceedings*, pages 211–218. IEEE, October 2000.

[HMBG01] H. Hauser, L. Mroz, G.-I. Bischi, and M.E. Groller. Two-level volume rendering. *Visualization and Computer Graphics, IEEE Transactions on*, 7(3):242 –252, September 2001.

[HMH+03] Nick Halper, Mara Mellin, Christoph S. Herrmann, Volker Linneweber, and Thomas Strothotte. Psychology and non-photorealistic rendering: The beginning of a beautiful relationship. In *Mensch & Computer*, 2003.

[HSS+05] Markus Hadwiger, Christian Sigg, Henning Scharsach, Khatja Bühler, and Markus Gross. Real-time ray-casting and advanced shading of discrete isosurfaces. *Computer Graphics Forum*, 24(3):303–312, 2005.

[Hub96] Philip M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Trans. Graph.*, 15:179–210, July 1996.

[HWSB99] Geoffrey S. Hubona, Philip N. Wheeler, Gregory W. Shirah, and Matthew Brandt. The relative contributions of stereo, lighting, and background scenes in promoting 3D depth visualization. *ACM Transactions on Computer-Human Interaction*, 6(3):214–242, September 1999.

[IFP95] Victoria Interrante, Henry Fuchs, and Stephen Pizer. Enhancing transparent skin surfaces with ridge and valley lines. In *Proceedings of the 6th conference on Visualization '95*, VIS '95, page 52–, Washington, DC, USA, 1995. IEEE Computer Society.

[IFP96] Victoria Interrante, Henry Fuchs, and Stephen Pizer. Illustrating transparent surfaces with curvature-directed strokes. In *Proceedings of the 7th conference on Visualization '96*, VIS '96, page 211–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

[IH04] Milan Ikits and Charles D. Hansen. A focus and context interface for interactive volume rendering, 2004. http://www.cs.utah.edu/~ikits/.

[IKN98] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[Int97] Victoria Interrante. Illustrating surface shape in volume data via principal direction-driven 3D line integral convolution. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, page 109–116, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[ISC07] S. Islam, D. Silver, and Min Chen. Volume splitting and its applications. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):193 –203, April 2007.

[JBB+08] W. M. Jainek, S. Born, D. Bartz, W. Straßer, and J. Fischer. Illustrative hybrid visualization and exploration of anatomical and functional brain data. *Computer Graphics Forum*, 27(3):855–862, September 2008.

[JDA07] Tilke Judd, Frédo Durand, and Edward Adelson. Apparent ridges for line drawing. *ACM Trans. Graph.*, 26(3):19, 2007.

[JH09] C.R. Johnson and Jian Huang. Distribution-driven visualization of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):734 –746, October 2009.

[JLA96] Ammar Joukhadar, Christian Laugier, and Inria Rhone Alpes. Adaptive time step for fast converging dynamic simulation system. In *In Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, page 418–424, 1996.

[KD98] Gordon Kindlmann and James W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *Proceedings of the 1998 IEEE symposium on Volume visualization*, VVS '98, page 79–86, New York, NY, USA, 1998. ACM.

[KDMF03] Robert D. Kalnins, Philip L. Davidson, Lee Markosian, and Adam Finkelstein. Coherent stylized silhouettes. *ACM Trans. Graph.*, 22(3):856–861, 2003.

[KHEK76] M. Kuwahara, K. Hachimura, S. Eiho, and M. Kinoshita. 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.

[KHW+09] A. Knoll, Y. Hijazi, R. Westerteiger, M. Schott, C. Hansen, and H. Hagen. Volume ray casting with peak finding and differential sampling. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1571 –1578, December 2009.

[KJL+11] Joel Kronander, Daniel Jönsson, Joakim Löw, Patric Ljung, Anders Ynnerman, and Jonas Unger. Efficient visibility encoding for dynamic illumination in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 99(1), 2011.

[KKG09] Thomas Kalbe, Thomas Koch, and Michael Goesele. High-quality rendering of varying isosurfaces with cubic trivariate c1-continuous splines. In *Proceedings of the 5th International Symposium on Advances in Visual Computing: Part I*, pages 596–607, Las Vegas, Nevada, 2009. Springer-Verlag.

[KKH01] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proceedings of the conference on Visualization '01*, VIS '01, page 255–262, Washington, DC, USA, 2001. IEEE Computer Society.

