Volume Visualisation Via Variable-Detail Non-Photorealistic Illustration

by

Joanne L. McKinley

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, September, 2002

Author's Declaration for Electronic Submission of a Thesis

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The rapid proliferation of 3D volume data, including MRI and CT scans, is prompting the search within computer graphics for more effective volume visualisation techniques. Partially because of the traditional association with medical subjects, concepts borrowed from the domain of scientific illustration show great promise for enriching volume visualisation. This thesis describes the first general system dedicated to creating user-directed, variable-detail, scientific illustrations directly from volume data. In particular, using volume segmentation for explicit abstraction in non-photorealistic volume renderings is a new concept. The unique challenges and opportunities of volume data require rethinking many non-photorealistic algorithms that traditionally operate on polygonal meshes. The resulting 2D images are qualitatively different from but complementary to those normally seen in computer graphics, and inspire an analysis of the various artistic implications of volume models for scientific illustration.

Acknowledgments

I thank Bill Cowan for supervising me through this journey. I appreciate his encouragement of both academic and personal growth. Although defining my own topic was rarely easy, at the end I feel a great sense of pride and ownership that I doubt I would have otherwise. I would also like to thank Ian Bell and Dan Brown for their patience, comments, and suggestions during the revision process.

Financial support for the research presented in this thesis was provided by the National Sciences and Engineering Research Council of Canada (NSERC) and the University of Waterloo.

Many thanks go to my colleagues in the CGL for providing a fun and supportive day-to-day "working" environment.

I thank my parents for their unconditional love and support. Dan Collens deserves a special thanks. I appreciate his insight into the technical and personal challenges I faced during the pursuit of this degree, among other things....

Trademarks

GeForce3 and GeForce4 are trademarks of NVIDIA Corporation.

Maya is a registered trademark of Silicon Graphics, Inc., exclusively used by Alias|Wavefront.

OpenGL is a registered trademark of Silicon Graphics, Inc.

OpenGL Volumizer is a trademark of Silicon Graphics, Inc.

All other products mentioned in this thesis are trademarks of their respective companies.

Contents

List of Figures

Chapter 1

Introduction

The field of computer graphics generally concerns itself with the visual depiction of three dimensional (3D) objects. Such objects can arise in one of two ways: construction or measurement. Constructed objects exist solely within a digital context, and are typically produced by artists using special-purpose 3D modelling software. Since they are essentially created from scratch, the production of complex models is often quite time-consuming. Alternatively, 3D objects can be formed nearly automatically from real-world measurements. Real-world data can yield perfectly proportioned representations in a fraction of the time an artist requires to manually fashion a similar model.

Graphical objects arising by either construction or measurement can also be classified according to their mode of data representation: boundary or volume (Figure 1.1). The majority of 3D graphics models use boundary representations. The exterior surfaces of objects are represented explicitly, often by polygonal meshes. By contrast, 3D volume representations consist of some property sampled at arbitrary locations in world space, so no explicit notion of surfaces exists. This research addresses 3D objects falling in the latter halves of both classifications: physically measured objects represented as 3D volumes. With the advent of magnetic resonance imaging

(MRI) and computed tomography (CT), these representations are easy to generate but remain difficult to understand.

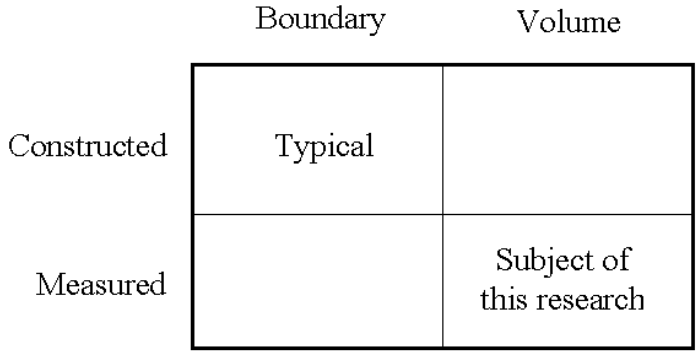


Figure 1.1: 3D Object Classifications

Incorporating well-established artistic principles into depictions of 3D volumes promises to address the relative shortage of effective volume visualisation techniques. Computer graphics researchers have only recently begun to realise the advantages of *non-photorealistic* depiction. The majority of previous work in this area involves the straightforward application of various non-photorealistic techniques borrowed from the art world (watercolour, pen-and-ink, etc.) to explicit surface representations for artistic effect. Abstraction (removing irrelevant or unnecessary detail) is a core tenet of non-photorealistic rendering, and current approaches to abstraction in 3D non-photorealistic volume rendering exhibit the shortcomings listed in Section 3.4. Since abstraction is vitally important to effective non-photorealistic communication, the lack of attention paid to explicit abstraction in volumes is somewhat surprising.

Chapter 2 contains a complete discussion of the high-level goals of this research, which are briefly as follows:

1. To specify an alternative volume visualisation technique that exhibits the advantages of non-photorealistic rendering as seen in scientific illustration.

2. To provide a semi-automatic system that allows the artist/user creative freedom without the tedium of drawing everything by hand.

3. To define a more flexible, general mechanism for abstraction than seen in previous approaches to non-photorealistic volume rendering.

4. To support a range of detail with this new abstraction mechanism.

5. To reduce the time and tedium inherent in creating scientific illustrations by hand.

6. To allow individuals other than highly trained professional artists to also create these meticulous illustrations.

SIV (Scientific Illustrator for Volumes) is a prototype software system implemented to achieve those goals. To our knowledge, SIV is the first comprehensive, general system dedicated to creating user-directed, variable-detail, scientific illustrations directly from volume data. In particular, explicitly using segmentation for abstraction in non-photorealistic volume rendering is a new concept. The system is a generalisation of and an extension to some of the basic non-photorealistic volume rendering concepts that have appeared in the computer graphics literature in recent years [ER00, TC00, Int97, CMH+01].

A fully automatic but general system is unfortunately unattainable due to several important barriers. First, as we show in Section 2.1.2, it is virtually impossible for a computer program to accurately predict the relative importance of different objects in the volume. The second problem is knowing which artistic techniques to apply to achieve a desired look. The software can help the artist navigate all the different options, but cannot evaluate final images for effective data communication.

As implemented, SIV is a software pipeline that takes a 3D volume as input and outputs a variety of 2D computer-generated illustrations. One component helps the user identify various segments and assign an *interest label* to each. The other component associates different artistic

techniques to each interest label, and renders the identified segments according to these parameters. Four different rendering techniques are provided: outlines, copper plate lines, short hatching lines, and greyscale shading.

Noise and low resolution are two common shortcomings of volume data that make rendering especially challenging and have implications for the communicative ability of the final images. Noise inherent in the scanning process affects the final images, but turns out to be a benefit for some algorithms because it helps make images that appear real and lifelike, instead of plastic and static. Low sampling rates require paying careful attention to interpolation. SIV can successfully create high resolution images from a relatively small amount of initial data.

Moreover, examining the final images, one can formulate some "rules of thumb" (useful but neither necessary nor sufficient conditions) for successful images which were impossible to predict prior to the research presented in this thesis. While SIV will clearly not replace human illustrators any time soon, several generated images do exhibit artistic merit as determined by professional artists. This realisation of artistic value is a happy side-benefit of SIV. The special qualities of volume models have both positive and negative artistic implications, yielding images that are quite unlike those normally seen in computer graphics.

Chapter 2

Motivation and Requirements

Scientific illustrations are 2D visual interpretations of 3D natural objects by specially trained artists. The subjects of MRI or CT scans are typically natural objects, and the resulting scanned volumes are simply a different interpretation (i.e. a discrete sampling in three dimensions). Our primary goal is to develop an alternative visualisation technique that effectively communicates the information resident in scanned 3D volumes in 2D images. Ideally, the generated illustrations will also demonstrate inherent "artistic merit" although we do not concretely define this term or design for it. Our semi-automatic system for generating scientific illustrations is intended to reduce the time and tedium inherent in creating these meticulous illustrations by hand while still allowing creative flexibility. A complementary goal is enabling less highly trained individuals to also create these illustrations.

2.1 Scientific Illustration

Scientific illustration is "the production of drawings of measured accuracy that help the scientist-author to communicate" [Hod89, p. xi]. Scientific illustrators are highly trained professional artists with specialised knowledge of biology, often having taken a postgraduate degree or certifi-

cate specifically in scientific illustration. Due to the large amount of specialised domain knowledge required to interpret a subject precisely and correctly, a scientific illustrator might only specialise in one or two genera in the domain of living organisms. This specialised knowledge enables the scientific illustrator to interpret the subject, adding value beyond a simple, uninterpreted photograph.

2.1.1 Measured Accuracy

The first half of Hodges's definition of scientific illustration is the requirement of measured accuracy: the subject is drawn to scale. Where possible, the artist carefully measures the specimen with a ruler to maintain proper proportion. 3D scanned volumes inherently display the property of measured accuracy; for each sampled point in a volume, the (x, y, z) coordinates in world space are known.

2.1.2 To Communicate

Hodges's definition of scientific illustration concludes with the concept of helping the scientist-author to communicate. According to Hodges, scientific illustrations should convey to the reader the same concepts that are in the mind of the author commissioning the illustration [Hod89, p. xi]. A good scientific illustrator develops this skill, using artistic techniques often applied so subtly that the undiscriminating viewer might not even notice. For example, she might completely omit an irrelevant portion of the specimen, while rendering a region of interest in great detail.

Essentially, scientific illustrators rely on their extensive artistic and scientific backgrounds, along with the visions and ideas of their commissioning authors, to decide how to best interpret a particular subject. Hence, forming a purely automatic, passive system for generating scientific illustrations is clearly impossible. Any computerised system must take the artist's vision as input, and will be semi-automatic at best.

Thus there exists a fundamental tradeoff: flexibility vs. automation. Only a skilled human can make judgments on the relative importance of portions of a subject, and evaluate the aesthetic merit of a completed illustration. Any computer programmer attempting to implement such an algorithm is necessarily imposing her judgments on the completed illustrations, and presumably she is not an expert. Thus, we propose that the best approach is to expose parameters that affect the overall artistic message, and reduce or eliminate tedious repetition otherwise.

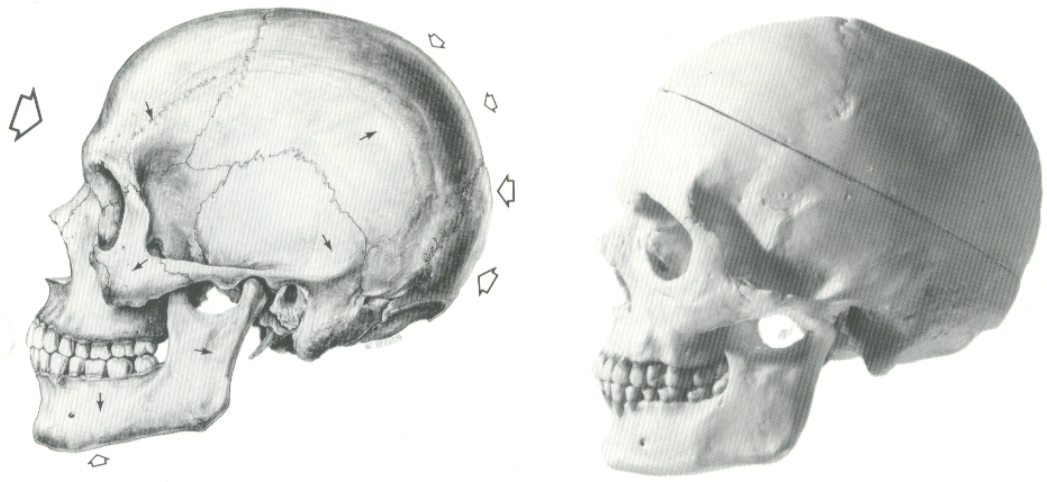
2.2 Non-Photorealism

Non-photorealism is defined by what it is not: it is not *photorealism*, which describes images that accurately duplicate what the camera perceives of the real world environment in terms of lighting, texture, scale, etc. "Non-photorealism" can refer to either computer-generated images or more traditional media, but the phrase *non-photorealistic rendering* (NPR) is associated with computer-generated images.

Figure 2.1 shows two images taken from the Hodges book [Hod89, p. 89]. The left one is an illustration and the right one is the corresponding photograph. Examining (a) by itself, one might say it looks very "realistic". Even though (b) is photorealistic, it is not nearly as successful as (a) at communicating structure, partly because key portions of the skull are obscured in shadow. While photorealism is generally useful and worthwhile, examples like this show that it is not always the best choice for conveying the essential nature of an object. According to Arnheim, a perfect likeness of some object can still be visually incomprehensible and therefore devoid of artistic content [Arn69, p. 90].

2.2.1 Abstraction

In a practical sense, abstraction implies removing irrelevant or unnecessary detail [Str98, p. 13]. For example, some photorealistic shadows in Figure 2.1 (b) are removed in the non-photorealistic



(a) (b)

Figure 2.1: "Realistic" vs. Photorealistic. Illustration and photograph by William L. Brudon. From [Hod89, p. 89], reprinted by permission of John Wiley & Sons, Inc.

rendering of Figure 2.1 (a). Most photorealistic images exhibit the same exhaustive level of detail throughout.[1] Similarly, traditional direct volume rendering is consistently detailed and displays intensity levels throughout the entire volume.

By contrast, non-photorealism regularly incorporates principles of abstraction to advantage. Certain details may not be important—a viewer may simply not be interested, or they may even detract from an image by drawing the viewer's attention away from the real object of interest. A better rendering is one that omits unnecessary detail and focuses on unambiguous, telling characteristics instead [Arn69, p. 121]. However, leaving out a particular detail does not happen in isolation; in the process, other features may be emphasised. According to Strothotte, "in practical situations there may in fact be a continuum between encoding a feature and not encoding it" [Str98, p. 14].

[1]By the above definition, it is possible for photorealism to realise a limited form of abstraction, with judicious light source placement, depth cueing, clipping planes, careful camera placement, etc. affecting the perceived level of detail. However, removing irrelevant or unnecessary detail is clearly not the major goal of photorealism.

Any software system generating scientific illustrations from volumes clearly requires a mechanism for identifying regions of interest in a volume. One way to support various levels of detail is to select portions of the volume and assign them different *interest labels*. Interest labels support a discrete range approximating a continuum between encoding a feature and not encoding it. We apply the term *explicit abstraction* to this general, flexible, user-directed mechanism that treats abstraction as an integral step in image generation. *Implicit abstraction* refers to other graphical techniques that enable abstraction as a secondary side effect.

The NPR image in Figure 2.1 emphasises different regions of the subject by locally varying light and shadow. However, a mechanism for identifying regions of interest in a volume is a prerequisite. This research focuses on specifying explicit abstraction in volumes. The local variation of light and shadow based on regions of interest is left to a future project.

2.2.2 Rendering Techniques

The artist/user requires a mechanism to associate non-photorealistic styles with the various interest labels. The provided styles must easily support a range of detail. Various flavours of greyscale rendering techniques fulfil this criterion and enjoy many advantages.

In a 3D volume, each voxel location is associated with a single intensity value. Taking MRI as an example, this value measures the magnetic field of hydrogen atoms present at a particular location. (Additional detail on the physics of MRI is given in [HB97a]). The intensity value is not correlated with colour at a location, and cannot be used by itself to calculate physically realistic colour. Thus, for simplicity, a first implementation might restrict itself to greyscale only.

However, upon closer examination, this restriction is not that serious. There is a rich tradition in scientific illustration of rendering in discrete black lines or dots on a white background. This technique, called *line illustration*, is popular partially because the renderings can be duplicated without a halftone screen. Reproduction is more detailed and cheaper than reproduction of continuous tone drawings [Hod89, p. 95]. Hodges asserts that the use of pen-and-ink (a subset

of line illustration) is the most important technique for a scientific illustrator to master [Hod89, p. 95]. Additionally, due to their linear quality and use of the same ink on the same paper, these types of illustration blend particularly well with text in a printed publication [WS94].

Line illustration is especially appropriate for rendering various levels of detail. A plain outline is the simplest way to render an object. Greater detail is provided with special pen-and-ink techniques, including stippling and cross-hatching [Hod89, p. 109].

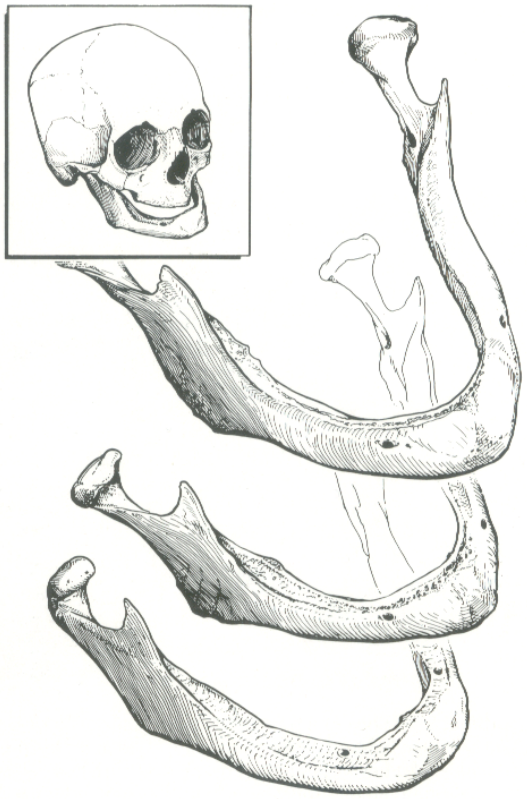


Figure 2.2: Differing Levels of Detail. *The Aging Mandible*, by William L. Brudon.

Even though line illustration is composed only of black lines on a contrasting background, in no way does this limit its expressive potential. Small irregularities in path and pressure render each individual stroke more expressive [WS94]. Strokes can actually take on a number of different

roles, to portray silhouettes, tone, texture, light and shade, perspective, modelling properties, etc. The exact placement of strokes, their relationship to other strokes, and line styles employed are the main vehicles for this communication. (Two classic books that explain more are [Gup61] and [Pit57].) Admittedly, these various and sometimes conflicting roles can sometimes be tricky for the artist to accomplish [Hod89, p. 91], but any semi-automated system should take these considerations into account. Figure 2.2 is a good example of the expressive potential of line illustration. The outlines of the skull on the upper left have varying widths, with thicker outlines suggesting shadow. On the mandibles themselves, the interior lines are curved to suggest the shape of the bone, and are broken in the areas of highlight. Careful study of this and other successful illustrations shows one the expressive potential of line illustration.

Nevertheless, the same restrictions and circumstances that exist for most scientific illustrations do not necessarily apply to computer-generated images. In particular, there is little practical advantage to generating discrete lines only. Alternatives can be considered, such as continuous tone as seen in typical pencil and wash illustrations. Since individual strokes are not discernible, one advantage of these techniques is that strokes may not accidentally be interpreted as grooves or other surface texture [Hod89, p. 120]. These forms also portray various levels of detail well, with dimmer values usually implying less detail.

2.3 Summary

The chief motivations for this research are the inherent advantages of non-photorealism for communicating an idea, especially as implemented in scientific illustration. The research community has only superficially addressed their application to volume models. The next chapter describes the current state of computer graphics research related to the ideas addressed here. The discussion so far addresses requirements at a high level only; Chapter 4 discusses the requirements of a software system implementing these ideas.

Chapter 3

Related Work

Volume visualisation and non-photorealistic rendering are the two areas in the computer graphics literature most closely related to the research presented in this thesis. Both are extremely broad, so the discussion in Sections 3.1 and 3.2 is necessarily brief and limited only to the most relevant topics. There is very little previous work combining volume rendering with non-photorealistic rendering, so Section 3.3 addresses all the major published techniques.

3.1 Volume Visualisation

Volume visualisation involves extracting and displaying information obtained from 3D volumes. Several general classes of techniques have been proposed for this purpose. Todd Elvins's 1992 survey [Elv92] provides an excellent overview of the basics of volume visualisation.

3.1.1 Direct Volume Rendering

Marc Levoy coined the term *direct volume rendering* in his seminal 1988 paper, entitled "Display of Surfaces from Volume Data" [Lev88]. This paper is one of the most often cited in volume graphics, and most subsequent papers follow his terminology. All voxels in the volume potentially

contribute to the final image, since ray casting[1] with equal sampling is used. Levoy also defines the volume rendering pipeline, which is an influential framework upon which new visualisation techniques are often based. The key result from the paper is a mechanism for displaying weak or fuzzy surfaces with smooth silhouettes and few aliasing artifacts. Since 1988, the literature has described many optimisations and enhancements to the basic method.

Rendering Pipeline

Rays are cast into the volume from the eyepoint, one for each pixel in the final image. Each ray is evenly sampled along its length. At each sample location, the intensity levels at the surrounding eight voxels are trilinearly interpolated to give an estimation for the sample. This value is optionally mapped through precomputed classification functions, yielding colour and opacity values. Classification functions are chosen to achieve certain purposes, with Levoy choosing his to emphasise surfaces in the volume. The contribution from the sample point is also independently modulated by a Phong lighting model (gradients are calculated using a central difference operator). The final colours in the image are determined by compositing colours and opacities back-to-front along the individual rays. Thus, the final rendering emphasises the overall distribution of voxels in the volume.

3.1.2 Segmentation

The purpose of segmentation is to partition the image space into meaningful regions [GW77, p. 320]. Segmentation has its roots in 2D digital image processing, and the extension to voxels in 3D volumes is straightforward. Accurate segmentation is critically important for psychology and medical research, a typical application being the separation of white matter from grey matter in 3D MRI brain volumes.

[1] Ray casting differs from ray tracing, in which rays are bounced when they hit reflective objects [Elv92]. In ray casting, rays continue in a straight line until they exit the volume. Thus, shadows do not exist in volume rendering as defined by Levoy.

A multitude of different techniques have been applied to segmentation, some general (e.g. [AB94, SSWC88, PL90]) and some exploiting specialised domain knowledge to improve the segmentation (e.g. [KSG99, TSW97]). Methods may be fully automatic (requiring no user input) or semi-automatic (requiring at least some user input). All have advantages and disadvantages, and there is no universal solution for all problem domains.

Unless explicitly mentioned, the term *segmentation* implies a discrete classification where each voxel is assigned a single type of material. Fractional segmentation (e.g. [WWG+99, NG01]) is a generalisation which computes a spectrum of materials present at each voxel. Fractional methods are more precise but are typically optimised for a specific type of dataset; for example, the Warfield technique [WWG+99] is motivated by grey matter and abnormal white matter in MRI scans having nearly the same intensity values.

Region Growing For Discrete Segmentation

Region growing methods are simple and general, relying on the assumption that neighbouring voxels of similar value probably belong to the same group. (When this assumption does not hold, fractional segmentation is more appropriate [WWG+99].) Neighbours are compared using a particular criterion of homogeneity, which varies from image to image and must be chosen carefully for a successful segmentation [AB94].

In 1994, Adams et al. [AB94] proposed seeded region growing for 2D images, which requires the input of a *seed* voxel that is representative of the desired region. The remainder of the region is identified automatically, similar to floodfill in a 2D drawing program. This form of control is intuitive for users, requiring no careful tuning of parameters. According to the authors, even unskilled users are able to achieve good segmentations on their first attempts. Thresholded region growing is another variant that renders a region growing algorithm less sensitive to noise in the raw data.