[KKH02] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, July 2002.

[KKMB96] D. Kersten, D. C. Knill, P. Mamassian, and I. Bülthoff. Illusory motion from shadows. *Nature*, 379(6560):31, January 1996. PMID: 8538738.

[KN01] Tae-Yong Kim and Ulrich Neumann. Opacity shadow maps. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, page 177–182, London, UK, 2001. Springer-Verlag.

[KNBH12] Evangelos Kalogerakis, Derek Nowrouzezahrai, Simon Breslav, and Aaron Hertzmann. Learning hatching for pen-and-ink illustration of surfaces. *ACM Trans. Graph.*, 31(1):1:1–1:17, February 2012.

[KPB12] Thomas Kroes, Frits H. Post, and Charl P. Botha. Exposure render: An interactive photo-realistic volume rendering framework. *PLoS ONE*, 7(7), 2012.

[KPH+03] J. Kniss, S. Premoze, C. Hansen, P. Shirley, and A. McPherson. A model for volume lighting and modeling. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):150–162, April 2003.

[KPHE02] Joe Kniss, Simon Premoze, Charles Hansen, and David Ebert. Interactive translucent volume rendering and procedural modeling. In *Proceedings of the conference on Visualization '02*, VIS '02, page 109–116, Washington, DC, USA, 2002. IEEE Computer Society.

[KSSE05] Thomas Klein, Magnus Strengert, Simon Stegmaier, and Thomas Ertl. Exploiting frame-to-frame coherence for accelerating high-quality volume raycasting on graphics hardware. *Proceedings of IEEE Visualization '05*, pages 223—230, 2005.

[KST08] Michael Kolomenkin, Ilan Shimshoni, and Ayellet Tal. Demarcating curves for shape illustration. *ACM Trans. Graph.*, 27(5):157:1–157:9, December 2008.

[KSW06] J. Krüger, J. Schneider, and R. Westermann. ClearView: an interactive context preserving hotspot visualization technique. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):941–948, 2006.

[KV06] Y. Kim and A. Varshney. Saliency-guided enhancement for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):925–932, October 2006.

[KvDK92] J. J. Koenderink, A. J. van Doorn, and A. M. Kappers. Surface perception in pictures. *Perception & psychophysics*, 52(5):487–496, November 1992. PMID: 1437481.

[KW03] J. Krüger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 38. IEEE Computer Society, 2003.

[KWTM03] Gordon Kindlmann, Ross Whitaker, Tolga Tasdizen, and Torsten Moller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 67. IEEE Computer Society, 2003.

[KYYL08] Yongjin Kim, Jingyi Yu, Xuan Yu, and Seungyong Lee. Line-art illustration of dynamic and specular surfaces. *ACM Trans. Graph.*, 27(5):156:1–156:10, December 2008.

[LA94] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Trans. Comput.-Hum. Interact.*, 1(2):126–160, June 1994.

[LB00] Michael S. Langer and Heinrich H. Bülthoff. Depth discrimination from shading under diffuse lighting. *Perception*, 29(6):649 – 660, 2000.

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

[LE05] A. Lu and D.S. Ebert. Example-based volume illustrations. In *Visualization, 2005. VIS 05. IEEE*, pages 655 – 662, October 2005.

[Lev88] M. Levoy. Display of surfaces from volume data. *Computer Graphics and Applications, IEEE*, 8(3):29–37, 1988.

[Lev90] Marc Levoy. Efficient ray tracing of volume data. *ACM Trans. Graph.*, 9(3):245–261, 1990.

[LHJ01] Eric LaMar, Bernd Hamann, and Kenneth I. Joy. A magnification lens for interactive volume visualization. In *Proceedings of the 9th Pacific Conference on Computer Graphics and Applications*, PG '01, page 223–, Washington, DC, USA, 2001. IEEE Computer Society.

[LHV04] C. H. Lee, Xuejun Hao, and A. Varshney. Light collages: lighting design for effective visualization. In *IEEE Visualization, 2004*, pages 281– 288. IEEE, October 2004.

[Lin98] Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30:79–116, 1998.

[LL94] Philippe Lacroute and Marc Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 451–458. ACM, 1994.

[LLC04] Shih-Kuan Liao, Jim Z. C. La, and Yeh-Ching Chung. Time-critical rendering for time-varying volume data. *Computers & Graphics*, 28(2):279 – 288, 2004.

[LLVT03] Thomas Lewiner, Hélio Lopes, Antônio Wílson Vieira, and Geovan Tavares. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, 8(2):383–366, December 2003.

[LLYM04] Patric Ljung, Claes Lundstrom, Anders Ynnerman, and Ken Museth. Transfer function based adaptive decompression for volume rendering of large medical data sets. In *VV '04: Proceedings of the 2004 IEEE Symposium on Volume Visualization and Graphics*, pages 25–32, Washington, DC, USA, 2004. IEEE Computer Society.

[LM02] Eric B. Lum and Kwan-Liu Ma. Hardware-accelerated parallel non-photorealistic volume rendering. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 67–ff, Annecy, France, 2002. ACM.

[LM04] Eric B. Lum and Kwan-Liu Ma. Lighting transfer functions using gradient aligned sampling. In *Proceedings of the conference on Visualization '04*, VIS '04, page 289–296, Washington, DC, USA, 2004. IEEE Computer Society.

[LMC02] E.B. Lum, Kwan-Liu Ma, and J. Clyne. A hardware-assisted scalable solution for interactive volume rendering of time-varying data. *Visualization and Computer Graphics, IEEE Transactions on*, 8(3):286–301, 2002.

[LME⁺02] Aidong Lu, Christopher J. Morris, David S. Ebert, Penny Rheingans, and Charles Hansen. Non-photorealistic volume rendering using stippling techniques. In *Proceedings of the conference on Visualization '02*, pages 211–218, Boston, Massachusetts, 2002. IEEE Computer Society.

[LMHB00] Adam Lake, Carl Marshall, Mark Harris, and Marc Blackstein. 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.

[LMK03] Wei Li, Klaus Mueller, and Arie Kaufman. Empty space skipping and occlusion clipping for texture-based volume rendering. In *In Proc. IEEE Visualization 2003*, pages 317—324, 2003.

[LMLH07] Yunjin Lee, Lee Markosian, Seungyong Lee, and John F. Hughes. Line drawings via abstracted shading. *ACM Trans. Graph.*, 26(3):18, 2007.

[LMT⁺03] Aidong Lu, C.J. Morris, J. Taylor, D.S. Ebert, C. Hansen, P. Rheingans, and M. Hartner. Illustrative interactive stipple rendering. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):127 – 138, June 2003.

[LR10] Florian Lindemann and Timo Ropinski. Advanced light material interaction for direct volume rendering. In Rüdiger Westermann and Gordon Kindlmann, editors, *IEEE/EG International Symposium on Volume Graphics*, page 101–108. Eurographics Association, 2010.

[LR11] Florian Lindemann and Timo Ropinski. About the influence of illumination models on image comprehension in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1922–1931, December 2011.

[LS01] Xinyue Li and Han-wei Shen. Adaptive volume rendering using fuzzy logic. *IN PROCEEDINGS OF JOINT EUROGRAPHICS-IEEE TCVG SYMPOSIUM ON VISUALIZATION*, 2001.

[LS02] Xinyue Li and Han-Wei Shen. Time-critical multiresolution volume rendering using 3D texture mapping hardware. In *Volume Visualization and*

Graphics, 2002. Proceedings. IEEE / ACM SIGGRAPH Symposium on, pages 29–36, 2002.

[LS07] François Labelle and Jonathan Richard Shewchuk. Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. *ACM Trans. Graph.*, 26(3), July 2007.

[LSM03] Eric B. Lum, Aleksander Stompel, and Kwan-Liu Ma. Using motion to illustrate static 3D shape- kinetic visualization. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):115–126, 2003.

[LV00] Tom Lokovic and Eric Veach. Deep shadow maps. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*, pages 385–392, Not Known, 2000.

[LW90] Marc Levoy and Ross Whitaker. Gaze-directed volume rendering. *SIGGRAPH Comput. Graph.*, 24(2):217–223, February 1990.

[LWM04] Eric Lum, Brett Wilson, and Kwan-Liu Ma. High-quality lighting and efficient pre-integration for volume rendering. In *Proceedings of Joint Eurographics-IEEE TVCG Symposium on Visualization*, pages 25–34, May 2004.

[LYS+10] Byeonghun Lee, Jihye Yun, Jinwook Seo, Byonghyo Shim, Yeong-Gil Shin, and Bohyoung Kim. Fast high-quality volume ray casting with virtual samplings. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1525 –1532, December 2010.