3.1.3 Surface Fitting and Reconstruction

Isosurfaces are constant-value contour surfaces in volumetric datasets [Elv92]. Surface fitting algorithms typically attempt to fit planar surfaces (usually polygons) to isosurfaces. Thus, the volume data is converted to a common representation, taking advantage of existing algorithms and special-purpose hardware for efficient rendering. Extracting a specific isosurface effectively discards the remaining information contained in the original dataset.

Marching cubes [LC87] is the most common surface fitting algorithm. A certain threshold (the isovalue) is chosen by the user. Cells of voxels whose corners are all above or all below the threshold are ignored. For the boundary cells, triangles are specially fitted according to how the "surface" intersects the cube. This algorithm is complicated to implement, with fifteen distinct special cases [LC87] of surface intersections to consider for each boundary cube.

There are many variations on surface fitting and reconstruction algorithms (e.g. [TO99, MSV95]), each exhibiting its own advantages and disadvantages. An important disadvantage common to all algorithms is that specifying the isovalue (an essentially arbitrary number) is not intuitive. Since intensity values are not correlated with any physically observable property, proper isovalue selection requires either previous knowledge of the dataset or lengthy experimentation.

3.1.4 Volume Slicing

Direct volume rendering, as described by Levoy [Lev88], is strictly software-based. Because all areas of the volume potentially contribute to the final image, rendering times are relatively slow. Offloading some of the processing to special-purpose graphics hardware promises to increase performance, especially during rotations which normally require the entire volume to be reprocessed. Experimenting with various parameter settings is the only way the user can gain insight into a particular dataset, so interactive frame rates accelerate this process.

In 1994, Wilson et al. described a method using hardware-assisted 3D texture mapping for direct volume rendering [WGW94], which later became known as *volume slicing*. First, voxels are mapped through classification functions into red, green, blue, and alpha (RGBA) values. These four channels are stored in a 3D texture map, a generalisation of a 2D texture map. A set of planes, parallel to the projection plane, slice the 3D texture like a loaf of bread. Texture coordinates are specified at the four corners of each plane. For rendering, the graphics subsystem interpolates texture coordinates across the area of each individual plane, performing the appropriate texture lookup at each location. The final image is created by rendering the planes back-to-front, accumulating colours and opacities in a compositing step.

The major advantage of volume slicing is that, after the 3D texture map is created, the specialised texture hardware can perform the slice rendering and compositing very quickly [WGW94]. Scales, translations, rotations, and perspective projections have little additional overhead. Image definition and clarity are comparable to that of software methods [WGW94]. However, 3D texturing hardware was not widely available in 1994, nor was texture memory very large. The authors were correct in believing 3D texture mapping in hardware would become more common. With its relatively low price, the NVIDIA GeForce3 family of graphics processing units, released in 2001 [NVI01], fulfilled that prediction.

Of course, the basic method was expanded upon in subsequent years. Van Gelder et al. described an extension for directional lighting [GK96]. Westermann et al. explained how OpenGL functions can be exploited for effective volume visualisation using 3D textures [WE98], achieving virtually realtime performance. The paper mentions that arbitrary clipping planes can be used to suppress portions of the dataset. Many tricks using OpenGL's various buffers are possible for other effects. These ideas and more are implemented in SGI's OpenGL Volumizer interface, the only commercially available high-level volume programming interface [Sil01]. Unfortunately, OpenGL Volumizer is restricted to expensive high-end SGI graphics systems, which is a disadvantage considering that the cheaper, ubiquitous GeForce3 also supports 3D textures.

3.1.5 Summary

Direct volume rendering (Section 3.1.1) and volume slicing (Section 3.1.4) have a common goal: the visualisation of intensity distributions within a volume modulated by various classification functions. Software-based direct volume rendering is more flexible but slower than hardware-accelerated volume slicing. However, volume slicing is only appropriate for those applications whose functionality can be implemented within the constraints imposed by the hardware architecture. For example, supporting volume rotation with light shining from a fixed direction is quite difficult with volume slicing. SIV employs volume slicing in the segmentation application where interactivity is critical, and takes advantage of direct volume rendering's flexibility in the non-photorealistic rendering application.

Segmentation (Section 3.1.2) and surface fitting algorithms (Section 3.1.3) have slightly different purposes. Classifying voxel content via segmentation is often performed as a preprocessing step; based on their contents, voxels can be treated differently further along in the volume rendering pipeline. Surface fitting algorithms eliminate the volume context by extracting a set of representative polygons. For extracting interesting regions in volumes, SIV uses seeded region growing segmentation instead of surface fitting algorithms, for reasons described in Section 4.2.3.

3.2 Non-Photorealistic Rendering

Compared to the rich history of photorealism in computer graphics, only relatively recently has the research community begun to address non-photorealistic rendering (NPR). NPR grew out of early 2D interactive paint systems [Gre99]. Modifying the classic 3D computer graphics rendering pipeline to add NPR effects is a more recent research approach [Gre99]. A wide variety of rendering effects have been described, simulating pen-and-ink, watercolour, cartoon-like cel drawing, mosaic, oil painting, and copper plate etching. The most valuable research goes beyond

simply copying an artistic medium, providing techniques that use abstraction as an explicit means of communication.

3.2.1 Geometric Buffers

Saito and Takahashi's 1990 paper, entitled "Comprehensible Rendering of 3D Shapes" [ST90], is regarded by many as the first non-photorealistic rendering paper and is still quite influential. The paper describes drawing algorithms for discontinuities, edges, contour lines, and curved hatching, using 2D image processing operations. These generated lines add to or substitute for conventional surface rendering, yielding images that more clearly communicate shape and structure. The authors claim that these techniques help make hand drawn illustrations more comprehensible, and apply them to various digital domains like edge enhancement, line drawing illustrations, topographical maps, medical imaging, and surface analysis [ST90].

The geometric buffer (G-buffer) is the mechanism proposed by Saito and Takahashi to enable line generation. G-buffers are two-dimensional *intermediate images*, formed by projecting a geometric scene onto a viewplane in a manner similar to ray tracing (hidden surfaces are implicitly removed). Each G-buffer encapsulates a single geometric property, such as depth or normal vector, of the visible object at each pixel [ST90]. Various 2D image processing techniques are then applied to the G-buffers to achieve different NPR effects. For example, profiles and internal edges can be extracted from the depth map with a first order differential operator [ST90]. Image processing operators are applied to G-buffers to obtain the final rendering. Since geometric properties are isolated in the G-buffers, they can be used multiple times provided the scene geometry and camera position do not change.

A few minor problems exist with this technique, including aliasing artifacts on surface borders, digitisation problems, and the inability to enhance reflected or transparent images [ST90]. However, Saito and Takahashi find the major problem for synthesising comprehensible images is determining the most suitable combination of enhancement techniques. Comprehensibility

depends on the scene, purpose, and viewer's preferences, so the only way to find the best combination is through trial and error [ST90]. A key advantage of the G-buffer technique is its speed,[2] which enables interactive user experimentation.

Saito and Takahashi also provide an image generated from a CT dataset as an example of an enhanced rendering. After extracting the skull from its context (possibly with segmentation, but no information is given on how they did this), they raytrace the voxel data to generate the G-buffers. The final rendering combines four techniques: profiles, shading, contour lines, and colour bands [ST90]. This is perhaps the only mention in the computer graphics literature of extracting maps of geometric information directly from voxel representations (with no intermediate polygonal representation) for non-photorealistic rendering.

Oliver Deussen extends the G-buffer approach in [Str98, p. 105-119]. He gives practical algorithms for calculating short and long hatching lines, simulating half-toning techniques. He tests his algorithms on polygonal models with slightly curved but complex geometry, which are traits shared by most medical models [Str98, p. 106].

3.2.2 Line Illustration

Detecting and Rendering Lines

Practical issues in properly detecting and rendering lines are a necessary consideration for many NPR systems. Many different techniques have been proposed (e.g. [Str86, HLW93, BS00]), which are generally optimised for different circumstances. Approaches typically fall into two categories: analytical and pixel-based. Analytic approaches are more general and flexible; being vector-oriented, these techniques are resolution-independent and can easily be transformed without creating artifacts [Str98, p. 66]. In instances where explicit analytical representations of models are not readily available, pixel-based methods (such as Saito and Takahashi's G-buffers)

[2] G-buffers can be reused without recalculation, and 2D image processing operators are generally fast.

provide an alternative. In general, pixel-based methods are faster and simpler to implement than analytical methods [Str98, p. 106].

Hertzmann presents a useful survey of image space outline generation algorithms in [Her99]. A reasonable way to generate simple outlines is by extracting edges from a depth map, observing that differences in depth should be minimal between adjacent pixels of a single object and large between objects. The same method can also be used to extract edges from a normal map, since normals should vary smoothly across a single surface but show a discontinuity between surfaces. The actual edge detection mechanism is inconsequential; Hertzmann recommends the Sobel filter (his source models are smoothly varying polygonal meshes with no noise). Each method detects different types of edges—depth edges reflect C^0 discontinuities and normal edges reflect C^1 discontinuities [Her99]. The two types of edges are combined to yield all the perceptually significant silhouettes in the source model. Of course, the detected edges only exist as lists of pixels. For further processing, such as modifying the line qualities, analytical curves must be calculated from the edge map [Her99].

Schlechtweg and Raab describe a flexible, practical line model in [Str98, p. 80-83]. The *path* expresses the course or geometry of the line, typically using a parametric curve over the interval $[0, 1]$ [Str98, p. 81]. All other deviations from the exact path, which might include pen pressure and ink saturation, are encoded in the line *style*. These attributes are then mapped into visual properties, which might be line width and brightness in this case. Lastly, Schlechtweg et al. give a scan conversion algorithm for mapping parametric curves and superimposed line styles into pixels for the final image.

Pen-and-Ink

In a 1994 paper, "Computer-Generated Pen-and-Ink Illustration" [WS94], Winkenbach et al. investigate the application of pen-and-ink illustration to architectural forms (which have a well-documented set of conventions). In the second section, the authors distill important principles

of pen-and-ink from a classic text [Gup61], although they only incorporate some of these in their system. Although their work is inspired by Saito and Takahashi, they take a somewhat different approach by directly texturing surfaces with strokes.

The pen-and-ink rendering system is basically a standard graphics pipeline, with a few notable changes. The authors begin with typical 3D polygonal models. Texture and tone are conveyed with specially pre-made, multiresolution *stroke textures*, which are applied to the various surfaces. A Phong lighting model is used to compute a reference solution, which modulates the tone production (individual strokes in the stroke texture are prioritised, enabling some to be easily omitted).

The paper elicits several interesting observations. Principles of abstraction, called *indication* by the authors, are addressed extremely briefly. The authors note that in a conventional pen-and-ink illustration, abstraction is put into practice by engaging the imagination of the viewer, rather than drawing every last stroke, and is one of the most difficult techniques for a pen-and-ink student to master [WS94]. The authors implement a semi-automatic method in which the user specifies directly on the 2D image where the detail should appear. Secondly, the authors spend considerable effort making individual strokes appear hand drawn, rather than mechanical. They vary the thickness of a line along its length, and add randomness via waviness functions [WS94].

Copper Plate Engraving

Also in 1994, Leister presented a technique for computer-generated images reminiscent of copper plate [Lei94]. Briefly, conventional copper plate printing is an old-fashioned technique in which ink is transferred from etched grooves in a copper plate to a sheet of paper. For centuries scientific publications and book illustrations used copper plate printing, because it existed long before modern reproduction techniques for photographs [Lei94]. Leister simulates copper plate etching by intersecting a scene with a set of closely-spaced, equidistant, parallel planes. A modified ray

tracer detects the surface curves resulting from these intersections. G-buffers containing geometric properties such as colour and depth modulate the final line style.

3.2.3 Automated Systems for Illustration

Several researchers have created systems that attempt to "predict" what kind of illustration a viewer would like to see. Assuming that a huge variety of possible illustrations must be narrowed down to a single one, some sort of auxiliary input is clearly necessary. Seligmann and Feiner use an intent-based approach [SF91], borrowing the concept of a knowledge base from the artificial intelligence literature. Rules that describe *communicative intent* are first defined. Stylistic decisions that reflect particular goals are used to generate the actual illustrations. The authors test their system with a set of virtual control panels.

Schlectweg et al. describe a system [SRS97] that monitors user interaction with a virtual text to infer the intent of the viewer. Supporting graphics are generated dynamically based on the user's interaction with the text, according to a set of heuristics described in the paper. Surface models are rendered and displayed with OpenGL [SRS97], but the authors note their work can also be extended to voxel models. Although the paper primarily describes the multimedia possibilities of a text-based system driving graphics generation, it is one of a very few places in the graphics literature which mentions computer-generated medical illustration, a specialised subset of scientific illustration.

Design Galleries [MAB+97] is a general approach that assists the discovery of input values that yield desirable outputs in a variety of domains. The method is especially applicable to graphics processes that are computationally expensive and/or have unquantifiable output qualities, which include NPR. Design Gallery interfaces automatically present a broad selection of perceptually different graphics to the user, who recognises and selects the appealing ones. The user finds this process less painful than manually tuning multidimensional input parameters. The authors successfully apply their method to classification function specification for volume rendering.

3.2.4 Technical Illustration

A short series of recent papers, beginning with [GSG+99], addresses computer-assisted technical illustration. Technical illustration is similar to scientific illustration in that it is typically non-photorealistic, but the subjects being depicted are non-living. Carefully observing successful technical illustrations shows a different set of conventions than those underlying scientific illustrations. Thus, the motivations behind their research are similar to those described in Chapter 2, in that both hope to bring greater insight to our respective renderings than what photorealism entails, yet their results are different due to their contextual differences.

Gooch et al. concentrate on efficient non-photorealistic rendering of 3D polygonal meshes in OpenGL. Supported effects include line weighted depth cueing, fast silhouettes, shading using cool-to-warm lighting methods, light and highlight motion, and a non-parametric method of representing metal [GSG+99].

3.2.5 Summary

The field of non-photorealistic rendering lacks well-defined boundaries because it is defined by what it is not. The term often implies superficial artistic effect, but more precisely, NPR strives for improved communication through abstraction. Well-established traditions in fine arts inspire most non-photorealistic techniques, such as line illustration described above. NPR research usually emphasises the communicative ability of novel imaging techniques over the actual algorithms used in their production.

SIV uses Saito and Takahashi's G-buffer idea extensively. Ideas from Hertzmann's survey of image space outline generation algorithms, Leister's copper plate simulation, and Deussen's image space hatching algorithms inspire some of the NPR techniques described in Chapter 6. The remainder of the content in this section is simply closely related to our work.

3.3 Combining Volume and Non-Photorealistic Rendering

Direct volume rendering, with its use of arbitrary classification functions mapping voxel intensities to RGBA values, has little basis in physical reality and is non-photorealistic in the strictest sense.[3] However, when the computer graphics community speaks of non-photorealistic volume rendering, another meaning is implied. This section describes how principles from NPR research (Section 3.2) can be applied specifically to volume models.

3.3.1 Volume Illustration

In 2000, Ebert and Rheingans [ER00] published what is probably the best-known work combining volume rendering and NPR. They augment the volume rendering pipeline with NPR techniques, expecting to display volume structure better than physics-based or classification-based renderings. Called *volume illustration*, the method produces graphical illustrations suitable for figures in textbooks, scientific articles, and educational video [ER00].

Volume illustration provides a flexible collection of NPR techniques, applied in arbitrary combinations. The techniques fall into two categories: feature enhancement and depth/orientation cues. Feature enhancement includes boundary enhancement, silhouettes, fading, and sketch lines. Cues for depth and orientation include distance colour blending, feature halos, and tone shading. To implement this the volume rendering pipeline needs one small modification. After the classification functions have been applied to a voxel, the resulting colours and opacities are subject to transformations encapsulating the volume illustration techniques. Sample location and value, gradient, minimal change direction, view direction, and light information are among the possible parameters for these transformations [ER00]. The compositing step is unchanged; all voxels in the volume still contribute to the final image.

[3] Physics-based volume rendering is an exception to this. An example is Kajiya's original work on cloud rendering [KH84], which incorporates a physically-based illumination model and atmospheric attenuation.

The volume illustration techniques are tested on a human abdominal CT dataset, and many of the final images are indeed more revealing than traditional volume renderings (for example, boundary enhancement reveals some of the kidney's internal structure [ER00]). This provides anecdotal evidence that NPR techniques can be useful in a scientific visualisation setting. Ebert and Rheingans claim that their new methods require much less manual tuning than the design of a good classification function. However, a local implementation of this work required much parameter tuning before effective images emerged. This tuning required prior knowledge of the dataset, since the tunable parameters often depend heavily on the exact voxel intensity values.

3.3.2 Lines and Contours

Treavett and Chen [TC00] were the first to apply pen-and-ink methods to volume visualisation. In their 2+D method, 2D strokes are generated based on 3D information associated with original 3D objects (G-buffers in the Saito and Takahashi terminology). Isosurfaces are extracted from the raw dataset and are treated as opaque objects for the generation of image buffers. These image buffers, representing properties including lighting, depth, normals, horizontal curvature, and vertical curvature, are used as inputs to different pen-and-ink filters to synthesise a final image. Details are lacking on exactly how individual strokes are generated. The paper shows compelling examples of these techniques applied to datasets with complex geometry, but the authors make the following strong assertions with little supporting evidence. They state that the 2+D method provides pen-and-ink drawings close to those made with the human hand in both style and quality, with "very little user interaction". They also claim to show that these techniques are a cost-effective tool for generating graphical illustrations from volume datasets, and can also enhance the process of extracting meaningful information from these datasets.

A series of papers, beginning with [Int97], describes a novel line illustration technique for visualising volume isosurfaces. Interrante first calculates the set of principal directions and principal curvatures in a volume. The perceptual psychology literature suggests that lines in the

principal directions of curvature communicate surface shape better than lines in other directions [GIHL00]. Thus, she places an evenly distributed set of tiny opaque particles on the isosurfaces of interest, and advects the particles through the vector field defined by the principal directions. The paths of the particles defined by this advection are treated as a 3D line texture, which shows the shape of the identified isosurfaces. Other advantages include viewpoint independence and the ability to visualise multiple isosurfaces simultaneously [Int97]. The author admits that, to make this technique more applicable to NPR, issues like algorithmic efficiency and artistic line quality must be addressed. In addition, showing line direction alone is not necessarily sufficient for an aesthetically pleasing rendering [GIHL00].

In contrast to the direct volume rendering and isosurface NPR techniques described above, in 2001 Csébfalvi et al. presented a fast NPR volume visualisation technique based on object contours [CMH+01]. Object contours are characterised by locally high gradient values, and give a rough estimation of physical boundaries in a dataset. Instead of using predefined classification functions, the resulting colour and opacity for a particular voxel is based only on gradient magnitude and angle between viewing direction and gradient vector. Tuning parameters for an effective visualisation is much faster than usual because there is no direct dependence on voxel intensity values [CMH+01]. However, because of this enormous reliance on gradient reconstruction, the authors find that the conventional central difference gradient estimation method is not nearly accurate enough (it only takes a narrow voxel neighbourhood into account). For better results, they recommend using a more sophisticated method based on 4D linear regression [NCKG00], which SIV employs also. Similar to this contour-based method is [CG01], in which isosurfaces are visualised as thin semi-transparent membranes similar to blown soap bubbles.

3.3.3 Miscellaneous

Two final papers combining volume rendering and non-photorealistic rendering complete this survey. The first is a generalisation of Treavett's 2000 paper [TC00] describing pen-and-ink

techniques for volumes (Section 3.3.2). In the newer paper, Treavett et al. describe how a wide variety of artistic effects can be incorporated into different stages of the volume rendering pipeline [TCSJ01]. Included in the authors' definition of "artistic" are expressive (or surreal) effects. Thus, the principle of measured accuracy is violated for artistic gain. The focus of the research seems to be less on gaining insight into particular datasets through effective visualisation techniques, and more on synthesising art from volume models.

Lum's 2002 paper [LM02] is the first to specifically use graphics hardware acceleration for non-photorealistic volume rendering. The goal is to increase efficiency so as to allow greater interactivity; the paper gives a good description of why this interactivity is desirable. Effective visualisations depend greatly on a (oftentimes large) number of associated parameters, including those for NPR techniques. There is no universally "correct" set of parameters since their settings depend on individual volume properties and visualisation goals. Thus, setting these parameters is an ad hoc, iterative, time-consuming process [LM02]. Interactivity is important because it enables quick experimentation. Lum takes advantage of some sophisticated features found in the NVIDIA GeForce3, such as 3D textures, multi-texturing, and paletted textures, to implement in hardware some of the Ebert and Rheingans volume illustration techniques [ER00] (Section 3.3.1). Lum also makes the significant, correct observation that non-photorealistic volume rendering techniques are often more complex and less efficient than their photorealistic counterparts, despite the abstractness of the resulting images [LM02].

3.4 Context

Since Section 3.3 includes all papers of which we are aware that combine volume rendering with non-photorealistic rendering, there is still room for development in the field. Our proposed assignment of interest labels to volume regions is more general and flexible than approaches to abstraction taken in previous work. For example, volume illustration [ER00] conveys the en-

tire voxel distribution throughout the volume. In terms of abstraction, it operates identically to Levoy-style direct volume rendering which also provides opacity modulation. Opacity modulation does render certain voxels more opaque or more transparent than others (usually) based on voxel intensity values.[4] However, since opacity mapping globally affects all voxels in the volume, treating voxels differently based on their spatial position within the volume is impossible. Csébfalvi's volume contour method [CMH+01] relies on opacity modulation by gradient magnitude, so it suffers from the same shortcomings regarding abstraction.

Isosurface extraction (used in Treavett and Chen's pen-and-ink approach [TC00] and Interrante's 3D line textures [Int97]) also suffers from certain shortcomings. Like opacity modulation, isosurfacing operates globally on the entire volume and ignores spatial context. Isosurfacing is essentially a binary classification; a point in space is either located on the surface or it is not. Both opacity modulation and our improved abstraction mechanism simulate a range of interest, while isosurfacing implies that the isosurface is the only interesting portion of the volume.

Since abstraction is vitally important to effective non-photorealistic communication, the lack of attention paid to quantifying regions of interest in volumes is somewhat surprising. Opacity modulation and isosurfacing in volume visualisation existed prior to to the emergence of non-photorealistic rendering as a legitimate research area, and are applied in the above papers almost as an afterthought. Certainly their suitability for supporting abstraction is never discussed. Chapter 4 describes how explicit abstraction via segmentation with interest labels can simultaneously address variable level of detail, potential differentiation based on voxel location within the volume, and ease of use compared to opacity modulation or isosurface-based methods.