[Max95] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, June 1995.

[MBG08] Ann McNamara, Reynold Bailey, and Cindy Grimm. Improving search task performance using subtle gaze direction. In *Proceedings of the 5th symposium on Applied perception in graphics and visualization*, APGV '08, page 51–56, New York, NY, USA, 2008. ACM.

[MC10] Nelson Max and Min Chen. Local and global illumination in the volume rendering integral. In Hans Hagen, editor, *Scientific Visualization: Advanced Concepts*, volume 1 of *Dagstuhl Follow-Ups*, page 259–274. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2010.

[MDM10] S. Marchesin, J.-M. Dischler, and C. Mongenet. Per-pixel opacity modulation for feature enhancement in volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):560 –570, August 2010.

[MFS06] Gerd Marmitt, Heiko Friedrich, and Philipp Slusallek. Interactive volume rendering with ray tracing. In *Eurographics State of the Art Reports*, 2006.

[MH80] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, February 1980.

[MHDH07] P. Muigg, M. Hadwiger, H. Doleisch, and H. Hauser. Scalable hybrid unstructured and structured grid raycasting. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1592 –1599, December 2007.

[MSRMH09] Jennis Meyer-Spradow, Timo Ropinski, Jörg Mensmann, and Klaus Hinrichs. Voreen: A rapid-prototyping environment for ray-casting-based volume visualizations. *IEEE Computer Graphics and Applications*, 29(6):6–13, 2009.

[MTB03] Michael J. McGuffin, Liviu Tancau, and Ravin Balakrishnan. Using deformations for browsing volumetric data. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, page 53–, Washington, DC, USA, 2003. IEEE Computer Society.

[MW97] Heinrich Müller and Michael Wehle. Visualization of implicit surfaces using adaptive tetrahedrizations. In *Scientific Visualization Conference*, page 243, Los Alamitos, CA, USA, 1997. IEEE Computer Society.

[NCKG00] László Neumann, Balázs Csébfalvi, Andreas König, and Eduard Gröller. Gradient estimation in volume data using 4D linear regression. Technical Report TR-186-2-00-03, Institute of Computer Graphics and Algorithms, Vienna University of Technology, February 2000.

[Net97] Frank H. Netter. *Atlas of Human Anatomy*. Rittenhouse Book Distributors Inc., 1997.

[NJA$^+$01] S. G. Nikolov, M. G. Jones, D. Agrafiotis, D. R. Bull, and C. N. Canagarajah. Focus+Context visualisation for fusion of volumetric medical images. In *Proceedings of the 4th International Conference on Information Fusion*, Montreal, QC, Canada, August 2001.

[NSW02] Zoltan Nagy, Jens Schneider, and Rüdiger Westermann. Interactive volume illustration. In *VMV'02*, pages 497–504, 2002.

[OBA08] James P. O'Shea, Martin S. Banks, and Maneesh Agrawala. The assumed light direction for perceiving shape from shading. In *Proceedings of the 5th symposium on Applied perception in graphics and visualization*, APGV '08, page 135–142, New York, NY, USA, 2008. ACM.

[OBS04] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Ridge-valley lines on meshes via implicit surface fitting. In *ACM SIGGRAPH 2004 Papers*, pages 609–612, Los Angeles, California, 2004. ACM.

[OKL$^+$10] Kevin J. O'Leary, Nita Kulkarni, Matthew P. Landler, Jiyeon Jeon, Katherine J. Hahn, Katherine M. Englert, and Mark V. Williams. Hospitalized patients' understanding of their plan of care. *Mayo Clinic Proceedings*, 85(1):47–52, January 2010. PMID: 20042561 PMCID: PMC2800283.

[PB07] Bernhard Preim and Dirk Bartz. *Visualization in Medicine: Theory, Algorithms, and Applications (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[Pho75] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, June 1975.

[PM08] Eric Penner and Ross Mitchell. Isosurface ambient occlusion and soft shadows with filterable occlusion maps. In Hans-Christian Hege, David H. Laidlaw, Renato Pajarola, and Oliver G. Staadt, editors, *Volume Graphics*, pages 57–64. Eurographics Association, 2008.

[PSL$^+$98] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan. Interactive ray tracing for isosurface rendering. In *Visualization '98. Proceedings*, pages 233–238, 1998.

[PTD05] Bernhard Preim, Christian Tietjen, and Christina Dörge. NPR, focussing and emphasis in medical visualizations. In *Simulation und Visualisierung*, pages 139–152, 2005.

[Pue89] Antonio Medina Puerta. The power of shadows: shadow stereopsis. *Journal of the Optical Society of America A*, 6(2):309–311, February 1989.

[RBD06] Szymon Rusinkiewicz, Michael Burns, and Doug DeCarlo. Exaggerated shading for depicting shape and detail. *ACM Trans. Graph.*, 25(3):1199–1205, 2006.

[RBEG07] P. Rautek, S. Bruckner, and M. Eduard Gröller. Semantic layers for illustrative volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1336 –1343, December 2007.

[RBFS10] Marc Ruiz, Imma Boada, Miquel Feixas, and Mateu Sbert. Interactive volume illustration using intensity filtering. In *Computational Aesthetics*, pages 51–58. Eurographics Association, 2010.

[RBV+08] M. Ruiz, I. Boada, I. Viola, S. Bruckner, M. Feixas, and M. Sbert. Obscurance-based volume rendering framework. In *IEEE/ EG Symposium on Volume and Point-Based Graphics*, 2008.

[RC06] David Rodgman and Min Chen. Refraction in volume graphics. *Graphical Models*, 68(5–6):432–450, September 2006.

[RD08] Niall Redmond and John Dingliana. A hybrid technique for creating meaningful abstractions of dynamic scenes in real-time. In *Proceedings of WSCG'08: The International Conference in Eastern Europe on Computer Graphics, Vision and Visualisation*, Plzen, Czech Republic, February 2008.

[RD09a] Niall Redmond and John Dingliana. Influencing user attention using real-time stylised rendering. In *TPCG 2009: Proceedings of Theory and Practice of Computer Graphics Conference*, Cardiff, UK, 2009. Eurographics UK Chapter.

[RD09b] Niall Redmond and John Dingliana. Investigating the effect of real-time stylisation techniques on user task performance. In *APGV '09: Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization*, page 121–124, New York, NY, USA, 2009. ACM.

[RDRS10] T. Ropinski, C. Doring, and C. Rezk-Salama. Interactive volumetric lighting simulating scattering and shadowing. In *2010 IEEE Pacific Visualization Symposium (PacificVis)*, pages 169–176. IEEE, March 2010.

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

[RGW$^+$03] Stefan Roettger, Stefan Guthe, Daniel Weiskopf, Thomas Ertl, and Wolfgang Strasser. Smart hardware-accelerated volume rendering. In *Proceedings of the symposium on Data visualisation 2003*, VISSYM '03, page 231–238, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[Rit07] Tobias Ritschel. Fast GPU-based visibility computation for natural illumination of volume data sets. In P Cignoni and J Sochor, editors, *Short Paper Proceedings of Eurographics 2007*, page 17–20, September 2007.

[RKE00] S. Rottger, M. Kraus, and T. Ertl. Hardware-accelerated volume and isosurface rendering based on cell-projection. In *Visualization 2000. Proceedings*, pages 109–116, 2000.

[RKH08] Timo Ropinski, Jens Kasten, and Klaus H. Hinrichs. Efficient shadows for GPU-based volume raycasting. In *Proceedings of the 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2008)*, page 17–24, 2008.

[RMD$^+$08] Timo Ropinski, Jennis Meyer-Spradow, Stefan Diepenbrock, Jörg Mensmann, and Klaus Hinrichs. Interactive volume rendering with dynamic ambient occlusion and color bleeding. *Computer Graphics Forum*, 27(2):567–576, April 2008.

[RSH06] Timo Ropinski, Frank Steinicke, and Klaus H. Hinrichs. Visual exploration of seismic volume datasets. *Journal Proceedings of the 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG06)*, 14:73–80, 2006.

[RSKU$^+$10] Marc Ruiz, Lázló Szirmay-Kalos, Tamás Umenhoffer, Imma Boada, Miquel Feixas, and Mateu Sbert. Volumetric ambient occlusion for volumetric models. *The Visual Computer*, 26(6-8):687–695, April 2010.

[RW08] Sundaresan Raman and Rephael Wenger. Quality isosurface mesh generation using an extended marching cubes lookup table. *Computer Graphics Forum*, 27:791–798, 2008.