Due to the lack of attention paid to non-photorealistic volume rendering, there exists a unique opportunity to explore the implications of new artistic techniques for volume models. Treavett and Chen [TC00] began this process with their work in pen-and-ink volume models, but their

[4]Roughly, opacity modulation might simulate a continuous range of interest, from opaque being very interesting to transparent being not interesting at all.

paper lacks sufficient implementation detail and its conclusions fail to convince. Hodges's book on scientific illustration [Hod89] is often used to motivate NPR research due to its candid, practical descriptions of artistic techniques, but the explicit connection between scientific illustration and biological 3D volume subjects has never previously been made. Scientific illustration incorporates a wide body of artistic techniques. Technologically, this allows the adaption of a variety of polygonal NPR algorithms ([ST90], [Her99], [Lei94], Deussen in [Str98, p. 105-119], etc.) to 3D volume models. Visually, this enables the display of the various levels of detail identified by the abstraction process.

Chapter 4

Technical Overview

To achieve our various goals, our implementation must support two key ideas. Abstraction visible in successful scientific illustrations is impossible to specify using current methods; a new approach is required. Second, these levels of abstraction must be visually realised, preferably using software simulations of successful artistic techniques applied in scientific illustrations. A software implementation has not been explicitly discussed until now; Chapters 2 and 3 describe high-level research motivations and shortcomings in the current research literature. This chapter describes the core software design decisions, providing a good background for both application specifics (Chapters 5 and 6) and results (Chapter 7).

4.1 Overall Architecture

Figure 4.1 is a schematic diagram of the software architecture. The implementation is structured as a pipeline which takes raw or generated 3D datasets as input, and outputs 2D NPR illustrations. The software implementation consists of two main pieces: the segmentation application for supporting abstraction, and the NPR application for supporting 2D NPR image generation. The main reason for this separation is practical; multiple NPR renderings can be produced with-

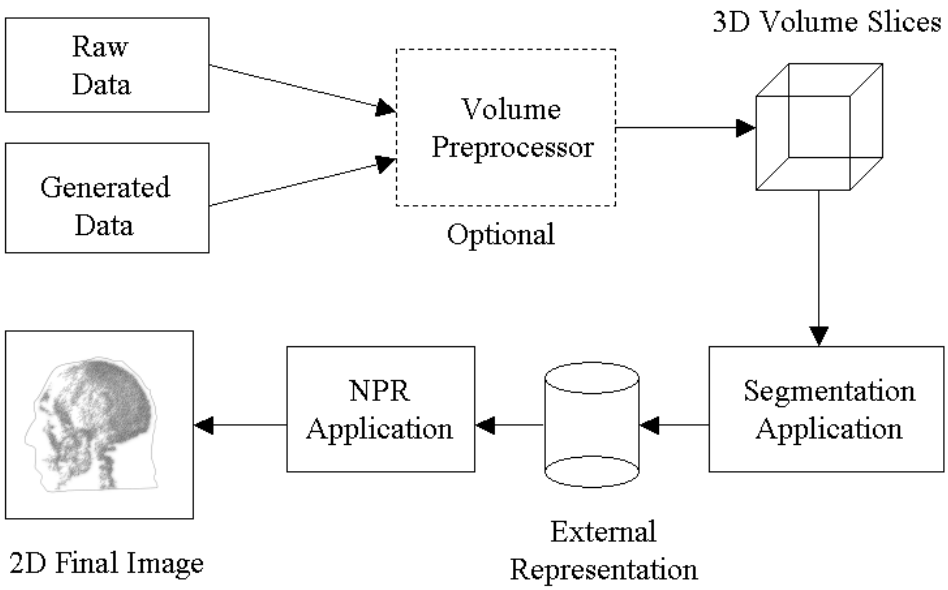


Figure 4.1: SIV Software Architecture

out having to respecify abstraction. The two applications comprise a software system we have christened SIV, short for "Scientific Illustrator for Volumes".[1]

The sole purpose of the segmentation application is to separate the 3D volume into *segments*, which abstract groups of voxels significant to the user. The input is a sequence of 8-bit 2D volume slices, a common data representation. Any volume preprocessing to convert volumes into the proper format precedes segmentation. Scripts written with the Visualization Toolkit [SML98] easily perform such operations.

The NPR application comprises the core research work, generating images which differentially display the identified segments using a subset of NPR techniques. The segmentation application, which required substantial implementation, exists only to support the NPR application. It provides the prioritised data format needed by the NPR application.

[1]SIV and *sieve* are homophones. One of SIV's purposes is filtering out unsuccessful illustrations from successful ones.

4.2 Supporting Abstraction

Three desirable characteristics for supporting abstraction within volumes are: variable levels of detail, potential differentiation based on voxel location within the volume, and ease of use. Isosurfacing and opacity modulation were used in previous non-photorealistic volume rendering attempts, but both approaches lack most of these desirable characteristics. A solution using segmentation with interest labels exhibits all three attributes, and is simple to implement (an advantage since the exact mechanism for supporting abstraction is relatively unimportant compared to the above properties). More sophisticated techniques relying less on discrete classification, such as Frisken's distance fields [Fri98] or fractional segmentation, could be substituted in the future.

Segmentation provides the general procedure for identifying objects in a raw dataset. CT and MRI measurement techniques record identical scalar values for the same physical material (modulo noise inherent in real-world measurements). A *region* is defined here as a group of connected voxels with similar intensity values, and correlates closely with distinct objects within the dataset. Helping the user identify meaningful regions within the volume is the overall goal of the segmentation application. The application provides feedback by assigning each identified region a different colour.

4.2.1 Segmentation

Many different algorithms have been proposed for segmentation (Section 3.1.2). The exact segmentation algorithm we use here is simple and modular for tuning, fast for interactivity, and accepts user assistance. We have used seeded region growing with thresholding, which is similar to floodfill in 3D.

```
Input: seed voxel S
Output: identified region R

Add S to queue Q
Loop
    Remove front voxel V from Q
    If V not yet visited
        Mark V as visited
        Add V to region R
        For each of 6 neighbours of V
            If neighbour N within threshold of S
                Add N to Q
End loop when no more items in Q
```

Like other floodfill algorithms, seeded region growing is susceptible to leakthrough in slender gaps. Narrowing the threshold can alleviate this problem, and proved quite effective as the final images in Chapter 7 demonstrate. Substituting a more sophisticated segmentation algorithm is always a future option.

4.2.2 Presenting Raw Datasets

Requiring no prior user knowledge is a main goal of the abstraction process. We want to help the user understand a previously-unknown dataset. Three main issues direct the appropriate presentation of raw datasets: interface operations, interactivity, and voxel display techniques.

We have included a few interface operations to help the user visualise the volume from any perspective. The volume can be viewed in any orientation via arbitrary 3D rotation. A clipping plane slices through the volume, exposing its innards by clipping away all data on one side of the plane. This clipping plane can also be arbitrarily rotated and translated. Since seeded region-growing segmentation allows the user to select a starting voxel, the clipping plane provides a convenient, easy way to expose the desired seed.

Interactivity is crucial for effective, satisfying operation of the user interface. Without timely feedback, rotation becomes particularly confusing. Unfortunately, software volume rendering

pipelines can be quite slow because of the memory access bottleneck. Fortunately, the consumer-grade GeForce series of graphics accelerators features 3D texture mapping in hardware. Volume slicing via 3D textures (Section 3.1.4) enables excellent interactivity.

For display, volume intensity levels must be converted into the RGBA format supported by 3D textures. Traditional volume rendering supports arbitrary classification functions mapping intensity levels to RGBA [Lev88]. Specifying useful classification functions takes practice and is often time-consuming, but can provide new insights into datasets. For flexibility, the user can modify four piecewise-linear functions associated with the RGBA display channels (affecting volume presentation in the segmentation application only). By default, these classification functions are linear ramps representing $1:1$ mappings. For example, high intensity objects appear white and nearly opaque. This default is intuitive for segmentation since objects of uniform intensity are easily perceived.

4.2.3 Assigning Interest Labels

As an operation, segmentation identifies interesting regions within a volume. Multiple regions may be added together to form a *segment*, which are conceptually considered part of the same object by the user but are not necessarily connected. As defined, segmentation is a binary classification: the act of identifying a region automatically implies interest in the region. To generalise this concept, we have quantified the level of interest in each segment by assigning it an *interest label*. The relationship between segments and interest labels is many-to-one.[2] In the terminology of Preim and Hoppe, the geometric model is augmented with structural information to form an *enriched model* [Str98, p. 42-62]. Here, the geometric model consists of the identified segments while the structural model consists of the associated interest labels. The structural model is displayed separately from the geometric model, but the two views are synchronised.

[2]SIV assumes that all voxels in a segment are equally interesting, so a single segment maps to a single interest label.

For supporting abstraction, one could also apply interest labels to multiple isosurfaces extracted from the raw dataset instead of segments. However, this solution is still inferior to segmentation because of isosurfacing's algorithmic deficiencies. Specifying a numerical isovalue is a poor alternative to seeded region growing's direct manipulation interface, and requires prior knowledge of the dataset or extensive experimentation. Without special care, extracted surfaces may exhibit poor topological properties. Typical algorithms are complicated and computationally expensive (Section 3.1.3).

4.3 Image Generation

The abstraction process yields a set of segments and associated interest labels. Allowing the user creative flexibility is the most important consideration for visually displaying the segments. Following the lead of Ebert and Rheingans [ER00], the NPR application provides a flexible toolkit of NPR techniques for generating 2D images. The user associates each interest label with subsets of NPR techniques (a one-to-many mapping), to avoid programmatic biases or assumptions. Thus, different interest labels may be rendered differently, providing a basis for visually achieving variable detail. All NPR techniques associated with all interest labels are rendered simultaneously to generate the final 2D illustrations.

We have implemented four separate NPR techniques (greyscale shading, simple outlines, copper plate lines, and short hatching lines) to offer plenty of flexibility. Our extensible and modular design allows more to be added easily. These four NPR techniques incorporate important principles of line illustration and continuous tone illustration. They can be customised by the user to portray multiple object properties, including silhouettes, surface geometry, value, light and shade, and even a primitive form of texture. Therefore two interrelated goals are achieved by setting the NPR parameters: implementing abstraction for improved communication, and realising an overall artistic vision.

Since maximum flexibility necessitates many individual parameters, we must again keep rendering speed high enough to allow for interactivity. Quick exploration of the parameter space helps the user iteratively develop more effective illustrations. A fast, pixel-based, geometric buffer approach (Section 3.2.1) applies 2D image processing algorithms to generate the NPR effects. The 3D volume is traversed once to construct the 2D buffers, and the buffers are reused when NPR parameters change. Pixel-based techniques handle large or complex datasets better than analytic alternatives [Str98, p. 105]. The algorithms are fast since much of the processing is done in 2D rather than 3D, and the G-buffers provide hidden line elimination for free. Pixel-based methods assume all objects are opaque; [ST90] shows some extensions that relax this assumption.

4.3.1 Reconstruction

High resolution illustrations communicate more information and reproduce better than smaller images. Unfortunately, most source volume models are much smaller in all dimensions than the final 2D images, so we must constantly consider reconstruction and interpolation issues. This contrasts with normal practice, where NPR images are created at the G-buffer resolution.

The discrete sampled nature of the original volume models can lead to visible artifacts, especially because of the *Nyquist limit*.[3] Variable Gaussian blurring reduces the visible effect of false low frequencies in the raw data. Higher resolution volume models would improve final image quality but are rarely available.

4.3.2 Light and Colour

Computer-generated illustrations can derive great perceptual benefits from including lighting and shading information. The NPR application employs a simplified Phong lighting model, similar to

[3]The Nyquist limit is defined as half the sampling rate. [Coo86] discusses the implications of discreteness in sampling processes. Infinite sampling of continuous surfaces in three dimensions is the only way to completely avoid artifacts.

that used by Levoy [Lev88], which operates globally on all segments. Lighting information can be incorporated into all four NPR techniques. Occlusion shadows are ignored for two reasons. Volume self-shadowing algorithms [KH84, LV00] are expensive and rarely considered crucial to volume visualisation.[4] More importantly, shadows can unnecessarily complicate a rendering, hampering effective communication that abstraction enables. In the words of Foley et al., "if the ultimate goal of a picture is to convey information, then a picture that is free of the complications of shadows and reflections may well be more successful than a tour de force of photographic realism" [FvDFH96].

For displaying illustrations on a CRT monitor, white lines on a black background are usually preferable (simulates deep space). However, black lines on a white background are better for printing, a frequent goal of automated illustration. Therefore both modes are available in SIV.

4.4 Summary

SIV is the first comprehensive, general system dedicated to creating user-directed, variable-detail, scientific illustrations directly from volume data. The NPR application required more implementation effort than the segmentation application, which exists only to support the NPR application. SIV was created to demonstrate the feasibility of semi-automatically generating scientific illustrations with more flexible, general abstraction support than previous non-photorealistic volume rendering applications. The success or failure of the software implementation depends only on the communicative ability of the resulting images. We aim for images of a similar quality to the mandible (Figure 2.2) and have been reasonably successful as demonstrated in Chapter 7.

[4]The discrete set of segments identified by the abstraction process would simplify a volume self-shadowing implementation.

Chapter 5

The Segmentation Application

This chapter describes in detail the segmentation application sketched in Section 4.2. It assumes a basic knowledge of computer graphics theory and practice, including OpenGL for simple 3D applications. Two books providing useful overviews of computer graphics are [FvDFH96] and [HB97b], while [WNDS99] is the canonical OpenGL reference.

The main window of the segmentation application contains the volume display (Section 5.2). Two secondary windows encapsulating segmentation parameters (Section 5.3.3) and interest labels (Section 5.4) are located to the right of the main window. Temporary dialogs are invoked as needed for loading volumes (Section 5.1), classification functions (Section 5.2.4), and segment export.

5.1 Loading Volumes

A volume exists on disk as a set of sequentially numbered 2D volume slices. An initial import step loads these slices into memory. Since the segmentation application is optimised for interactive speed, the user nearly always imports the entire volume. However, the import mechanism is flexible so it allows the user to load a subset of the slices, if desired.

5.2 Displaying Geometry With OpenGL

A software-based volume rendering pipeline is the straightforward method for rendering a volume onscreen. Even with clever software optimisations, rendering times cannot approach volume slicing in hardware. Volume slicing (Section 3.1.4) produces images similar to software much faster. Volume rendering using the standard OpenGL programming interface (needed for harnessing hardware functionality) is more difficult to write than software-only versions. However, the promise of a realtime interactive user interface warrants the extra work.

5.2.1 The 3D Texture

The 3D texture encapsulates the structure that is visible onscreen, and differs from the loaded volume in several ways. First, its dimensions are always a power of 2 [WNDS99, p. 373], so the original dimensions are rounded up if necessary. The original data has a single channel with volume intensities, while the 3D texture supports four channels for RGBA. Colour and opacity are needed in the visual display for classification functions and distinguishing among unique segments.

For maximum interactivity, the 3D texture must fit entirely in video memory. New generations of graphics cards continually increase the amount of specialised on-card memory available. At the time of writing, $128 \times 128 \times 128$ volumes easily fit on GeForce3 cards while GeForce4 cards support $256 \times 256 \times 256$ volumes. SIV remains functional when the 3D texture does not fit entirely in video memory, but sacrifices interactive frame rates.

5.2.2 Volume Slicing

SIV constructs a set of parallel 2D rectangles that slice through the 3D texture like a loaf of bread. Upon rendering, the closely stacked planes are indistinguishable from a rectangular solid. The result is a credible visualisation of the colours and opacities embodied in the 3D texture.

Texture Coordinates and Display Lists

Wilson's original volume slicing method [WGW94] estimates depth integration between adjacent planes by (approximately) solving a linear differential equation. The colour and opacity values placed in the 3D texture result from a non-linear mapping (dependent partially upon slice thickness). The non-linear formulae fixes the orientation of the polygons in place, parallel to the projection plane. Supporting volume rotation requires that the 3D texture be "rotated" within the set of parallel polygons, recalculating texture coordinates at each step.

Recalculating texture coordinates at each rotation is slow. Instead, SIV uses the obvious linear mapping to place colour and opacity in the 3D texture, avoiding the non-linear formulae. The parallel polygons can be legally rotated, and texture coordinates need only be defined once. This estimation method is likely worse than Wilson's depth integration, but the resulting visualisations are still very effective (especially considering their relative unimportance in the overall context of SIV's goals). Since texture coordinates do not change, the set of polygons can be placed in a *display list*. A display list is a group of OpenGL commands that have been precomputed and cached for later execution [WNDS99, p. 256]. Display lists are a well-known mechanism for improving performance when redrawing the same geometry multiple times, and are ideal for interaction with the rectangular solid.

Linear alpha blending, used by the rendering system for compositing, requires that the parallel polygons be rendered from back to front. SIV defines six display lists (two directions along each of three axes). The largest component of the view vector (x, y, or z) and its sign determine which display list to execute for a particular orientation. Thus, although a set of polygons can be viewed obliquely to some extent, they are never viewed "edge on".

5.2.3 Extra Features for Effective Visualisation

Interactive rotation of the rectangular solid is critical for proper user comprehension during the segmentation process. However, performing arbitrary 3D rotation using a 2D input device is not straightforward from the user's point of view. The virtual sphere controller, first suggested by Chen et al. in 1988 [CMS88], provides a recognisable metaphor by simulating the mechanics of a physical 3D trackball. SIV also provides an independently movable clipping plane for suppressing portions of the dataset, as originally mentioned in [GK96]. These basic operations are flexible and reasonable, allowing the viewer to examine arbitrary portions of the volume. The clipping plane also helps generate cutaway views of the volume (Section 5.3.3).

Without additional aid, perceiving volume orientation and clipping plane location can be somewhat confusing. A *bounding box* consisting of thin lines drawn at the volume perimeters provides depth and orientation cues, helping the viewer disambiguate the scene. The stippled lines representing backward-facing surfaces suggest atmospheric perspective; their relative dimness is consistent with viewing them through a translucent solid. Figure 5.1 shows these conventions, which are subtle but effective.

Volume rendering typically uses an orthographic projection, while most other applications employ a perspective projection for improved realism. SIV provides both modes for flexibility.

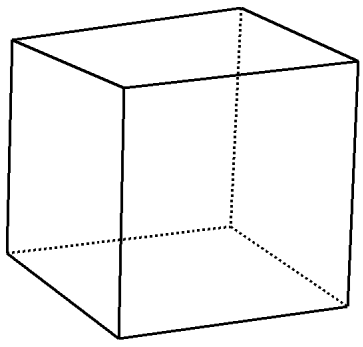


Figure 5.1: Volume Bounding Box

5.2.4 Classification Functions

Original data intensities are converted into RGBA for display with four independent classification functions (initially linear ramps). Changing the classification functions is an advanced procedure, not necessary for typical operation of the segmentation application. The functionality is provided only for instances when the user wishes an alternative version of the volume. These classification functions are represented by four miniature graphs, which are modifiable via direct manipulation. Figure 5.2 shows the graph for intensity versus opacity. Each graph has five degrees of freedom (there exists a tradeoff between generality and usability). The display does not update in realtime because the 3D texture must be reloaded for changes in the classification functions to take effect.

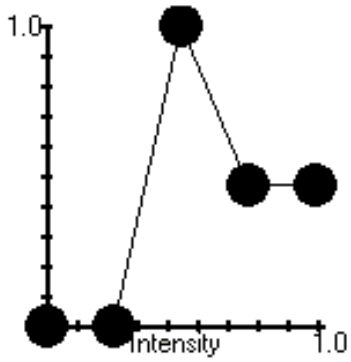


Figure 5.2: Intensity vs. Opacity Classification Function

As an alternative to the piecewise-linear function for α described above, SIV also provides a simple power function similar to that suggested by Ebert and Rheingans: [ER00]

$$\alpha = V^e$$

V is the original intensity value and the user specifies e. Exponent values less than 1 soften volume differences while values greater than 1 increase contrast within the volume [ER00]. Again,

flexibility is the only motivation for this feature. It provides yet another perspective on the data contained within the volume, and may help the user recognise interesting objects for segmentation.

5.2.5 Voxel Selection

A key feature of the segmentation application is its direct manipulation interface for selecting a seed voxel for segmentation. Given (x, y) screen coordinates where the user clicked the mouse, SIV must compute the corresponding seed voxel. SIV employs intersection tests similar to those done in ray tracing [HB97b, p. 534] to find the closest point of intersection of the pick ray with the rectangular solid. This standard technique is quick and reliable.

5.3 Segmentation

SIV assumes that each voxel comprises a single type of physical material differentiable by intensity value, and therefore each voxel belongs to (at most) a single segment. This assumption greatly simplifies the segmentation algorithm, segment display, and segment storage compared to fractional segmentation. Discrete segmentation is sufficiently flexible to generate the wide variety of illustrations shown in Chapter 7.

5.3.1 The Segmentation Algorithm

SIV employs a thresholded region-growing discrete segmentation algorithm, which operates similarly to floodfill in 3D (Section 4.2.1 gives the pseudocode). Although it has not been specifically optimised, the algorithm is linear in the number of voxels belonging to a region and runs reasonably quickly. Rendering times are typically subsecond, including reloading of the 3D texture.

After running the algorithm, SIV prints the voxel seed intensity and the number of voxels in the identified subset. Not only might it interest the user, this data can sometimes be used to improve subsequent segmentations. For example, if the most recently identified region is too

large or too small, checking the numerical intensity value of the seed voxel may help the user select a better noise threshold for next time. Printing to console is fairly unintrusive, so the user can easily ignore this information.

5.3.2 Recording Segments

The assumption that a voxel belongs to a single segment also simplifies the data structure design. SIV records the segment ordinal for each voxel in a single 8-bit *shadow volume* (up to 256 unique segments are supported). Fractional segmentation would require more memory to represent multiple simultaneous segments, compromise simplicity, and increase data structure access times.

An alternative storage method (under the discrete assumption) would be to explicitly record (x, y, z) coordinates for each voxel in an identified segment. For segmentations in which significant percentages (33% or more) of voxels are involved, the shadow volume method with its implicit voxel locations occupies less memory. The shadow volume method also features random access and is less complex to implement than this alternative.

5.3.3 An Interface For Segmentation

Figure 5.3 shows the segmentation parameters dialog, a central part of the segmentation application. The user interacts with it and the main volume display to identify regions during segmentation.

Forming Segments

Conceptually, a *segment* is a group of voxels significant to the user and are treated the same. However the segmentation algorithm identifies regions, which have the restriction that all voxels are connected. Therefore the ability to merge regions into a single logical segment is important. Although it requires the user to decide when a logical segment is complete, this mechanism is more general and comes closer to the ideal of identifying conceptual objects within the dataset.

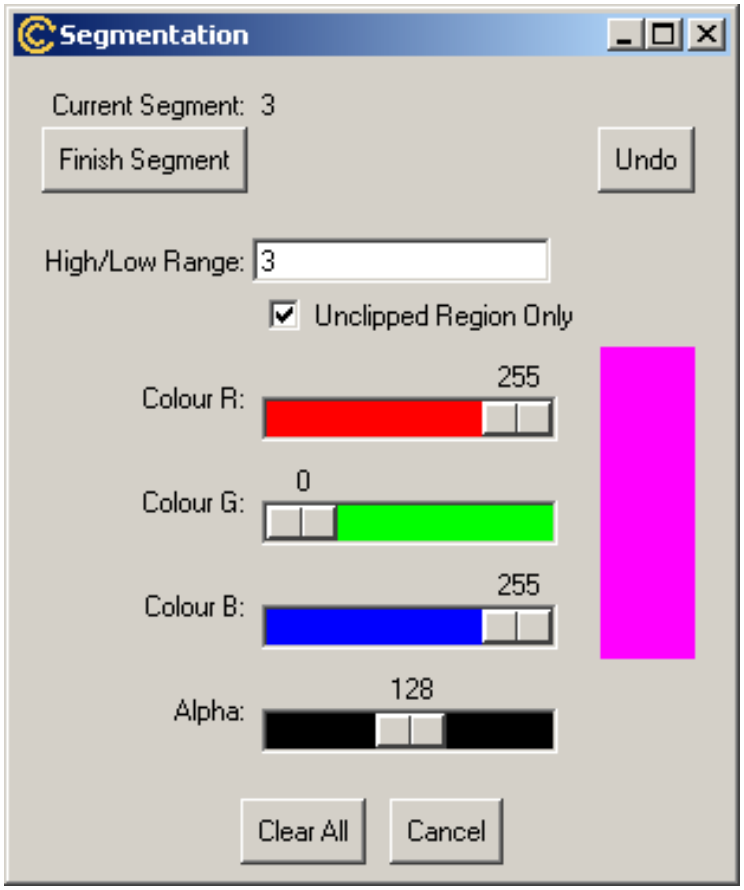


Figure 5.3: Segmentation Parameters Dialog

A sequence of segmentation operations normally identifies objects in their entirety, provided that object voxels are within the appropriate noise threshold. However, to examine cross-sections of objects, explicitly excluding parts of segments is necessary. To generate cutaway views, region-growing operations can be truncated based on the location of the clipping plane (Section 5.2.3). Supporting a single clipping plane is a reasonable compromise between flexibility and usability.

Segment Colours

On the main display, multiple segments are shown in contrasting colours. The user can easily distinguish the various segments from each other and from voxels which do not belong to any segment. The perceptual benefits are substantial and the only drawback is a slight delay as the 3D texture is reloaded. Normally the user views all segments concurrently. However, if the user wishes to examine the classified voxels as they appeared prior to segmentation, segment colours can be disabled temporarily.

The RGBA values associated with a logical segment are not appropriate for random selection; the user chooses them when beginning a new logical segment. In particular, segment opacity affects the perception of the subsequent volume. In the future, these user-specified segment colours might be incorporated into the NPR algorithms.

The assumption that a voxel belongs to a single segment also simplifies the visual display and the user's internal model of how segmentation operates. If a voxel could belong to multiple segments, the display would require either multiple volumes or an extended colour mechanism for displaying fractional segments. Supporting fractional segmentation might be a useful extension but would greatly complicate the visual display, contrary to the goal of effective visualisation.

Fixing Mistakes

Obtaining an unexpected result from segmentation is relatively easy. The user can accidentally select the wrong seed voxel, provide too generous a threshold resulting in unwanted region voxels, or provide too small a threshold in which portions of the desired region are left unidentified. Reversing the effects of past actions and restoring an object to its prior state is essential in interactive systems [ACS84], so SIV provides single-level undo functionality. Multiple-level undo is a possible future improvement, costing extra memory and/or computation time.

5.4 Interest Labels

Assigning differing interest labels to the various segments is the core mechanism for supporting variable levels of detail. The user assigns one of five interest labels to each segment in the structural view. The user completely determines how the interest labels are assigned and later used in the NPR application. Assigning interest labels with a direct correlation to levels of detail (e.g. 1 being the least detailed and 5 the most detailed) might be a useful concept for the user, but is not enforced by SIV. This design is more general and does not impose value judgments on particular artistic styles. Naturally, selecting five as the number of interest labels is a completely arbitrary decision, and SIV could be generalised to any number of interest labels.

All segments are assigned a default interest label of 3. Automatically allocating useful interest labels is impossible since SIV cannot possibly understand the user's overall artistic vision (Section 2.1.2).

5.5 Summary

The result of segmentation is a shadow volume recording the segment associated with each voxel, and a set of mappings from segment numbers to interest labels. All of the relevant information needed to recreate segments in the NPR application is exported to disk for immediate use or for future rendering. A prioritised data format is key for supporting abstraction in NPR renderings, and the segmentation application's sole reason for existence is to help the user develop such a data format.

Region-growing segmentation is simple, but works quite well. Its relative robustness in the presence of noise is a great asset. Obtaining too many or too few voxels in a single segmentation operation happens frequently, but the undo functionality repairs these cases. Luckily, anatomically "perfect" segmentations are likely not critical since the only metric for success is the artistic merit and communicative ability of the resultant NPR images.

The existence of successful final NPR images (Chapter 7) is sufficient proof that the segmentation application works well for its needs, but its true value extends beyond that. 3D texturing capabilities in consumer-grade graphics hardware are quite new, and the segmentation application showcases the future possibilities for interactive volume rendering on inexpensive systems.

Chapter 6

The Non-Photorealistic Renderer

The non-photorealistic renderer represents the chief novelty of this research and the majority of the implementation effort. The segmentation application (Chapter 5) has some interesting ideas of its own, but only exists as an aid to the non-photorealistic renderer. This piece of software generates 2D images as output, and their communicative ability determines the ultimate success or failure of the research. This chapter describes implementation details only; Chapter 7 displays and analyses images generated by SIV.

To avoid the need to resegment volumes each time NPR images are generated, segments and interest labels are stored externally on disk and the NPR application recreates them as needed. This separation is artificial, but enabled the rapid exploration of 3D NPR which is a primary goal. A tighter coupling between the segmentation application and the NPR application might prove better in day-to-day use.

6.1 A Pixel-Based Approach

Essentially, the NPR application projects 3D segments onto a 2D image plane, imitating successful principles from scientific illustration in the process. SIV takes a pixel-based approach for this

projection, imitating the G-buffers of Saito and Takahashi [ST90]. Sections 3.2.1 and 4.3 describe geometric buffers and their advantages over analytical representations.

Since SIV is the first prototype generating scientific illustrations directly from volume data, fixing the viewpoint is a reasonable restriction.[1] For simplicity, the viewing direction is the same as that of segmentation application when segments were exported to disk. Since viewing direction is constant, calculating basic G-buffers (essentially sampling a 3D volume down into a 2+D image) need only be done once. Therefore the NPR application differs from the segmentation application in that the entire volume need not be traversed for each frame. All subsequent processing is performed on the 2D G-buffers, which is typically fast even in software. Thus a software-based volume rendering pipeline is sufficient, avoiding the unintuitive volume slicing method employed in the segmentation application.

6.1.1 Ray Casting

Levoy's standard ray casting technique (Section 3.1.1) is typically used to convey the entire voxel distribution throughout a volume. SIV uses a slightly modified version to compute the initial G-buffers. The difference is that each ray is terminated when a segment voxel is found because pixel-based methods assume all objects are opaque.

First, SIV constructs the four corners of a virtual viewport in 3-space, using camera position and G-buffer size as provided by the user. One ray per G-buffer pixel is cast into the volume. SIV uses an orthographic projection, so ray directions $\vec{\mathbf{n}}$ are all orthogonal to the viewport. Orthographic projection is much more common than perspective projection in volume rendering because it simplifies ray casting.

Figure 6.1 depicts ray casting in 2D. For rays that intersect the volume, two metrics are tracked: the distance d_f to the farthest intersection point and the distance d_n to the nearest

[1] In the Western style of painting since the Renaissance, shape is restricted only to what can be seen from a fixed point of observation [Arn69, p. 32].

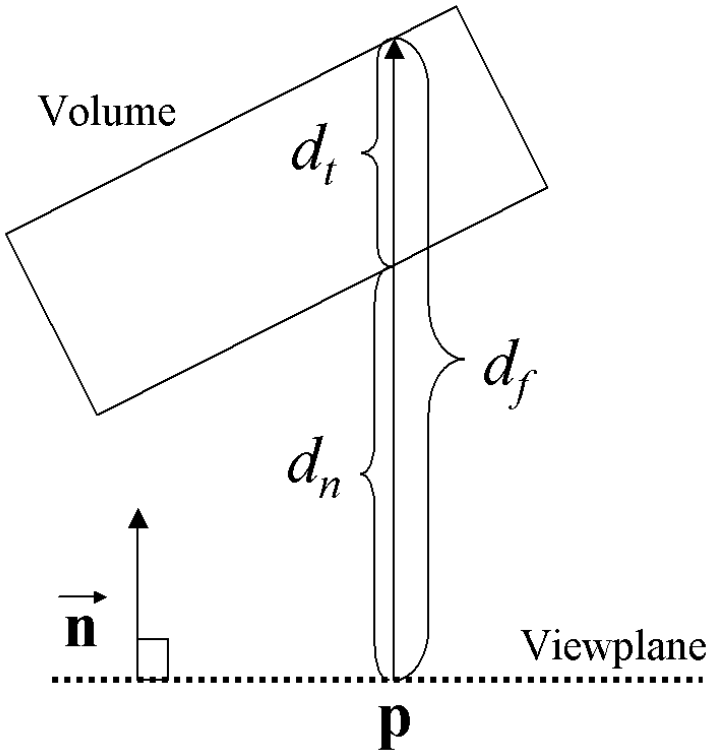


Figure 6.1: Casting a Ray

intersection point. Subtracting d_n from d_f yields the distance d_t traveled by the ray through the volume, from which the number of ray samples is calculated. For sampling, a distance of 1 voxel unit between samples guarantees that the first segment voxel along the ray is found. Thus the maximum number of ray samples (sampling front to back in the volume) is d_t. For each sample point $\mathbf{p_s}$, nearest neighbour interpolation (i.e. rounding x, y and z to their nearest integers) yields the closest voxel. Determining if this voxel belongs to a legal segment requires a discrete lookup, so trilinear interpolation is not appropriate. Terminating rays early greatly speeds up volume traversal times.

6.1.2 G-Buffer Types and Properties

SIV creates four static G-buffers from the original ray casting: depth map, normal map, world map which stores $\mathbf{p_s}$'s, and interest label map. How SIV uses these G-buffers is described later in the chapter. Figure 6.2 shows a typical depth map.

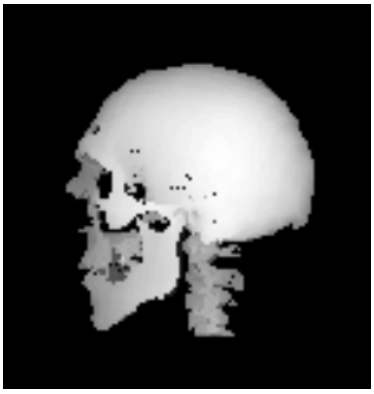


Figure 6.2: 128×128 Depth Map

The G-buffers are stored in linear arrays of double-precision floating-point values, which reduces quantisation error. Some geometric quantities are scalar but others require three channels. G-buffer pixels are zero if no segment voxel is found during ray casting.

6.2 User Interface Overview

The user interface supports two main goals: associating NPR techniques with interest labels, and customising the NPR techniques used. Associating the four NPR techniques with the five interest labels is done on the main window because this functionality is the most important. The NPR techniques are customised in a variety of satellite dialogs, which can be easily dismissed to avoid screen clutter. This grouping of UI controls by location is a logical way to organise the multitude of available parameters.

6.2.1 Main Window

The main window functions as a central command centre where the user associates the four NPR techniques with the five interest labels. For greatest creative flexibility, multiple NPR techniques may be simultaneously applied to each interest label. The disadvantage is that twenty technique/interest combinations must be explicitly included or excluded, a relatively large number.

6.2.2 Dialogs For Intermediate Image Interaction

The user customises the NPR techniques by modifying the appropriate *intermediate images*. Intermediate images are more general than G-buffers in that they may reflect multiple geometric properties per pixel. Whereas G-buffers are always the same resolution, intermediate images can be arbitrarily large. Therefore this section describes separate dialogs that visually display intermediate images and organise parameters related to their generation. To avoid overwhelming the user, this design allows the user to examine a subset of NPR parameters.

Each intermediate image simultaneously reflects all technique/interest combinations that the user includes in the main window. Each dialog also contains a variety of checkboxes, radio buttons, and textboxes for the user to provide context-specific parameters. There are two types of parameters: those that only affect the currently selected interest label (as shown on the main window) and those that affect all interest labels simultaneously. For maximum flexibility, the majority of parameters are of the former type. These have a pink background to distinguish them from parameters of the latter type, which have a grey background. Parameters of the latter type only exist for a technical reason specific to the situation. The actual 2D intermediate images are interactively updated to reflect the current set of parameters. Immediately showing the visual effect of different parameter settings helps the user narrow down the wide range of potential parameters to a satisfactory set.

Interest Label Dialog

The interest label dialog is the only one that contains no adjustable parameters. Its simple graphic portrays the interest label for each segment voxel discovered in the ray casting step. The five interest labels are each assigned a distinct, recognisable colour for display. A small legend below the 2D image conveys the label-to-colour mapping. Not intended for frequent use, the interest label dialog provides an alternative view of segmentation and proves valuable if the user forgets how interest labels were assigned in the segmentation step (can easily happen with decoupled segmentation).

Final Image Dialog

The final NPR image appears in another simple dialog. The final image consists of all user-specified NPR techniques applied simultaneously.

Final images may be saved in Portable Network Graphics (PNG) format [Wor96]. PNG is an open file format developed by the World Wide Web Consortium. It provides portable, lossless, well-compressed storage of raster images, and is supported by most common web browsers. If desired, the user can always import this file into any 2D image manipulation program for additional processing. This capability increases the scope of final image effects well beyond the four NPR techniques SIV currently offers.

6.3 Lighting Dialog

A judicious distribution of light serves to give unity and order to the shape of a complex object [Arn69, p. 255]. Lighting information can be incorporated into all four NPR techniques. For consistency and coherence, the same lighting information is used across all technique/interest

combinations.[2] The intermediate image representing lighting information is similar to a G-buffer that changes dynamically according to user input. A single white point light source, placed at infinity, shines upon the volume from a user-specifiable direction. A simplified Phong lighting contribution [Lev88] is calculated for every pixel:

$$I = V(k_a + k_d(\vec{\mathbf{n}} \cdot \vec{\mathbf{l}}) + k_s(\vec{\mathbf{r}} \cdot \vec{\mathbf{v}})) \tag{6.1}$$

I is the final lighting contribution, V is the original voxel intensity, k_a, k_d, and k_s are ambient, diffuse, and specular material coefficients provided by the user, $\vec{\mathbf{n}}$ is the computed gradient described in the next section, $\vec{\mathbf{l}}$ is the incoming light direction provided by the user, $\vec{\mathbf{r}}$ is $\vec{\mathbf{l}}$'s reflected direction, and $\vec{\mathbf{v}}$ is the viewing direction taken from the segmentation application. This simplified model assumes that all light is white, ambient light and the point light source are equal in power, diffuse and specular object colours are identical, rays do not attenuate, and material coefficients are constant across all interest labels. Since rays are parallel and projection is orthographic, the three vectors $\vec{\mathbf{l}}$, $\vec{\mathbf{r}}$, and $\vec{\mathbf{v}}$ are constants. The lighting model is simple and fast to compute while still providing an extensive variety of volume lighting effects.

The lighting model requires five independent user inputs: k_a, k_d, k_s, θ, and ϕ. The last two parameters are spherical coordinates (θ, ϕ) that specify the incoming light direction. Spherical coordinates default to $(90, 0)$, representing light raking the volume from the left. (Raking light is sometimes used in scientific illustrations to emphasise texture that might be obscured by conventional lighting [Hod89, p. 83].)

[2] A skilled scientific illustrator often breaks this convention for subtle artistic effect, locally controlling illumination in an illustration to clarify detail [Hod89, p. 89], as in the NPR image of Figure 2.1.

6.3.1 Gradient Calculation

Used in the lighting model, gradients are an estimation of surface normals in the volume. Gradient directions are precalculated from original volume data upon startup and cached for later use.

Levoy [Lev88] uses the simple but unsophisticated central difference operator for gradient estimation. This operator estimates directional derivatives by subtracting values of voxels neighbouring on either side (Figure 6.3, left). Csébfalvi et al. [CMH+01] note that conventional central difference is not nearly accurate enough for NPR applications, so, following their lead, SIV applies a more expensive and sophisticated method based on 4D linear regression [NCKG00].

4D Linear Regression for Gradients

The intensity function $f(x, y, z)$ in a neighbourhood close to a typical voxel can be approximated linearly:

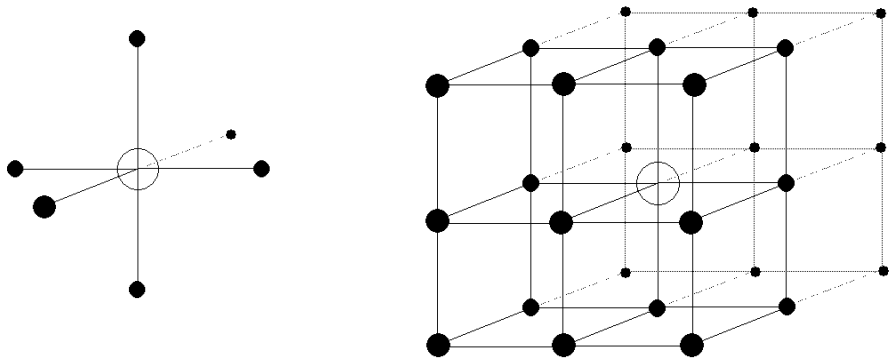
$$f(x, y, z) \approx A \cdot x + B \cdot y + C \cdot z + D \tag{6.2}$$

The vector (A, B, C) gives the gradient direction. Since this is only an approximation to the measured intensity values, the error can be measured using a mean squared error calculation. The full derivation is given in [NCKG00], but basically what would normally be an expensive calculation reduces to the following simple-to-compute equations:

$$A = \sum_{k=0}^{26} w_k f_k x_k,\ B = \sum_{k=0}^{26} w_k f_k y_k,\ C = \sum_{k=0}^{26} w_k f_k z_k,\ D = \sum_{k=0}^{26} w_k f_k \tag{6.3}$$

x_k, y_k, and z_k are the local offsets with respect to the original voxel location, and f_k is the voxel intensity value. The w_k's are weights corresponding to the distance from the neighbour voxel to the original (SIV uses Euclidean distance since it is simple and works well). An intuitive justification for the superiority of the 4D linear regression method is that it considers 26 neigh-

bours in 3-space for its estimation, compared with central difference's 6 neighbours (Figure 6.3). Consequently, 4D linear regression is more expensive to compute, but not prohibitive.



6 Neighbours 26 Neighbours

Figure 6.3: Voxel Neighbours in 3-Space

D represents the normalised weighted sum of the measured values in the local neighbourhood, and hence can be considered a filtered value. Using D in a volume rendering pipeline in place of the raw data produces smoother images [NCKG00]. SIV does not support this option, but it could prove a useful alternative to variable Gaussian blurring.

6.3.2 Gaussian Blurring for Lighting

Images created from discrete sampled volumes sometimes contain false high frequencies from insufficient sampling rates. These artifacts appear as bands, uniformly coloured areas separated by steps. This problem is most obvious in the lighting image, possibly because the human visual system is well trained in perceiving patterns of light and dark. Because lighting affects all of the NPR techniques, sometimes with a fairly direct correspondence, the user should be able to smooth the lighting image. Applying Gaussian blurring [FvDFH96, p. 629, 635] to the lighting

image provides an artistic benefit by reducing or eliminating these distracting false frequencies. The drawback of this low-pass filter is that it tends to obscure fine detail.

SIV implements the obvious algorithm for Gaussian blurring, convolution with an appropriately constructed kernel.[3] Filter coefficients are filled in according to the standard Gaussian function:

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{6.4}$$

μ is the centre of the sliding window and σ is provided by the user. Dividing through by their sum normalises the filter coefficients. The Gaussian filter is separable: $G(x, y) = G(x)G(y)$ [GV97, p. 169], so SIV convolves first in the horizontal direction and second in the vertical direction. The smoothed version of the lighting map is simple to compute due to this separability.

6.3.3 Lighting Examples

Figure 6.4 shows a variety of lighting parameters (displayed via the greyscale shading technique, configured with a white background). The laser printer used to render the hard copy of these images is a Lexmark T520 at a resolution of 1200×1200 dots per inch.

6.4 NPR Technique — Greyscale Shading

Greyscale shading is the only continuous tone NPR technique, resembling charcoal renderings or wash. The direct incorporation of properties reflected in the lighting image (especially tone, light, and shade) is the motivation behind this technique. In contrast, line drawings are less explicit in their inclusion of these cues.

[3] The reason why the intermediate lighting image is not a true G-buffer is a consequence of Gaussian blurring. Gaussian convolution averages the effect of a particular pixel over a wider neighbourhood. Thus the criterion of a single geometric property per pixel is no longer met.