[Sal07] Christof Rezk Salama. GPU-Based monte-carlo volume raycasting. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, PG '07, page 411–414, Washington, DC, USA, 2007. IEEE Computer Society.

[SBS05] Zein Salah, Dirk Bartz, and Wolfgang Straßer. Illustrative rendering of segmented anatomical data. In *In Proc. of Simulation und Visualisierung*, page 175–184, 2005.

[SD04] Anthony Santella and Doug DeCarlo. Visual interest and NPR: an evaluation and manifesto. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, NPAR '04, page 71–150, New York, NY, USA, 2004. ACM.

[SDGPG07] Kartic Subr, Pablo Diaz-Gutierrez, Renato Pajarola, and M. Gopi. Order independent, attenuation-leakage free splatting using FreeVoxels, 2007.

[SEA09] N.A. Svakhine, D.S. Ebert, and W.M. Andrews. Illustration-inspired depth enhanced volumetric medical visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 15(1):77–86, 2009.

[Ser83] Jean Serra. *Image Analysis and Mathematical Morphology*. Academic Press, Inc., 1983.

[Sha49] Claude E. Shannon. Communication in the presence of noise. *Proc. Institute of Radio Engineers*, 37(1):10–21, January 1949.

[SIGM+11] S.K. Suter, J.A. Iglesias Guitian, F. Marton, M. Agus, A. Elsener, C.P.E. Zollikofer, M. Gopi, E. Gobbetti, and R. Pajarola. Interactive multiscale tensor reconstruction for multiresolution volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2135 –2143, December 2011.

[SKS02] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics*, 21, July 2002.

[SLM02] Aleksander Stompel, Eric B. Lum, and Kwan-Liu Ma. Visualization of multidimensional, multivariate volume data using hardware-accelerated non-photorealistic rendering techniques. In *Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, page 394. IEEE Computer Society, 2002.

[SMP11] P. Schlegel, M. Makhinya, and R. Pajarola. Extinction-based shading and illumination in GPU volume ray-casting. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1795 –1802, December 2011.

[SP98] Jennifer Sun and Pietro Perona. Where is the sun? *Nature Neuroscience*, 1(3):183–184, July 1998.

[ŠPBV10] Veronika Šoltészová, Daniel Patel, Stefan Bruckner, and Ivan Viola. A multidirectional occlusion shading model for direct volume rendering. *Computer Graphics Forum*, 29(3):883–891, June 2010.

[SPH⁺09] Mathias Schott, Vincent Pegoraro, Charles Hansen, Kévin Boulanger, and Kadi Bouatouch. A directional occlusion shading model for interactive direct volume rendering. *Computer Graphics Forum*, 28(3):855–862, June 2009.

[ŠPV11] Veronika Šoltészová, Daniel Patel, and Ivan Viola. Chromatic shadows for improved perception. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*, NPAR '11, page 105–116, New York, NY, USA, 2011. ACM.

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

[SSS06] J. Schreiner, C. Scheidegger, and C. Silva. High-quality extraction of isosurfaces from regular and irregular grids. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):1205–1212, 2006.

[ST90] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-d shapes. *SIGGRAPH Comput. Graph.*, 24(4):197–206, September 1990.

[Ste03] A. James Stewart. Vicinity shading for enhanced perception of volumetric data. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, page 47–, Washington, DC, USA, 2003. IEEE Computer Society.

[SYR11] Erik Sundén, Anders Ynnerman, and Timo Ropinski. Image plane sweep volume illumination. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2125–2134, 2011.

[TC00] S. M. F. Treavett and M. Chen. Pen-and-ink rendering in volume visualisation. In *Proceedings of the conference on Visualization '00*, pages 203–210, Salt Lake City, Utah, United States, 2000. IEEE Computer Society Press.

[TIP05] Christian Tietjen, Tobias Isenberg, and Bernhard Preim. Combining silhouettes, surface, and volume rendering for surgery education and planning. In Ken Brodlie, David J. Duke, and Kenneth I. Joy, editors, *EuroVis*, pages 303–310. Eurographics Association, 2005.

[VGKG04] Ivan Viola, Armin Kanitsar Meister Eduard Gröller, Armin Kanitsar, and Meister Eduard Gröller. Importance-driven volume rendering. *IN PROCEEDINGS OF IEEE VISUALIZATION*, pages 139—145, 2004.

[VKG05] I. Viola, A. Kanitsar, and M.E. Groller. Importance-driven feature enhancement in volume visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 11(4):408–418, 2005.

[vPVvdW10] R. van Pelt, A. Vilanova, and H. van de Wetering. Illustrative volume visualization using GPU-Based particle systems. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):571 –582, August 2010.

[Wes89] Lee Westover. Interactive volume rendering. In *Proceedings of the 1989 Chapel Hill workshop on Volume visualization*, pages 9–16, Chapel Hill, Noth Carolina, United States, 1989. ACM.

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

[WGS07] C. Wang, A. Garcia, and H.-W. Shen. Interactive level-of-detail selection using image-based quality metric for large volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):122 –134, February 2007.

[WKE99] R. Westermann, L. Kobbelt, and T. Ertl. Real-time exploration of regular volume data by adaptive reconstruction of iso-surfaces. *The Visual Computer*, 15:100–111, 1999.

[WM08] Chaoli Wang and Kwan-Liu Ma. A statistical approach to volume data quality assessment. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):590 –602, June 2008.

[WPSH06] Chris Wyman, Steven Parker, Peter Shirley, and Charles Hansen. Interactive display of isosurfaces with global illumination. *IEEE Transactions on Visualization and Computer Graphics*, 12(2):186–196, March 2006.

[WWE04] Daniel Weiskopf, Manfred Weiler, and Thomas Ertl. Maintaining constant frame rates in 3D texture-based volume rendering. In *Proceedings of the Computer Graphics International*, CGI '04, page 604–607, Washington, DC, USA, 2004. IEEE Computer Society.

[WWH$^+$00] Manfred Weiler, Rüdiger Westermann, Chuck Hansen, Kurt Zimmermann, and Thomas Ertl. Level-of-detail volume rendering via 3D textures. In *Proceedings of the 2000 IEEE symposium on Volume visualization*, VVS '00, page 7–13, New York, NY, USA, 2000. ACM.

[WWLM11] Yu-Shuen Wang, Chaoli Wang, Tong-Yee Lee, and Kwan-Liu Ma. Feature-preserving volume data reduction and Focus+Context visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(2):171 –181, February 2011.

[WZMK05] L. Wang, Y. Zhao, K. Mueller, and A. Kaufman. The magic volume lens: an interactive focus+context technique for volume rendering. In *Visualization, 2005. VIS 05. IEEE*, pages 367 – 374, October 2005.

[YC04] Xiaoru Yuan and Baoquan Chen. Illustrating surfaces in volume. In *Proceedings of Joint IEEE/EG Symposium on Visualization (VisSym'04)*, pages 9–16, color plate 337. the Eurographics Association, 2004.

[YLX$^+$12] Fei Yang, Qingde Li, Dehui Xiang, Yong Cao, and Jie Tian. A versatile optical model for hybrid rendering of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 18(6):925 –937, June 2012.

[YS93] Roni Yagel and Zhouhong Shi. Accelerating volume animation by space-leaping. In *Proceedings of the 4th conference on Visualization '93*, VIS '93, page 62–69, San Jose, California, 1993. IEEE Computer Society. ACM ID: 949862.

[ZDT04] Jianlong Zhou, Andreas Döring, and Klaus D. Tönnies. Distance based enhancement for focal region based volume rendering. In *IN PROCEEDINGS OF BILDVERARBEITUNG FÜR DIE MEDIZIN'04*, page 199–203, 2004.

[ZHX$^+$11] Long Zhang, Ying He, Jiazhi Xia, Xuexiang Xie, and Wei Chen. Real-time shape illustration using laplacian lines. *IEEE Transactions on Visualization and Computer Graphics*, 17(7):993 –1006, July 2011.

[ZIK$^+$98] Sergej Zhukov, Andrej Inoes, Grigorij Kronin, George Drettakis, and Nelson Max. An ambient light illumination model. In *Rendering Techniques '98*, pages 45–56. Springer-Verlag Wien New York, 1998.

[ZMK04] Christopher Zach, Stephan Mantler, and Konrad Karner. Time-critical rendering of huge ecosystems using discrete and continuous levels of detail. *Presence: Teleoper. Virtual Environ.*, 13:656–667, December 2004.