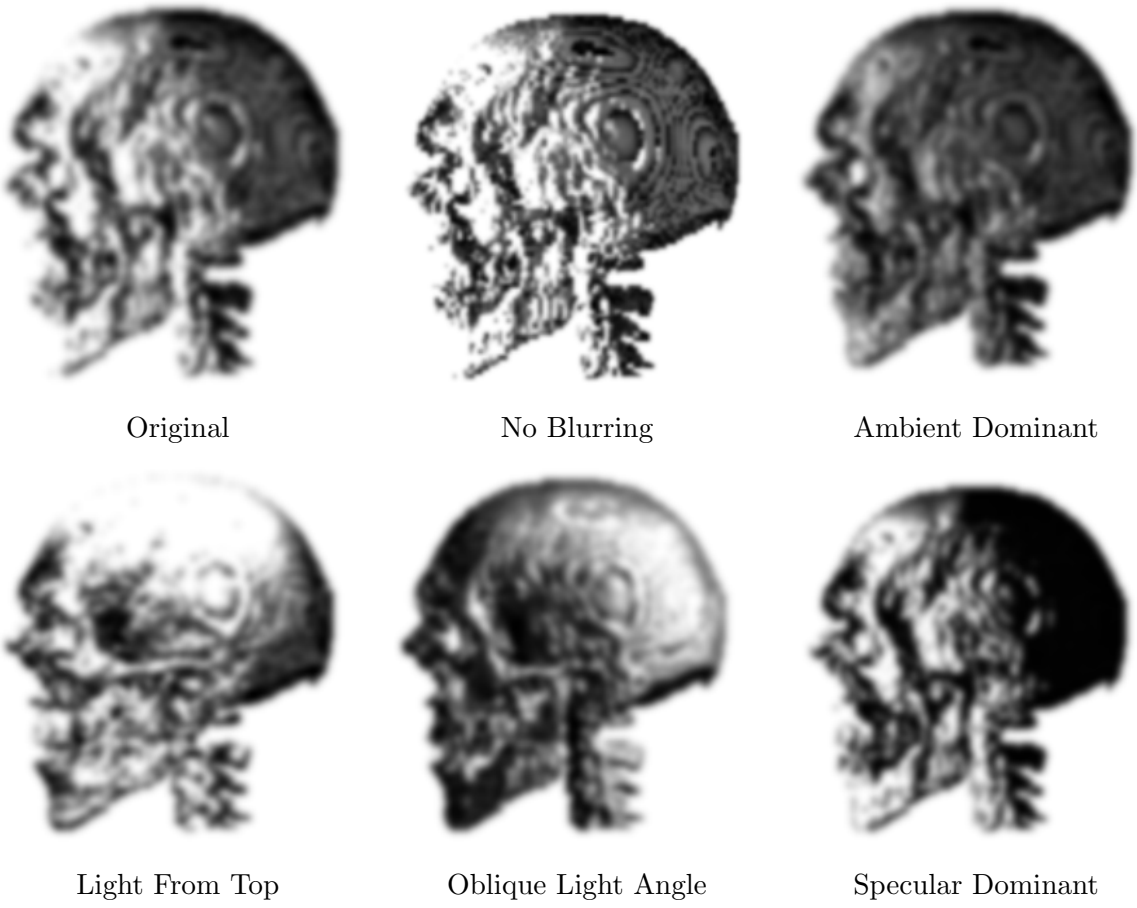


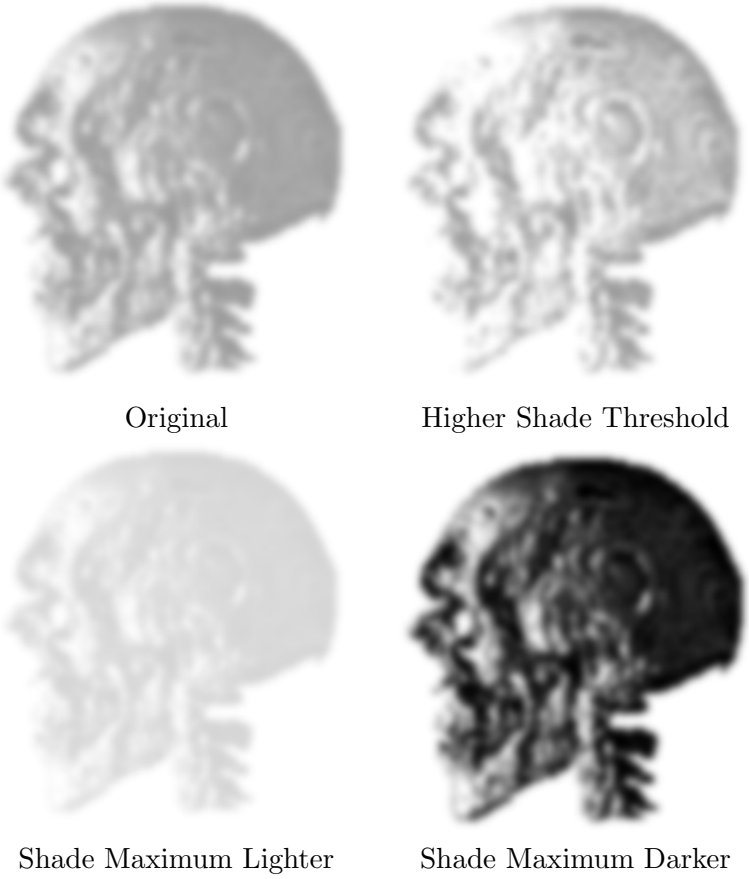
Figure 6.4: Various Lighting Parameters

Greyscale shading should offer a smooth gradation from background to lit areas of the subject, as typical charcoal renderings and washes do. This property does not hold for the light map because its lit areas are white and background is black. Therefore, to form a greyscale shaded image, SIV "reverses" the non-background entries of the unsmoothed light map.[4]

Two parameters modulate the greyscale shading appearance. The shade threshold controls how much of the reversed lighting map actually appears; as the shade threshold increases, the

[4]Gaussian convolution's spread beyond object boundaries requires that it occur after light map reversal.

lit areas blend into the background gradually until only the darkest shadows remain. The shade maximum controls the highest contrast pixel value, and all pixels are scaled to fit in that range. The shade threshold can be set independently for each interest label, while the shade maximum is set once due to its global operation on the greyscale shading image. More abstractly, shade threshold roughly controls highlight size and shade maximum influences colour contrast. A dim image probably corresponds to less detail, although the user is ultimately responsible for making this decision.



Original | Higher Shade Threshold

Shade Maximum Lighter | Shade Maximum Darker

Figure 6.5: Various Shading Parameters

Resampling with bilinear interpolation is used for enlarging the greyscale shaded image beyond its original resolution (this assumes linearity between pixels). Figure 6.5 shows some examples of the greyscale shading technique.

6.5 NPR Technique — Simple Outlines

The simple outline NPR technique aims to duplicate perceptually significant outlines and contours visualised almost automatically by artists. Simple outlines are an excellent complement to full greyscale shading, suited especially well for variable detail. By varying the line style, outlines also convey value and shading information.

The outline NPR technique was by far the most complicated to implement. SIV began with the pixel-based outline generation algorithms described in [Her99] (Section 3.2.2), which operate on G-buffers computed from polygonal meshes. Unfortunately, some aspects of the algorithms do not function well for volumes because they assume noiseless, high resolution models. Thus, their adaptation to noisy, low resolution volume models is one of the technical contributions of this research.

Pixel-based outline generation algorithms assume that C^0 discontinuities (from depth edges) and C^1 discontinuities (from normal edges) together yield all the perceptually significant outlines in the underlying model. While this assumption is not perfect, contours generated by SIV show it to be reasonable. The advantages of pixel-based outline generation over analytical alternatives are described in Sections 3.2.2 and 4.3.

6.5.1 Edge Detection Using Canny

The most important step in pixel-based outline generation is detecting edges in the depth and normal G-buffers. Hertzmann recommends Sobel, a simple derivative-based gradient edge detector. Derivative-based gradient edge detectors highlight regions of high spatial frequency, which

may or may not correspond to edges. Thus, these simple filters all suffer from the same fundamental problems: broken edges due to noise, thick bands representing edges, and numerous false positives. (Another obvious edge detection mechanism, thresholding a 3D gradient magnitude G-buffer, most likely suffers from the same problems.) The more sophisticated, non-linear Canny edge detector solves all such problems.

The Canny edge detector [Can86, DG01] is a well-known, popular edge detector for high end applications. It is the modern standard in computer vision, in the sense that new algorithms compare their results to those of Canny [HSSB98]. Assuming the presence of Gaussian noise, it is optimal according to the three criteria of good detection, good localisation, and narrow edges [Can86]. In addition, a comparative study [HSSB98] showed that Canny gives the most recognisable results; observers preferred the output from a well-tuned Canny detector over three other edge detectors. Another advantage for the programmer is that it generates ordered lists of pixels, eliminating the need for a separate pixel-to-vector conversion algorithm. Canny edge detection is rarely used in non-photorealistic rendering. The extra complexity is unnecessary for noiseless, synthetic 2D images generated from polygonal meshes.

The Algorithm

The main steps of the algorithm are as follows: [Can86, DG01]

1. Smooth the image with a Gaussian filter to reduce image details.

2. Calculate the gradient magnitude and direction at each pixel with a first derivative operator like the Roberts Cross.

3. If the gradient magnitude at a pixel is larger than those of its two neighbours in the gradient direction, mark the pixel as an edge. Otherwise, the pixel is background. This *non-maximal suppression* provides the basis for "thin" detected lines.

4. Remove the weak edges by hysteresis thresholding. Hysteresis also improves edge connectivity; noisy edges are not as likely to break into multiple line segments.

The algorithm is controlled by three parameters: the width of the Gaussian kernel used in the smoothing phase, and two thresholds used in hysteresis (t_1 and t_2, with $t_1 > t_2$). Tracking for hysteresis begins at a point on a maximal line with a gradient magnitude greater than t_1, and continues in both directions until values fall below t_2.

The basic Canny model does not consider Y-junctions, which are locations where three maximal lines meet in the gradient magnitude image. SIV detects Y-junctions and closes any gaps, ensuring that the three lines meet properly. Canny is defined on greyscale images; SIV extends the implementation to three channels so edge detection can be applied to the normal map.

6.5.2 Combining Line Segments Using The Hausdorff Distance

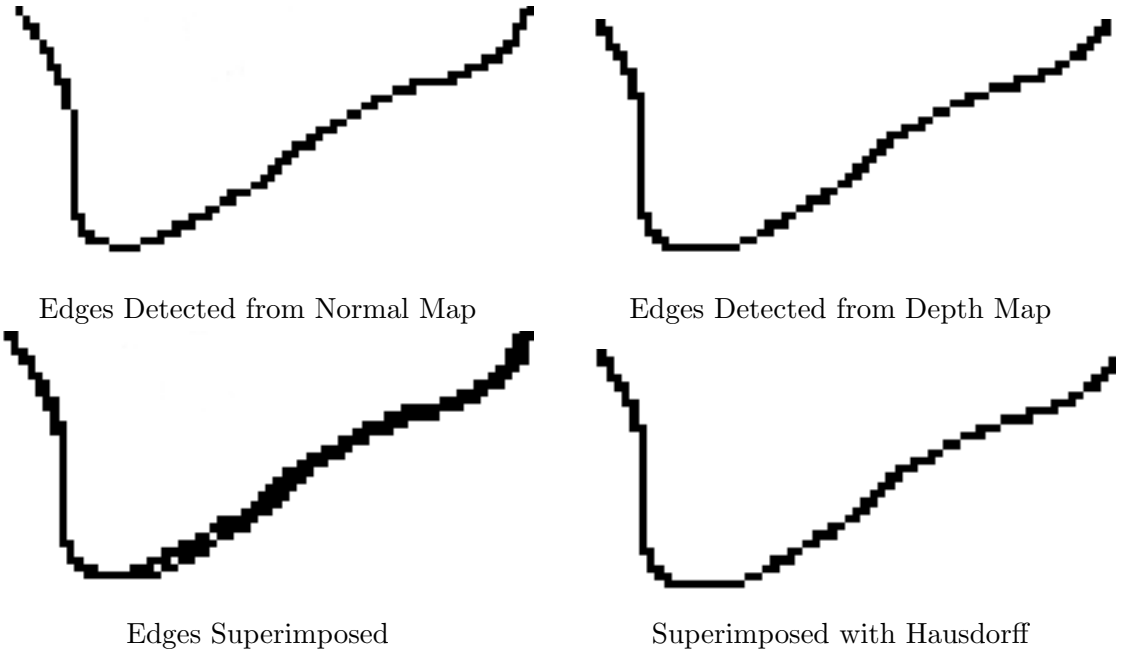


Figure 6.6: Edge Detection On the Skull Jawbone

Hertzmann provides no guidance on how depth and normal edges are combined. It turns out that, for a perceptual edge exhibiting both C^0 and C^1 discontinuities, the results rarely match exactly. Line subsets that should precisely correspond can disagree by up to several pixels due to the extreme sensitivity of Canny's non-maximal suppression. Figure 6.6 depicts this situation, magnified for clarity. The visual dissonance is extremely distracting on full-size images so SIV needs an algorithm that can sense when two edges are "almost" the same. The Hausdorff distance, described in the topology literature and then adapted to computer vision, determines the degree of resemblance between two superimposed line segments [HKR93].

Hausdorff Distance Defined

The Hausdorff distance $H(A, B)$ is defined between two finite point sets A and B:

$$H(A, B) = \max\{h(A, B), h(B, A)\} \tag{6.5}$$

where

$$h(A, B) = \max_{a \in A}\{\min_{b \in B}\|a \cdot b\|\} \tag{6.6}$$

Points a and b reside in sets A and B respectively, and $\|\cdot\|$ is some underlying norm on the points of A and B. $h(A, B)$ is called a *maximin* function because it identifies the point $a \in A$ that is farthest from any point in B, and measures the distance from a to its closest neighbour in B [HKR93]. Intuitively, the Hausdorff distance $H(A, B)$ measures the degree of mismatch between the two sets. This metric is unlike most methods of comparing shapes because there is no explicit pairing of points in A with points in B [HKR93].

An Algorithm for Eliminating Duplicate Edges

Let $\mathcal{D}$ represent the set of edges identified from the depth map, and let $\mathcal{N}$ represent those from the normal map. SIV calculates the Hausdorff distance $H(A, B) = d$ once for each pair of edges,

$A \in \mathcal{D}$ and $B \in \mathcal{N}$. If d is below a user-specified threshold t, then one of A or B is removed. By specifying the relative dominance of either depth or normal edges, the user indirectly determines which of A and B are eliminated when $d < t$. The line segments remaining after all Hausdorff distance comparisons form the image combining depth and normal edges.

SIV computes $h(A, B)$ with the obvious brute force algorithm, which runs in time $O(pq)$ for two point sets of size p and q [HKR93]. SIV uses the dot product as the underlying norm instead of Euclidean distance to avoid the computationally expensive square root operator.

6.5.3 Converting Edges to Spline-Fitted Curves

Pixelated Spline Fitted with Antialiasing

Figure 6.7: Two Approaches to Scaling Edges

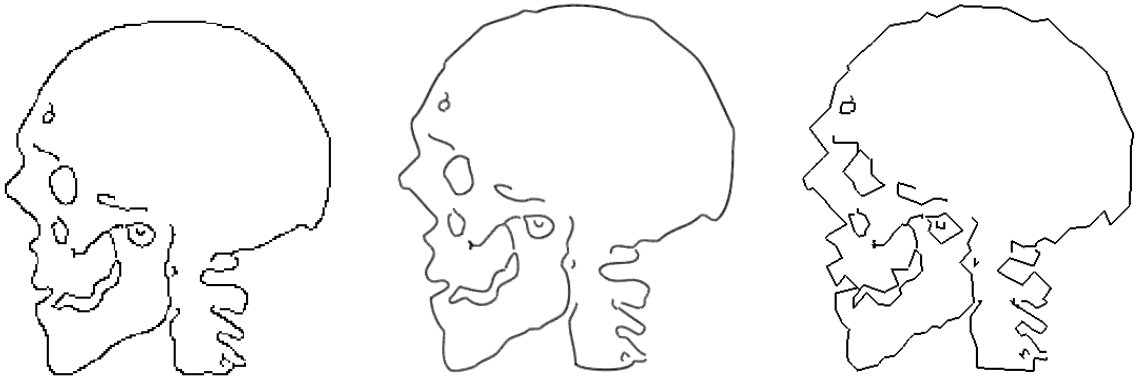
The previous algorithms output pixels at the same resolution as the original G-buffers. Naïve scaling looks very pixelated (Figure 6.7, left). The right side of Figure 6.7 shows the corresponding spline curve. Scaling a spline's control points before curve evaluation yields a smooth looking curve at all resolutions. Spline representations are also more memory-efficient.

Fitting Digitised Curves

Schneider's algorithm [Sch90] is commonly used for constructing an analytical representation from a discrete set of data points. The input is an ordered set of digitised points representing a single line segment and the output is a parametric piecewise cubic Bézier curve approximating the

original data points. The advantages of cubic Bézier segments are that they are well understood (e.g. Farin [Far88]) and can be smoothly[5] linked end-to-end.

The spline fitting problem is overconstrained because a long chain of pixels is approximated by curves with relatively few control points. Schneider solves an overdetermined system of linear equations with least squares data fitting techniques, minimising the root-mean-square error between the digitised points and the resulting curve. Schneider's algorithm is easy to control since the maximum error tolerance is the only tunable parameter. Figure 6.8 demonstrates Schneider's algorithm; the object contours of the first two images should look very similar. Tiny differences are due to user-defined error tolerance and antialiasing in the spline fitted curves.



Hausdorff Combined | Spline Fitted Curves | Control Polygons

Figure 6.8: The Effects of Fitting Splines to Identified Edges

Evaluating Curves

Given four points representing a cubic Bézier control polygon, SIV generates the corresponding polynomial curve using de Casteljau's algorithm [Far88, p. 27]. More formally, given

[5]Schneider's approach fits geometrically continuous (G^1) curves as opposed to most previous methods which fit parametrically continuous (C^1) curves [Sch90, p. 612].

$\mathbf{b}_0, \ \ldots \ , \mathbf{b}_3 \ \in \ \Re^2$ and $t \ \in \ \Re$, set

$$\mathbf{b}_i^r(t) = (1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t) \qquad \begin{cases} r = 1, \ldots, 3 \\ i = 0, \ldots, 3-r \end{cases} \tag{6.7}$$

and $\mathbf{b}_i^0(t) = \mathbf{b}_i$. Then $\mathbf{b}_0^3(t)$ is the point with parameter value t on the cubic Bézier curve.

The interval of evaluation is $[0, 1]$. The true curve requires an infinite number of evaluation points; SIV approximates it using a piecewise linear function with 20 evenly spaced points of evaluation.[6] The choice of 20 evaluation points is arbitrary, but the resulting "curves" appear smooth.

6.5.4 Line Styles

Up to this point, the described lines only communicate object contours. The viewer is reliant solely upon the spatial relationships between different lines for object recognition. Varying line properties provides a wider scope for perceptual cues and their creative application. The lighting map, with its incorporation of value and lighting information, helps the user modulate line styles.

There are three mutually exclusive options for line styles: static line width with static shade, varying line width with static shade, and static line width with varying shade. (Here, *shade* means greyscale intensity.) These options try to simulate line styles as they might be rendered using physical media. For varying line width, the light map values at Bézier control point locations are mapped into a user-specified range of widths. Widths become a third channel (in addition to x and y) for evaluation using de Casteljau's algorithm. Therefore line widths evolve gradually over the extent of each Bézier curve. Mapping shade values into the range provided by the user is a fourth channel, and is more general than the blind transfer of light map values.

[6] Although Equation 6.7 expresses a recurrence relation, $\mathbf{b}_0^3(t)$ is easier and faster to compute with a simple iterative algorithm.

OpenGL line rasterisation generally works well for rendering outlines onscreen, and avoids the need for line rasterisation routines in SIV. Generated curves are usually antialiased for smoother appearance. Without antialiasing, lines may appear jagged when examined closely.[7] The only noticeable troubles occur when rendering abnormally wide lines. Because the individual line segments comprising the line are actually rectangles, cracks are visible in portions of the line with high curvature. The problem does not often occur and is solved by increasing the number of Bézier curve evaluation points.

6.5.5 User Interface

Due to the relative algorithmic intricacy of outline generation, three separate dialogs control the process. The largest dialog encapsulates Canny edge detection, while the other two dialogs manage edge combination and spline generation respectively. Figure 6.9 shows the latter dialog. The parameters fall roughly into two groups: parameters that directly control the algorithm, and parameters that affect line style independent of geometry. This pattern is echoed in the dialogs for the remaining NPR techniques as well.

6.5.6 Commentary

NPR techniques are normally applied separately to different interest labels, enabling variable detail. In keeping with this strategy, if a particular interest label does not include outlines, then depth and normal maps are set to black wherever it occurs. Thus, the depth and normal maps used in the Canny step are a subset of the G-buffers calculated by ray casting. However, this suppression has unavoidable implications for edge detection, since strong edges always exist at the boundaries between black and non-black interest labels. (This is why edge detection is performed once, globally, for all interest labels. The alternative would identify many spatial

[7] These "jaggies" appear for the same reason as false banding in G-buffers: pixels require a discrete sampling of an otherwise-continuous line, and the sampling rate is often insufficient.

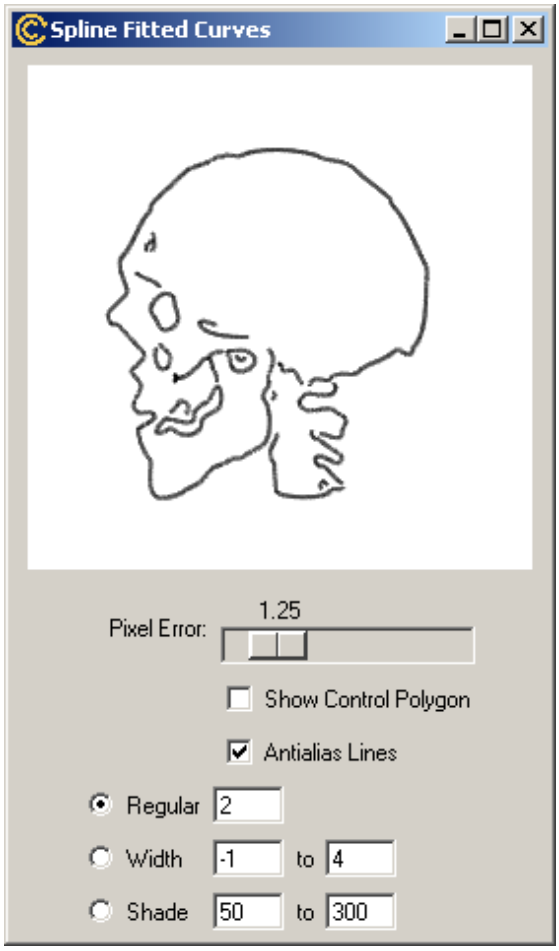


Figure 6.9: Spline Fitted Curves Dialog

boundaries between interest labels, which is not often desirable. The advantage is that the dozen parameters governing outline generation need only be set once.) Better results are achieved by performing edge detection on a complete map and carefully controlling the generated lines via Canny parameters.

Outline generation employs numerous different algorithms in software, none of which are particularly optimised. Computing Hausdorff distance is a brute force algorithm, and in digitised spline fitting, minimising the number of generated segments is more important than algorithmic

speed [Sch90, p. 623]. Regardless, outline generation still runs reasonably quickly. The user can update the various parameters in essentially realtime, which is a major goal of the 2+D pixel-based approach.

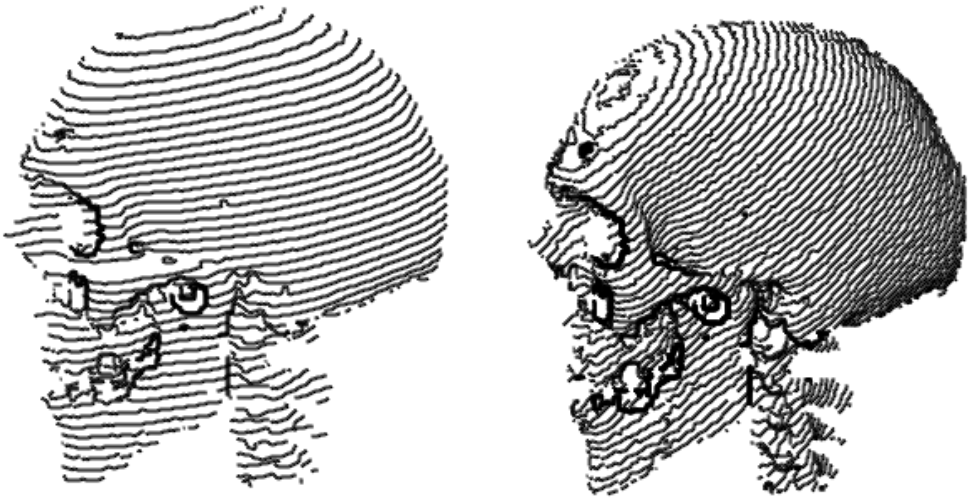
6.6 NPR Technique — Copper Plate Lines

Simple outlines are effective at showing object structure but not internal shape. Shape is one of the essential visual characteristics of objects. Long hatching lines, widely used in scientific illustration [Str98, p. 114], follow the shape of a surface. For their implementation, SIV combines ideas from copper plate illustation (Section 3.2.2) and pixel-oriented halftoning [Str98, p. 114-117]. Long hatching lines can also simultaneously convey value and shading properties as simple outlines do.

6.6.1 Constructing Parallel Planes

SIV constructs long hatching lines by intersecting volume points in 3-space with a set of 3D parallel planes. The effect is an approximation of the hatching lines seen along the jawbones of Figure 2.2. Specifying parallel planes in 3D is not as difficult for the user as it sounds, since immediate feedback reveals an obvious correspondence with hatching lines in 2D. The absolute number of computed intersection lines increases along with image size, so line *density* in screen space stays constant for all resolutions.

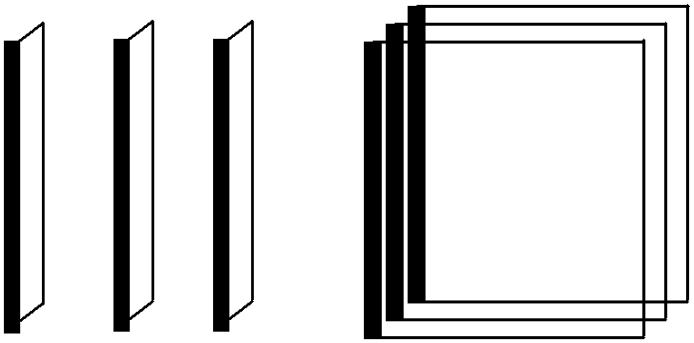
The user can rotate the planes to arbitrary orientations using trackball-style rotation as described in Section 5.2.3. However, rotating a set of planes is different from rotating a single clipping plane as in the segmentation application. As the set of planes is rotated further away from the line of sight, the 2D projections of the 3D intersections (i.e. the actual hatching lines) become closer together. This phenomenon is depicted in Figure 6.10; plane separation has not changed between the left and right images. To reduce this effect, the user can specify a different



Slight Rotation Increased Rotation

Figure 6.10: Hatching Lines Become Closer Together Upon Rotation

axis for the initial orientation of the intersection planes. The orientation of the thick lines in Figure 6.11 is identical between the left and right images, but their relative spacing in the image plane varies due to the different axes of initial orientation.

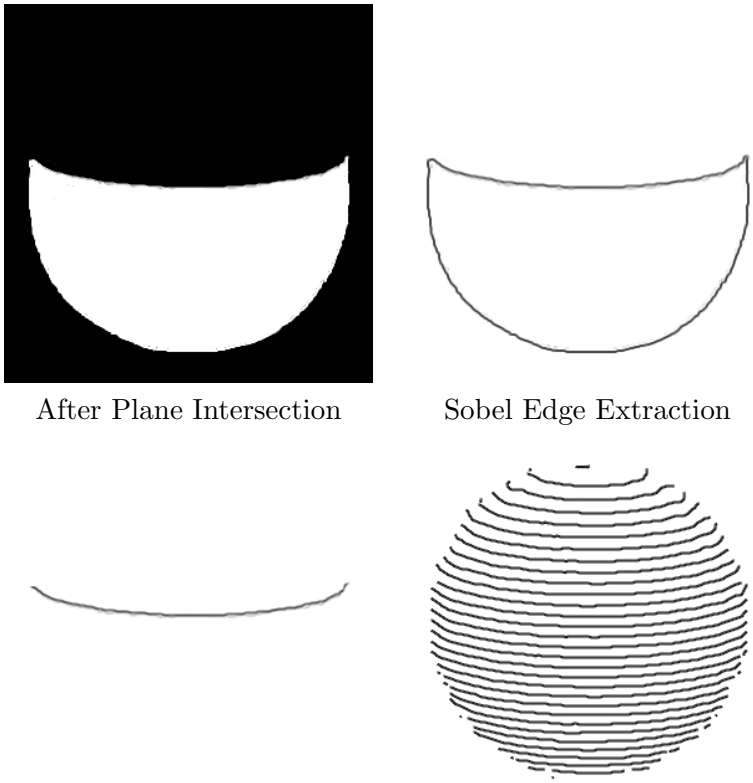


Planes Spaced Along x-Axis Planes Spaced Along z-Axis

Figure 6.11: Different Axes of Initial Orientation

6.6.2 Generating Long Hatching Lines

To support constant line density for all resolutions, intersections are computed in the full-size resolution as opposed to the original G-buffer resolution. Therefore SIV first creates an appropriately sized world map. Starting from the static world G-buffer created in the ray casting step, bilinear interpolation approximates any unknown values (bilinear interpolation among four points in 3-space yields another point in 3-space). This simple mode of interpolation works surprisingly well, with realistic results even for high resolutions.



After Plane Intersection

Sobel Edge Extraction

Subtract Whole Outline

Final Set of Lines

Figure 6.12: Calculating a Plane Intersection

Each plane is intersected individually with the constructed world map. All pixels representing points on the positive side of the plane are painted white on a black background in an intermediate image. A simple edge detector (explained shortly) extracts the edges. After subtracting the outline of the entire form, the final line of intersection remains. The process is repeated for all intersection planes, yielding the whole set of intersection lines. Figure 6.12 graphically illustrates the procedure.

6.6.3 Line Extraction

Many edge detectors generate diagonal lines that are "fat" compared to horizontal and vertical lines, as in Figure 6.6. For any lit pixel belonging to a "thin" diagonal, there are guaranteed to be at most two other lit pixels in its surrounding eight neighbours. Thin diagonals not only look better, they also greatly assist pixel-to-vector conversion. Deussen's improved edge detection operator [Str98, p. 110] based on Sobel edge detection supposedly delivers thin diagonals, although SIV required additional code for a few missing cases. The Canny algorithm is excessive for simple edge extraction from a black-and-white map, and does not handle diagonals as well as the modified Sobel. Therefore the modified Sobel proves a better choice for this limited case.

Pixel-to-vector conversion (i.e. generating an analytical representation from groups of lit pixels) is crucial for effective antialiasing; the antialiasing of individual points is not a meaningful operation. Finding the end of a line is easy since exactly one neighbour is lit. SIV traces the entire line recursively, similar to hysteresis tracking in the Canny algorithm. Due to model occlusion, obtaining multiple hatching lines per plane intersection is possible.

6.6.4 Line Styles

Scientific illustrations like Figure 2.2 often conceal hatching lines in areas of greatest highlight. Therefore the user can identify a *cutoff* parameter beyond which hatching lines are suppressed. For example, if the cutoff parameter is p, all pixels with lighting map values above p do not appear

in the final output. If the cutoff parameter is $-p$, all pixels with lighting map values below p do not appear in the final output. A cutoff parameter of 0 has no effect.

Otherwise, copper plate line segments are rendered in exactly the same way as simple outlines, and the interfaces controlling line styles are also identical. Figure 6.13 shows copper plate line renderings with a variety of parameters.

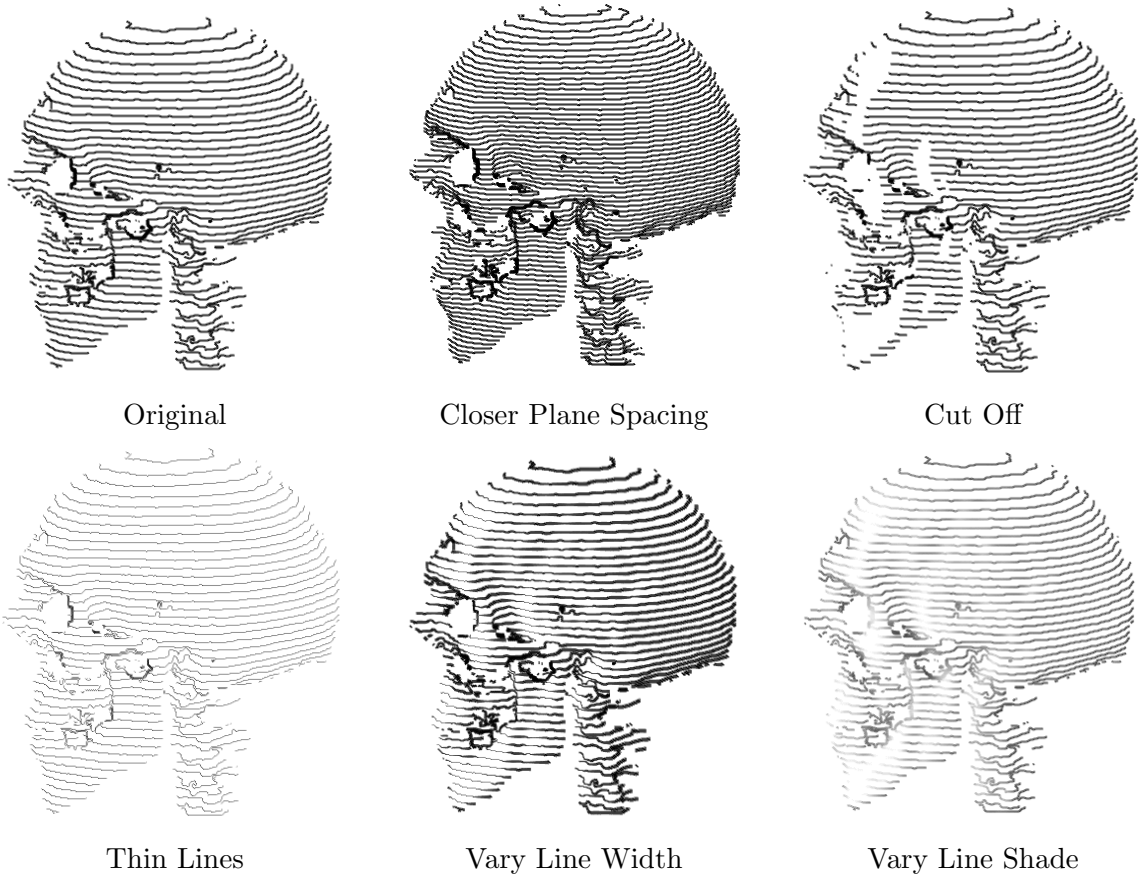


Figure 6.13: Various Copper Plate Parameters

6.6.5 Performance

Copper plate is the only NPR technique that does not update virtually instantaneously for all resolutions. 3D intersection tests (done in software for now) communicative object shape well but are computationally expensive. As image resolution increases, the number of intersection planes increases linearly ($O(n)$, where the image size is $n \times n$) to maintain constant line density. The cost of 2D image processing algorithms increases quadratically, $O(n^2)$. Since each plane intersection requires 2D image processing operations at full resolution, SIV's copper plate implementation has an $O(n^3)$ runtime, and high resolution images can take many seconds to generate. The easy solution is to tune parameters on low resolution images, which update fairly quickly, and generate the high resolution version once at the end.

6.7 NPR Technique — Short Hatching Lines

Short hatching lines (which includes a simulation of cross-hatching) is the fourth and final NPR technique. In contrast to long copper plate lines whose strength is conveying geometry, this technique's emphasis is on communicating shading information. Hence the resulting images informally resemble half-toning [FvDFH96, p. 568-573], though they use short hatching strokes of different densities instead of variable-size dots. According to Hodges, hatching is a relatively loose technique because the shading is built up at random according to the size of the drawing and the artist's skill [Hod89, p. 113].

Texture-based hatching approaches such as those found in [WS94] and [PHWF01] are well known (Section 3.2.2), but rely heavily upon a variety of artificial, pre-existing, multiresolution stroke textures. Since SIV generates 2D images from a static viewpoint, this artificial complexity can be avoided and short hatching lines are embedded directly into the image plane as in physical media. This work was inspired by [Str98, p. 112-114] and [TC00], but the algorithms presented here are new.

6.7.1 Stroke Placement

To achieve a random-looking placement of strokes, SIV *jitters* a regular grid. Jittering is a form of stochastic sampling that can be used to approximate a Poisson disk distribution[8] [Coo86], but is much cheaper to calculate. The grid of points is created at the full image resolution, so larger images have more strokes. The point locations are jittered independently in the x and y directions by adding Gaussian noise:

1. Generate a pseudo-random number p.
2. Evaluate $G(p) = q$ (Equation 6.4) using $\mu = 0$ and $\sigma = 1$.
3. Initially, q is in the range $[0, 1]$. Scale q into the range $[-\frac{d}{2}, \frac{d}{2}]$ where d is the distance between grid points.

The nature of the Gaussian distribution helps avoid excessive clustering of points. The final point locations are where the midpoints of the short hatching lines are eventually placed. Of course, not all jittered points land on actual objects to be hatched. A particular point is automatically discarded if it is not associated with an interest label including short hatching lines. Then more points are removed to help communicate shading information.

The lighting map influences stroke density of the hatched objects. This process is controlled by two shade thresholds, s_1 and s_2 with $s_1 > s_2$. For each remaining point, s is selected from $Unif(s_1, s_2)$. If the corresponding shade value in the lighting map is greater than s, the point is discarded. This procedure discriminates against bright areas in the light map, so more strokes appear in dimmer areas. Since the presence or absence of any particular point is determined stochastically, stroke density is roughly constant over similarly shaded areas. All remaining points represent strokes in the short hatching image.

[8] In a Poisson disk distribution, samples are placed randomly with the restriction that no two samples are closer together than a certain distance [Coo86]. Many graphical applications use it as the "ideal" random distribution for visual applications.

6.7.2 Stroke Generation

Once stroke placement is determined, user-defined orientation, length, and width affect the appearance of each stroke. The two endpoints of each stroke are computed analytically based on orientation and length, and OpenGL renders a line segment of the desired width.

All strokes (of the same interest label) may differ in their orientation. SIV provides two modes of stroke orientation: static and varying. In static mode, stroke orientation is constant across the image plane, regardless of object geometry. In varying mode, stroke orientation is controlled by a geometric property at each location.

Static Mode

Strokes can vary in two independent directions (for each interest label), to simulate cross-hatching. Stroke orientation alternates between the two directions with each successive stroke, and the stochastic processes ensure that the two directions are distributed approximately evenly throughout the image. Although strokes are not explicitly paired up, this even distribution enables a reasonable simulation of cross-hatching.

The user controls the two orientations with 2D rotation in the image plane. Both endpoints of the proposed line segment are tested to ensure they still fall within the proper interest label; if not, the stroke is truncated so that the original jittered point (guaranteed to lie within the proper interest label) forms one end. Constraining strokes within object boundaries helps the human visual system form implicit object contours.

Varying Mode

Alternatively, stroke orientation might vary according to a function, probably one that depends on object geometry. Following the suggestion of Meier [Mei96], SIV orients strokes according to surface normal, although another geometric property could easily be substituted. For each

point location, SIV bilinearly interpolates between the surrounding four normals in the gradient G-buffer. The 3D surface normal is projected onto the 2D image plane, and endpoint tests again constrain the stroke to the proper region.

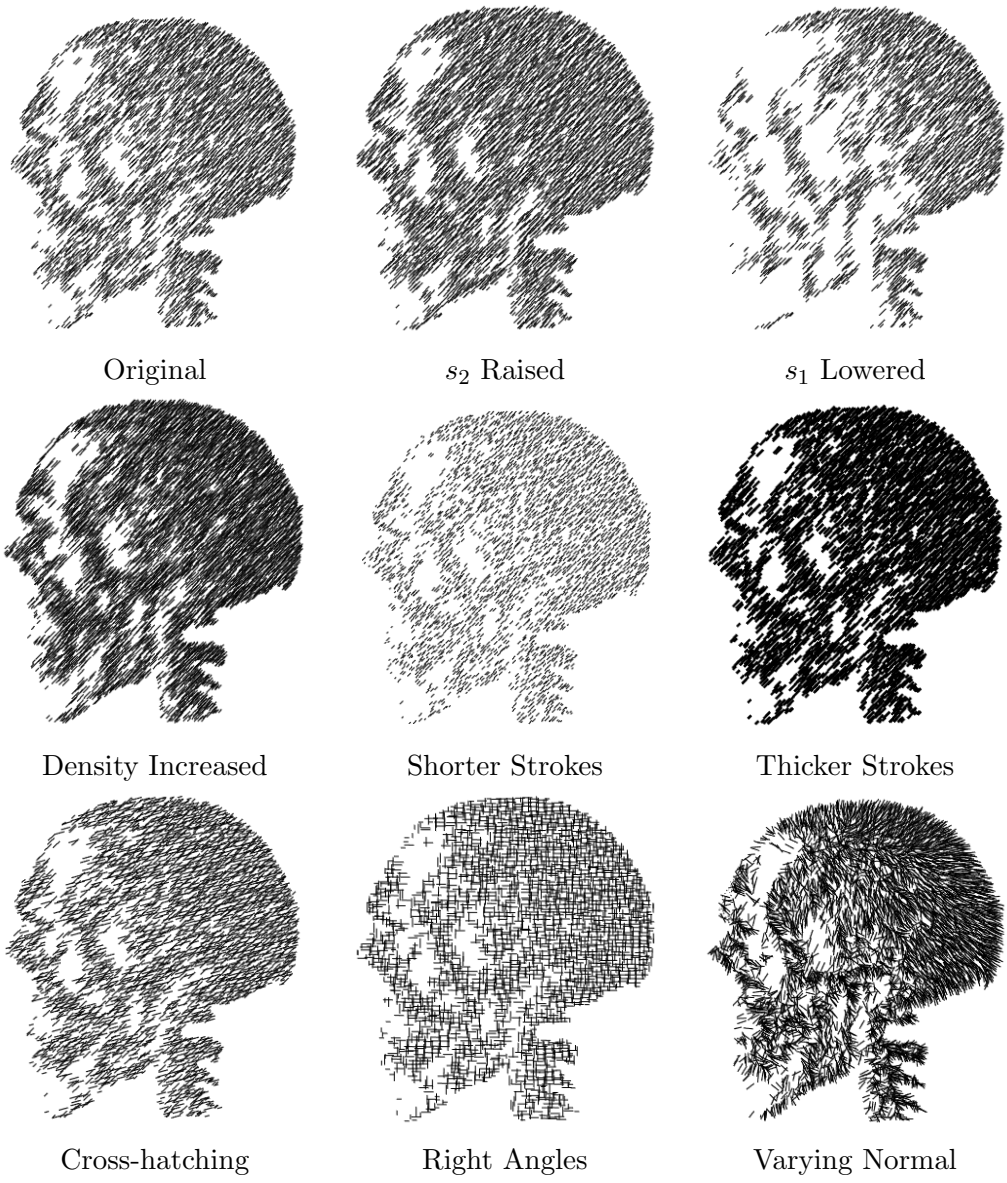


Figure 6.14: Various Short Hatching Parameters

6.8 Summary

Our work during the design and implementation of the NPR application was always informed by the tradeoff between flexibility and usability, with flexibility usually winning to enable the widest range of illustrations. Non-photorealism with abstraction involves many more degrees of freedom than does photorealism (whose only metric for success is realistic believability) and any user interface must address these extra degrees of freedom.

Unfortunately, the large number of NPR options makes a coherent interface extremely challenging to design. The current interface is general and consistent, but (likely) overwhelming for an untrained user. An associated problem, not specific to SIV, is that the mathematical significance of most parameters is meaningless to a typical artist. Animators using 3D graphics packages like Maya encounter the same difficulty. However, experienced animators learn to achieve the results they desire without the underlying mathematical knowledge. It is reasonable to expect a similar build-up of efficiency for users of SIV.

Chapter 7

Results

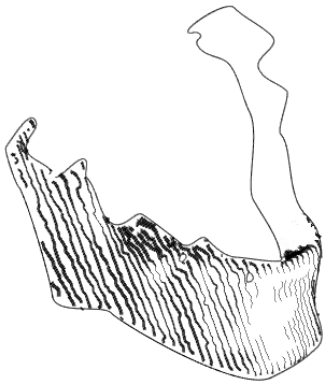
In this chapter, we analyse the results of our research. Our results are 2D images generated by the Scientific Illustrator for Volumes (SIV) and the "rules of thumb" we have inferred from them. The images mostly speak for themselves, but certain notable features are addressed in greater detail.

Figure 7.1 shows a reference illustration (a mandible from *The Aging Mandible* reprinted in Chapter 2), and two of SIV's attempts to simulate it. SIV was not created to duplicate existing illustrations exactly, but the comparison is still instructive. The physical proportions are slightly different between the simulations and the original, possibly because of changes in the aging mandible. The same low resolution $128 \times 128 \times 128$ head CT dataset that is used in most of the example illustrations in this thesis was segmented to isolate the lower jaw only. (Isolating the lower jaw only would be impossible in previous opacity modulation or isosurface-based approaches to abstraction in volumes.) Two NPR techniques, simple outlines and copper plate lines, were applied to a single interest label comprising the entire jaw. Atmospheric perspective is simulated in the right hand image by eliminating copper plate lines beyond a particular depth threshold.

Certainly the simulations are poorer than the original, but most of the differences are due to insufficient degrees of freedom in SIV as implementated. SIV assumes that all voxels in a

Aging Mandible from Figure 2.2, by William L. Brudon

Jaw with Weighted Outline



Jaw with Atmospheric Perspective

Figure 7.1: Simulating a Mandible

segment are assigned the same interest label, so the same NPR techniques are applied uniformly across the entire image. For example, outline widths must either vary or remain constant across the entire image; both styles cannot co-exist in different areas as in the reference illustration. Therefore, subsegment interest label specification would enhance SIV. The lighting could not be simulated exactly; either SIV's lighting model is too restrictive or the artist is locally varying light and shade within the image (perhaps both). The original illustration increases hatching line

density in areas of shadow, which SIV does not yet support with copper plate lines. Therefore, the copper plate lines are wider in shadowed areas to accomplish a similar effect.

Although differences are easily identified when comparing the simulations to the original, the simulated line illustration techniques (e.g. outlines with varying weight, copper plate lines indicating jaw shape, copper plate lines not shown in highlighted areas) represent a large step forward. Volume visualisation has not seen such extensive, flexible line illustration in the past. It is ironic that many of the shortcomings are due to insufficient degrees of freedom in SIV, when SIV's usability already suffers from its excess of tunable options.

Figure 7.2 would have been impossible prior to SIV. The skull and the surrounding soft tissue comprise two separate interest labels, and two NPR techniques are superimposed to form the skull. The simple outline shows the context of the surrounding soft tissue with respect to the skull, which is rendered in relatively higher detail. Naturally, SIV could produce an even better rendering in the hands of a more talented artist.

Figure 7.3 is an extremely satisfying reconstruction, about eight times the resolution of the original $128 \times 128 \times 128$ dataset. The copper plate lines are wavy as a result of noise and dataset sparseness. The inadvertent waviness is actually beneficial, realising a number of important advantages. Many real-life copper plate etchings purposefully demonstrate the same property, so SIV's simulated lines are definitely similar. The slight randomness also helps avoid Moiré patterns, a noticeable problem with many repetitive line patterns.

The waviness also affects the qualitative impact of the rendering. These lines are dynamic and alive; they almost vibrate compared to the static, artificial contours typical of most computer graphics renderings. The organic nature of the lines subtly reinforces the truth that subjects of scientific illustrations are living, biological organisms.

Figure 7.4 is a much less effective image than Figure 7.3. Figure 7.4 demonstrates several issues with the NPR techniques as implemented. Several of the generated splines appear somewhat lumpy, especially at the upper right. This is partially due to bandages wrapped around the

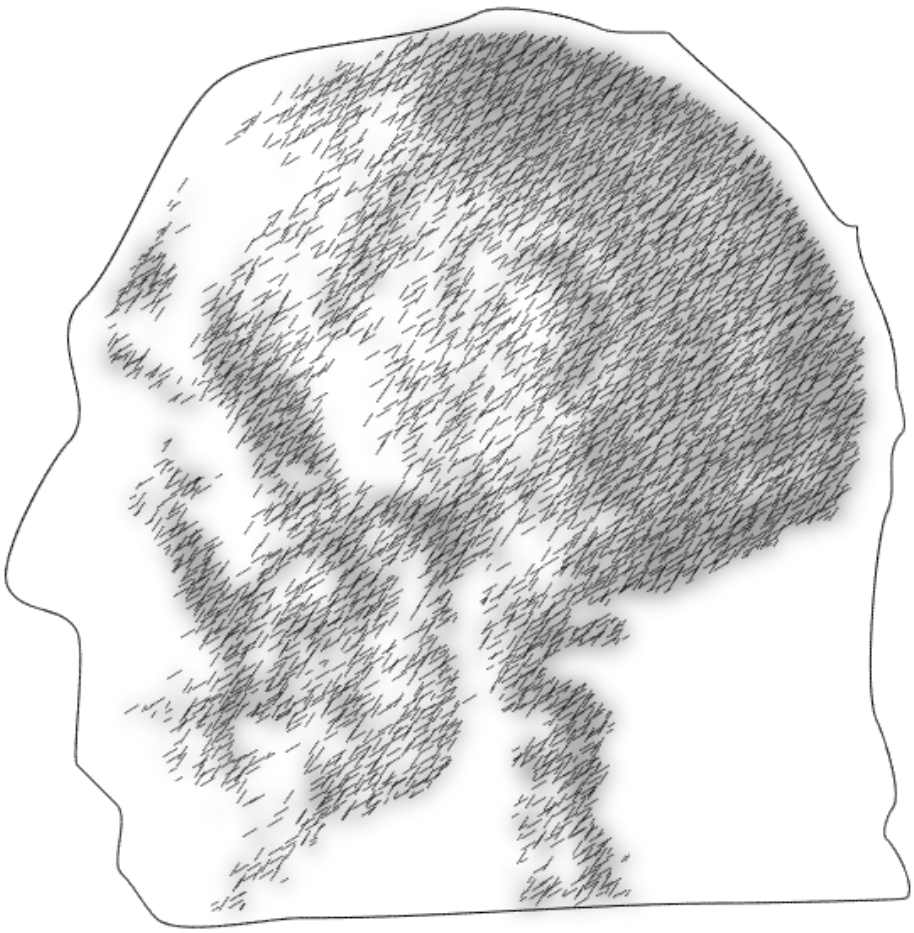


Figure 7.2: Skull With Soft Tissue Context

patient's head during the CT scan to keep it immobile, but also partially due to least-squares error fitting. The splines were fitted to a tolerance of two pixels at the G-buffer resolution, but this error is magnified along with the remainder of the image when the resolution is increased. The outline algorithms were implemented with smoothness as a major goal, although Figure 7.3 demonstrates that slight waviness is often more desirable. Winkenbach makes a similar observation about his

Figure 7.3: Copper Plate NPR Technique on Two Interest Labels

pen-and-ink system [WS94], and actually uses waviness functions to achieve a random, hand drawn effect. Unfortunately this would not address the issue of fitting error magnification.

A less tractable problem is the asynchrony exhibited between the two superimposed NPR techniques, as demonstrated along the jawline and bottom of skull in Figure 7.4. A jarring visual discontinuity exists in the underlying dataset at these locations, and the viewer expects to see the two perceptual edges (analytical outline and greyscale shading boundary) correspond exactly. Unfortunately they do not, since the two interpolation methods featured by the two NPR

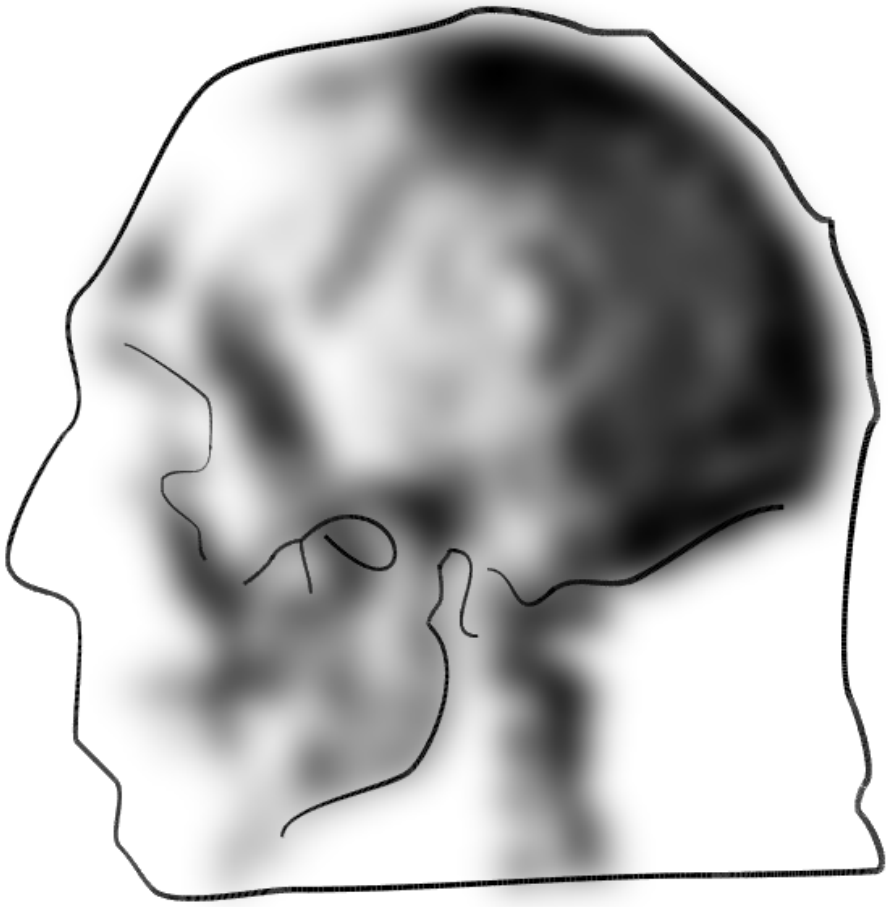


Figure 7.4: Greyscale Shading with Varying-Width Outline

techniques differ by a small but significant amount. Threshold error magnification for outlines and the averaging effect of Gaussian convolution[1] exacerbate the problem. In this particular instance, the visual dissonance causes the viewer to interpret the greyscale shading as receding while the analytical outlines come forward. Thus, a better use for analytical outlines is as shown in Figure 7.2, where they serve simply as context for the actual object of interest.

[1]Perceptual edges in greyscale shading are "spread" over a wider area as a result of Gaussian convolution. Anisotropic diffusion [McC99] is an alternative noise reduction technique that can selectively preserve edges, reducing this effect.

This fundamental misalignment issue is analogous to the *sensor fusion* problem which is encountered in many research areas. Sensor fusion involves the integration of multi-sensor information into an underlying situation-specific model. No general solution for sensor fusion exists, providing compelling evidence of this problem's general difficulty. Examining the image from further away (or reducing the image size) allows the human visual system to fuse the edges, although this solution is not particularly satisfactory.

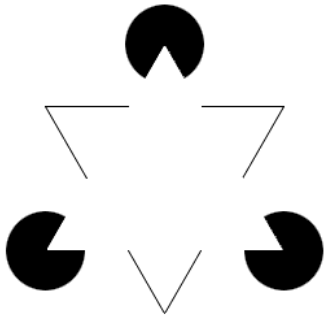


Figure 7.5: Kanisza Triangle

A much better solution is to create "virtual" edges and let the human visual system infer the boundaries. The bone/soft tissue boundary of Figure 7.3 is an effective example. At a minimum, discontinuities no longer exhibit sensor fusion problems. Figure 7.3 and the famous Kanisza Triangle (Figure 7.5) both rely on the same underlying principle of perceptual psychology, originally identified by the Gestalt psychologists. The visual system "fills in" contours not physically present in the images based on surrounding cues [Sau99]. This realisation that it is superior to rely on the human visual system to infer contours is one of the central conclusions of this research.

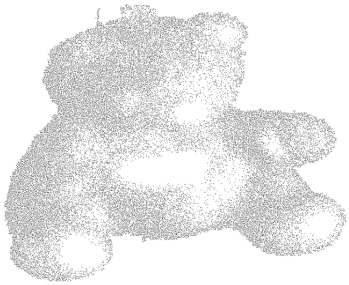
Output devices also qualitatively affect the NPR images. Many images appear sharper and clearer when printed than when displayed on a CRT monitor or wall projector. Although other unknown device-dependent factors may be involved, current CRT monitors do not have a sufficiently high resolution to do these images full justice.

Here is an example of an important perceptual problem with current CRT monitors. *Resolution acuity* refers to the eye's ability to detect a separation between objects, and *vernier acuity* refers to the eye's ability to detect a misalignment of objects [EW95]. Experiments show that vernier acuity in humans is an order of magnitude better than resolution acuity [EW95].[2] Pixel separation on CRT monitors (when viewed from a typical distance) is approximately the same as our resolution acuity threshold; effects noticeable only at the level of vernier acuity are impossible to display on today's CRT monitors. Images such as Figures 7.2 and 7.3 do look better when printed with a laser printer (which has a finer resolution) or when viewed from a greater distance, so they likely exhibit certain subtle features appreciable only at vernier thresholds. (The slight waviness of copper plate lines is a prime example; distinguishing slight variations in line alignment is the classic test of vernier acuity. Observing real-life copper plate etchings from a typical viewing distance corroborates this argument. Waviness should be an *impression*, rather than a key line feature that could be mistakenly interpreted as structure.) Simply put, CRT pixels are too large for characteristic viewing distances.

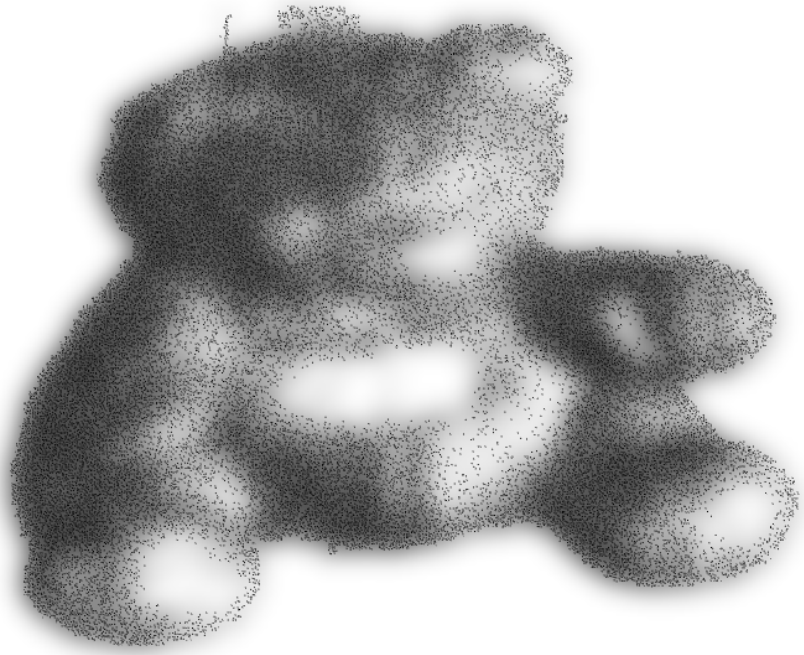
Figure 7.6 is an example of an effective image created from two constituents that are not particularly effective on their own. The subtle shading of the greyscale technique complements the suggestion of detail provided by short hatch lines. The four NPR techniques exhibit different aesthetic characteristics that, when mixed in certain combinations, balance and harmonise with one other.

[2]Vernier acuity is also known as hyperacuity, because humans can actually resolve detail with an accuracy better than one fifth of the size of the most sensitive photoreceptor in the eye [EW95].

Shade Only



Short Hatch Only



Teddy Bear with Both Techniques

Figure 7.6: The Whole is Greater Than the Sum of Its Parts

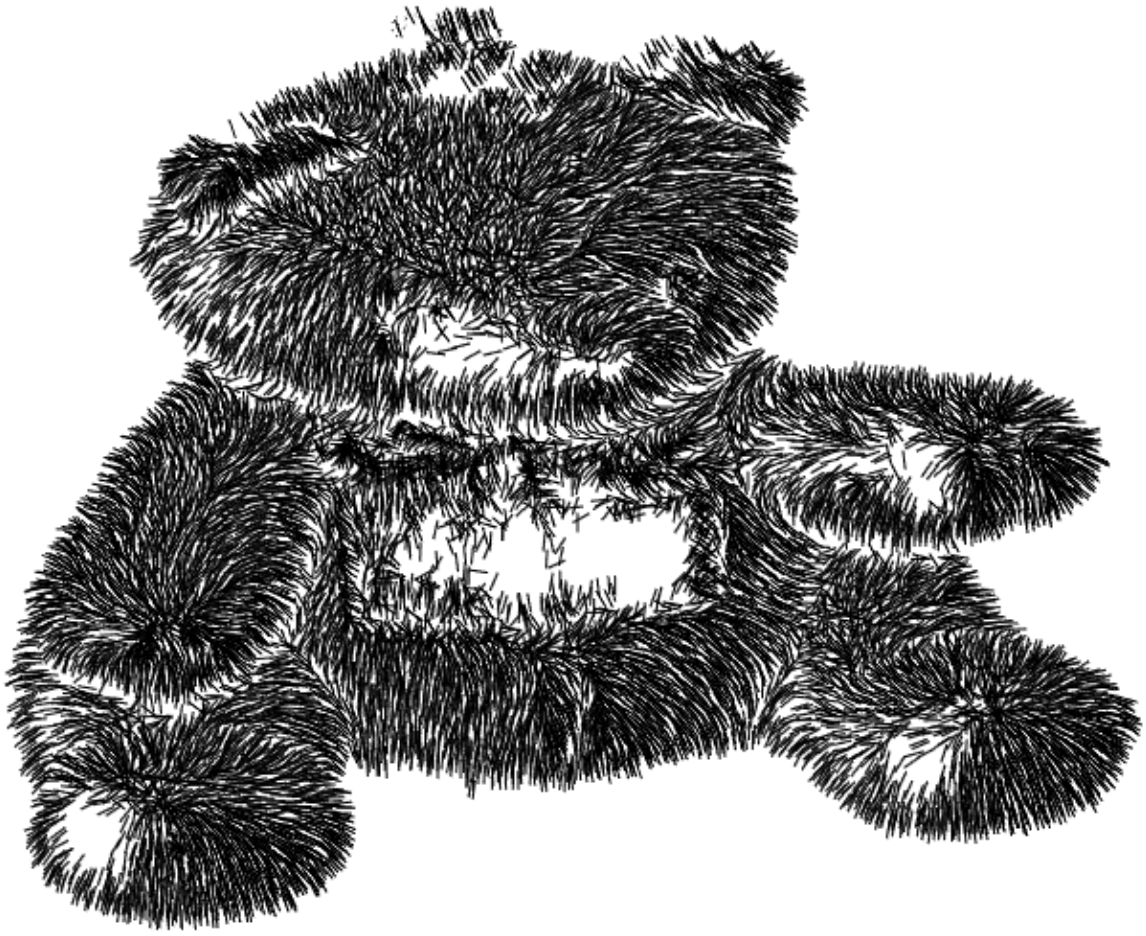


Figure 7.7: Short Hatch Teddy Bear with Varying Normals

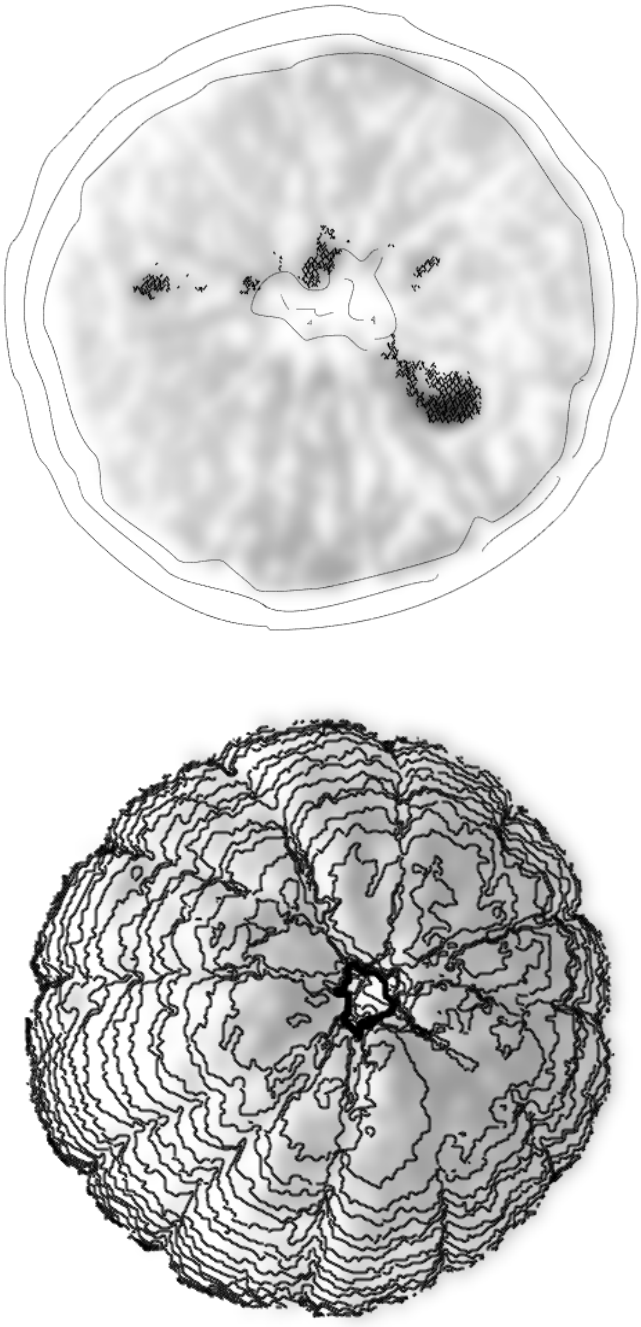


Figure 7.8: Orange with Various NPR Techniques

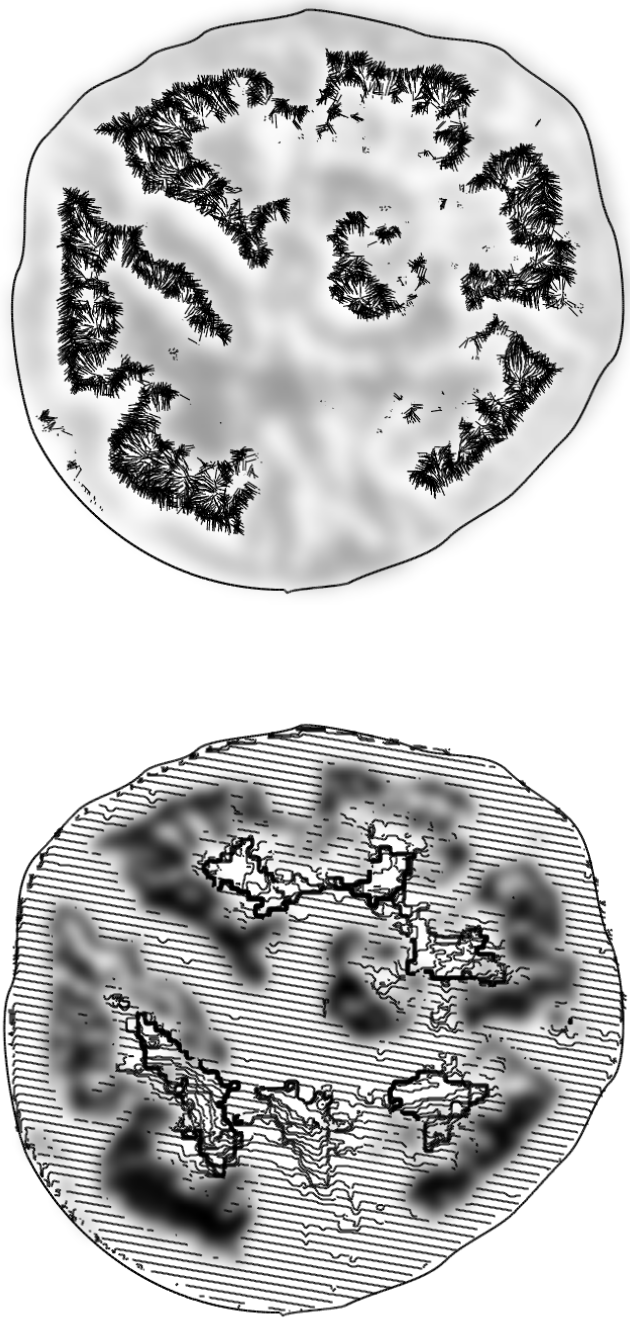


Figure 7.9: Tomato with Various NPR Techniques

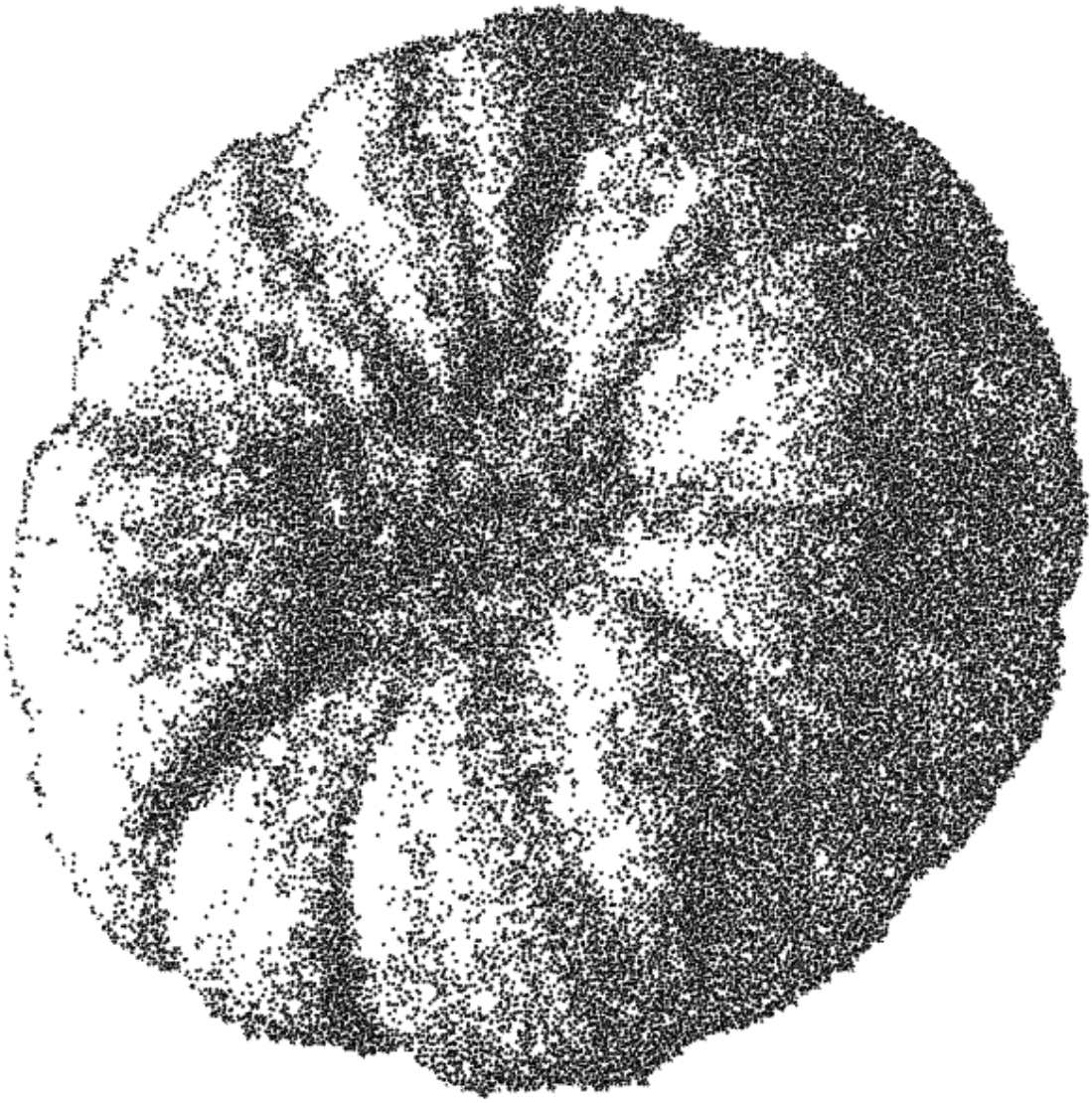


Figure 7.10: Pumpkin Stipple Simulation

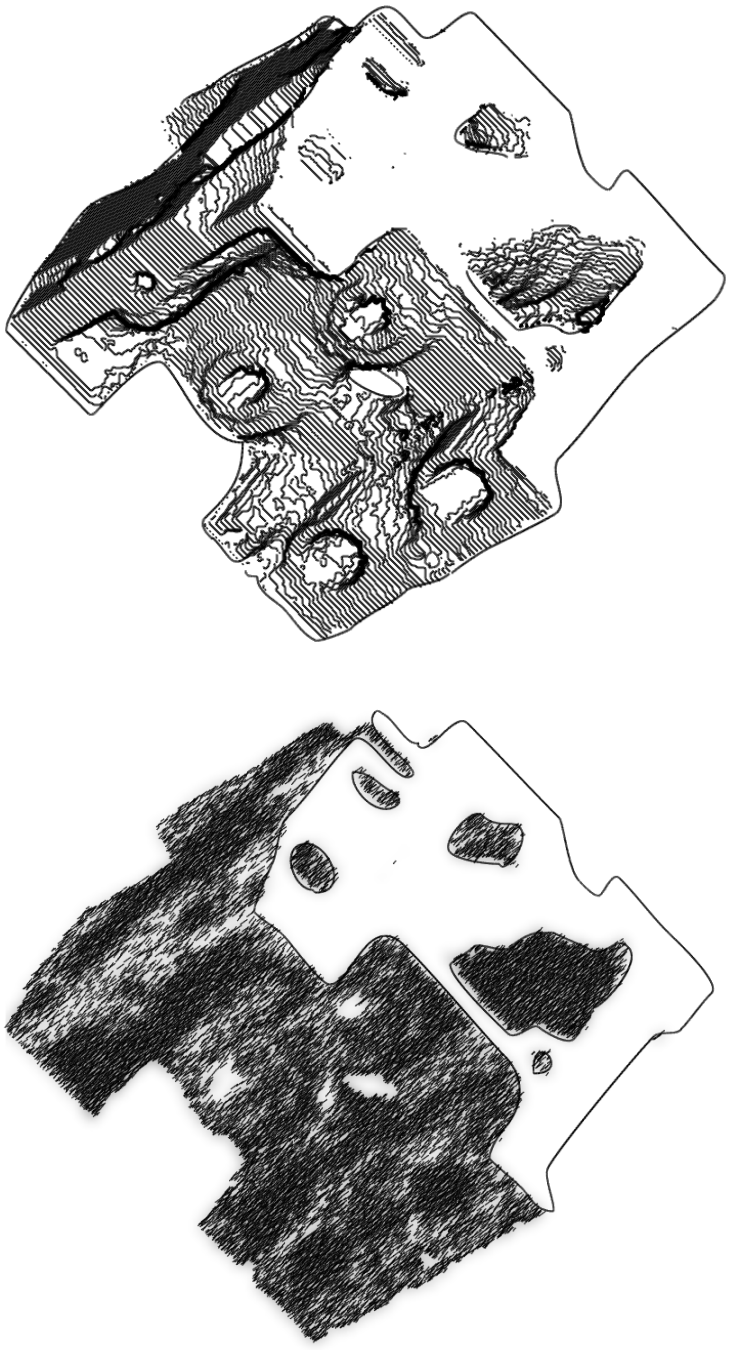


Figure 7.11: Engine Block with Various NPR Techniques

Figure 7.12: Head in a Loose Short Hatch Technique

Figures 7.7-7.12 demonstrate more illustrations[3] generated by SIV, and elicit a few short observations. The first is that grey food does not appear particularly appetising; colour should be included in a future version of SIV. The polar copper plate lines of Figure 7.8 (bottom) double as texture, suggesting a sense of abstract "orangeness". The copper plate lines in the bottom tomato image (Figure 7.9) convey structure between the seeded areas that is not visible in the top image. The varying-orientation short hatch lines of Figure 7.7 bear a strong resemblance to iron filings since they are all the same length. They convey a teddy bear's fur quite well, but a future version of SIV might foreshorten line segments based on surface normals.

Sharing and discussing the above images with colleagues in both computer graphics and fine arts, nearly every person had a different personal favourite and a different reason for their choice. In particular, the interest shown by professional artists indicates that some images do communicate with viewers on an aesthetic level. This realisation of artistic value is a happy side-benefit of SIV. Before undertaking the implementation, it was not necessarily expected that the generated images would enjoy aesthetic merit; the primary goal was simply to explore an alternative visualisation technique. The most successful illustrations (and this likely applies to all scientific visualisation) are interesting and compelling to look at, which is independent of their ability to efficiently summarise the relevant scientific data. Renderings that provide the required information with the utmost clarity and simultaneously delight the eye through the harmony of their design [Arn69, p. 123] should be the ultimate goal of scientific visualisation.

Effective illustrations do exist in the space of possible images that SIV can generate. Exposing a large number of parameters in the user interface increases the absolute number of "good" illustrations that are achievable. The preceding paragraphs list a number of factors that influence the relative effectiveness of the generated images. These are simply observations and suggestions, rather like "rules of thumb". They are neither necessary nor sufficient conditions for the pro-

[3]The orange, tomato, and pumpkin datasets are provided courtesy of the United States Department of Energy, Lawrence Berkeley National Laboratory.

duction of successful images. User experience shows that "good" images are fairly sparse in the vast multidimensional space of possible images; it can sometimes take extensive experimentation to discover such images. The next logical goal is to develop improved navigational capabilities to help the artist/user converge on effective illustrations more quickly and efficiently. Since illustrations lack a suitable objective function with which to evaluate their level of success, an organisational approach such as Design Galleries [MAB+97] shows great promise.

Chapter 8

Conclusions

The rapid proliferation of 3D volume data is prompting a search within computer graphics for more effective volume visualisation methods. This thesis presents a novel volume visualisation technique inspired by scientific illustration. Scientific illustration, with its established principles of non-photorealism, is the preferred method for visual communication of biological and medical information. Since these are the very subjects of most MRI and CT scans, scientific illustration and 3D volume data are a natural fit.

To demonstrate the feasibility and practicality of harnessing 3D volume data for scientific illustration, the prototype system described in Chapters 5 and 6 was developed. The focus was on automating the tedious details of illustration, such as exact stroke placement, while still allowing high-level control of parameters influencing the look and feel of the illustration. Some resulting images from this new method of volume visualisation are shown in Chapter 7.

Given enough time and talent, a human artist could easily replicate the illustrations generated by SIV. However, a notable advantage of this level of automation is the relative speed with which illustrations can be generated. A skilled artist often requires many days to develop a detailed illustration using traditional media. The quick turnover times enabled by SIV promote rapid experimentation and exploration of different image parameters impossible in traditional

practice. This interactive exploration of parameter space also provides a great benefit for volume visualisation, especially for 3D datasets of unknown structure.

Appropriate use of abstraction is paramount for comprehensible illustrations; without it, the viewer is overwhelmed with masses of unnecessary information. The sheer size of typical 3D datasets makes this an obvious risk in 3D volume visualisation. All previous approaches to abstraction in non-photorealistic volume rendering exhibit the shortcomings described in Section 3.4. SIV's approach to abstraction is much more explicit, using segmentation and generalised interest labels. Ebert and Rheingans's so-called "volume illustration" (Section 3.3.1) is the previous work most closely related to SIV, but it still conveys the entire voxel distribution throughout a volume. Their work certainly has separate advantages for volume visualisation, but we suggest that the explicit abstraction enabled by SIV renders it more deserving of the term "volume illustration".

Volume segmentation is typically used to create an analytical material representation suitable for additional processing. SIV is the first application of 3D volume segmentation specifically for non-photorealistic purposes. Its algorithm assumes no previous knowledge of a particular dataset, and takes advantage of the latest consumer-grade graphics hardware for interactive frame rates.

The obvious approach to the generation of scientific illustrations from 3D volumes is to extract a polygonal isosurface (ignoring the usability issues outlined in Section 3.1.3) and use pre-existing NPR algorithms that operate on polygonal meshes. Instead, SIV adapts polygonal NPR techniques to the unique challenges and opportunities presented by volume data. Not only does this eliminate an unnecessary and computationally expensive intermediate step, the resulting renderings enjoy some unique advantages. Working directly from the volume representation means that the noise inherent in real-world measurements is reflected more intuitively in the final renderings. Slight waviness in lines is highly prized in non-photorealistic rendering since it appears less artificial, and the noise inherent in the 3D volume data provides it here with no additional effort.

Due to the novelty of our volume visualisation technique, the resulting images are qualitatively unique compared to those typically seen in computer graphics. Lines that are organic in nature are definitely well suited to biological subjects. However, to realise an overall artistic vision, talented artists/users are indispensable. SIV is intended to complement the talents of human artists, not replace them.

8.1 Future Work

Suggestions for improvements and future work are sprinkled throughout Chapters 4-7 where they naturally occur. Obvious short-term incremental improvements include perspective projection, atmospheric projection, volume self-shadowing, and additional line styles such as stippled lines. Support for segment transparency and colour will be more challenging.

The exact NPR algorithms can also use extra consideration. The two NPR techniques of long copper plate lines and short hatch lines should be generalised into one; the current separation, done for technical reasons, seems somewhat artificial. Long hatching lines will benefit from cross-hatching as well. Instead of (or in addition to) orienting strokes by surface normal, strokes might be oriented according to the principal direction of curvature, following Interrante [Int97]. The addition of more NPR techniques will widen the range of possible illustrations.

Examining output formats more closely is also advisable. Postscript output for line segments will provide an analytical representation that can be more easily scaled to different resolutions. Also, illustrations should be viewed on higher resolution CRT displays to verify that the expected qualitative improvement is actually achieved; unfortunately such monitors are currently very expensive.

Revisiting the interest label concepts might also prove worthwhile. In particular, allowing subsegment interest label changes will enable an entire new set of illustrations. Developing a

clear, understandable user interface is the main challenge to overcome. Perhaps the first attempt should permit subsegment interest label specification in 2D, following Winkenbach [WS94].

SIV currently uses small, relatively simple datasets for testing. Chapter 7's illustrations all have a maximum of three identified segments, which, being relatively few in number, are easy to organise. A user will likely identify many more segments in more complicated datasets, requiring a more scalable framework for organisation. A flexible hierarchy can logically organise segments within segments. A hierarchy will parallel various biological classification schemes, appropriate since volume models typically represent biological subjects. Such a tree hierarchy was actually implemented in SIV but never used extensively. This idea should be revisited in the future when testing SIV with larger, more complex datasets.

Software performance is sufficient for the proof-of-concept NPR application, but additional acceleration for NPR algorithms (notably copper plate lines) will be very welcome. Current graphics hardware and the OpenGL programming interface have progressed to such a level that much of the NPR functionality can be relegated to graphics hardware. The realised performance gains will finally enable interative rotation of the viewpoint in the NPR view. This flexibility will open an entire new realm of possible visualisation applications.

Bibliography

[AB94] Rolf Adams and Leanne Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, Jun 1994.

[ACS84] James E. Archer, Richard Conway, and Fred B. Schneider. User recovery and reversal in interactive systems. *ACM Transactions on Programming Languages and Systems*, 6(1):1–19, Jan 1984.

[Arn69] Rudolph Arnheim. *Art and Visual Perception: A Psychology of the Creative Eye*. University of California Press, Berkeley and Los Angeles, California, 1969.

[BS00] J. Buchanan and M. Sousa. The edge-buffer: A data structure for easy silhouette rendering. In *NPAR 2000: First International Symposium on Non-Photorealistic Animation and Rendering*, pages 39–42, 2000.

[Can86] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[CG01] Balázs Csébfalvi and Eduard Gröller. Interactive volume rendering based on a "bubble model". In *Proceedings of Graphics Interface 2001*, pages 209–216, June 2001.

[CMH+01] Balázs Csébfalvi, Lukas Mroz, Helwig Hauser, Andreas König, and Eduard Gröller. Fast visualization of object contours by non-photorealistic volume rendering. In *Proceedings of EUROGRAPHICS 2001*, 2001.

[CMS88] Michael Chen, S. Joy Mountford, and Abigail Sellen. A study in interactive 3-D rotation using 2-D control devices. In *SIGGRAPH 1988 Conference Proceedings*, pages 121–129, 1988.

[Coo86] Robert L. Cook. Stochastic sampling in computer graphics. *Transactions on Graphics*, 5(1):51–72, Jan 1986.

[DG01] Lijun Ding and Ardeshir Goshtasby. On the Canny edge detector. *Pattern Recognition*, 34(3):721–725, Mar 2001.

[Elv92] T. Todd Elvins. A survey of algorithms for volume visualization. *Computer Graphics*, 26(3):194–201, 1992.

[ER00] David Ebert and Penny Rheingans. Volume illustration: Non-photorealistic rendering of volume models. In *Proceedings of IEEE Visualization 2000*, pages 195–202, 2000.

[EW95] Shimon Edelman and Yair Weiss. Vision, hyperacuity. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, Cambridge, Massachusetts, 1995. MIT Press.

[Far88] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, Inc., San Diego, California, 1988.

[Fri98] Sarah F. Frisken. Using distance maps for accurate surface reconstruction in sampled volumes. *ACM/IEEE Symposium on Volume Visualization*, pages 23–30, 1998.

[FvDFH96] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley Publishing Company, Reading, Massachusetts, second edition, 1996.

[GIHL00] Ahna Girshick, Victoria Interrante, Steven Haker, and Todd Lemoine. Line direction matters: An argument for the use of principal directions in 3D line drawings. In *NPAR 2000: First International Symposium on Non-Photorealistic Animation and Rendering*, pages 43–52, 2000.

[GK96] Allen Van Gelder and Kwansik Kim. Direct volume rendering with shading via three-dimensional textures. *ACM/IEEE Symposium on Volume Visualization*, pages 23–30, Oct 1996.

[Gre99] Stuart Green. Introduction to non-photorealistic rendering. SIGGRAPH 1999 Non-Photorealistic Rendering Course Notes, Aug 1999.

[GSG+99] Bruce Gooch, Peter-Pike J. Sloan, Amy Gooch, Peter Shirley, and Richard F. Riesenfeld. Interactive technical illustration. In *Symposium on Interactive 3D Graphics*, pages 31–38, 1999.

[Gup61] Arthur L. Guptill. *Drawing with Pen and Ink*. Reinhold Publishing Corporation, New York, New York, 1961.

[GV97] Jonas Gomes and Luiz Velho. *Image Processing for Computer Graphics*. Springer-Verlag, New York, New York, 1997.

[GW77] Rafael C. Gonzalez and Paul Wintz. *Digital Image Processing*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1977.

[HB97a] Ray Hashemi and William G. Bradley. *MRI: The Basics*. Lippincott Williams and Wilkins, Philadelphia, Pennsylvania, 1997.

[HB97b] Donald Hearn and M. Pauline Baker. *Computer Graphics*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, second edition, 1997.

[Her99] Aaron Hertzmann. Introduction to 3D non-photorealistic rendering: Silhouettes and outlines. SIGGRAPH 1999 Non-Photorealistic Rendering Course Notes, Aug 1999.

[HKR93] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8):850–863, Sep 1993.

[HLW93] S. C. Hu, I. H. H. Lee, and N. E. Wiseman. Skeletal strokes. In *ACM Symposium on User Interface and Technology*, pages 197–206, 1993.

[Hod89] Elaine R. S. Hodges, editor. *The Guild Handbook of Scientific Illustration*. Van Nostrand Reinhold, New York, New York, 1989.

[HSSB98] Mike Heath, Sudeep Sarkar, Thomas Sanocki, and Kevin Bowyer. Comparison of edge detectors: A methodology and initial study. *Computer Vision and Image Understanding*, 69(1):38–54, Jan 1998.

[Int97] Victoria Interrante. Illustrating surface shape in volume data via principal direction-driven 3D line integral convolution. In *SIGGRAPH 1997 Conference Proceedings*, pages 109–116, 1997.

[KH84] James T. Kajiya and Brian P. Von Herzen. Ray tracing volume densities. In *SIGGRAPH 1984 Conference Proceedings*, pages 165–174, 1984.

[KSG99] András Kelemen, Gábor Szekely, and Guido Gerig. Elastic model-based segmentation of 3-D neuroradiological data sets. *IEEE Transactions on Medical Imaging*, 18(10):828–839, Oct 1999.

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

[Lei94] Wolfgang L. Leister. Computer generated copper plates. *Computer Graphics Forum*, 13(1):69–77, 1994.

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

[LM02] Eric B. Lum and Kwan-Liu Ma. Hardware-accelerated parallel non-photorealistic volume rendering. In *NPAR 2002: Second International Symposium on Non-Photorealistic Animation and Rendering*, pages 67–74, 2002.

[LV00] Tom Lokovic and Eric Veach. Deep shadow maps. In *SIGGRAPH 2000 Conference Proceedings*, pages 385–392, 2000.

[MAB+97] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: A general approach to setting parameters for computer graphics and animation. In *SIGGRAPH 1997 Conference Proceedings*, pages 389–400, 1997.

[McC99] Michael D. McCool. Anisotropic diffusion for Monte Carlo noise reduction. *ACM Transactions on Graphics*, 18(2):171–194, Apr 1999.

[Mei96] Barbara J. Meier. Painterly rendering for animation. In *SIGGRAPH 1996 Conference Proceedings*, pages 477–484, 1996.

[MSV95] R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.

[NCKG00] László Neumann, Balázs Csébfalvi, Andreas König, and Eduard Gröller. Gradient estimation in volume data using 4D linear regression. In *Proceedings of EUROGRAPHICS 2000*, 2000.

[NG01] Aljaz Noe and James C. Gee. Partial volume segmentation of cerebral MRI scans with mixture model clustering. *Information Processing in Medical Imaging (IPMI)*, pages 423–430, 2001.

[NVI01] NVIDIA Corporation. NVIDIA Introduces GeForce3—Breaks New Ground in the Quest for Real-Time Cinematic Graphics. Press release, Feb 2001.

[PHWF01] Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein. Real-time hatching. In *SIGGRAPH 2001 Conference Proceedings*, pages 581–586, 2001.

[Pit57] Henry C. Pitz. *Ink Drawing Techniques*. Watson-Guptill Publications, New York, New York, 1957.

[PL90] Theo Pavlidis and Yuh-Tay Liow. Integrating region growing and edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):225–233, Mar 1990.

[Sau99] Eric Saund. Perceptual organization of occluding contours generated by opaque surfaces. In *Proceedings of the 1999 Conference on Computer Vision and Pattern Recognition*, pages 624–630, 1999.

[Sch90] Philip J. Schneider. An algorithm for automatically fitting digitized curves. In Andrew Glassner, editor, *Graphics Gems I*, pages 612–626, Cambridge, Massachusetts, 1990. Academic Press, Inc.

[SF91] D. D. Seligmann and S. Feiner. Automated generation of intent-based 3D illustrations. In *SIGGRAPH 1991 Conference Proceedings*, pages 123–132, 1991.

[Sil01] Silicon Graphics, Inc. SGI Delivers Powerful Software Tools for High-Performance Visualization. Press release, Aug 2001.

[SML98] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, second edition, 1998.

[SRS97] Stefan Schlechtweg, Andreas Raab, and Thomas Strothotte. Creating online graphical illustrations for medical texts. In *Multimedia Technology in Medical Training: Paper and Poster Proceedings of the European Workshop*, pages 53–56, 1997.

[SSWC88] P. K. Sahoo, S. Soltani, A. K. C. Wong, and Y. C. Chen. A survey of thresholding techniques. *Computer Vision, Graphics and Image Processing*, 41:233–260, 1988.

[ST90] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3D shapes. *Computer Graphics*, 24(4):197–206, Aug 1990.

[Str86] Steve Strassmann. Hairy brushes. In *SIGGRAPH 1986 Conference Proceedings*, pages 225–232, 1986.

[Str98] Thomas Strothotte, editor. *Computational Visualization: Graphics, Abstraction, and Interactivity*. Springer-Verlag, Berlin, Germany, 1998.

[TC00] S. M. F. Treavett and M. Chen. Pen-and-ink rendering in volume visualisation. In *Proceedings of IEEE Visualization 2000*, pages 203–210, 2000.

[TCSJ01] S. M. F. Treavett, M. Chen, R. Satherley, and M. W. Jones. Volumes of expression: Artistic modelling and rendering of volume datasets. In *Proceedings of Computer Graphics International 2001*, pages 99–106, 2001.

[TO99] Greg Turk and James O'Brien. Shape transformation using variational implicit functions. In *SIGGRAPH 1999 Conference Proceedings*, pages 335–342, 1999.

[TSW97] Patrick C. Teo, Guillermo Sapiro, and Brian A. Wandell. Creating connected representations of cortical gray matter for functional MRI visualization. *IEEE Transactions on Medical Imaging*, 16(6):852–863, Dec 1997.

[WE98] R. Westermann and T. Ertl. Efficiently using graphics hardware in volume rendering applications. In *SIGGRAPH 1998 Conference Proceedings*, pages 169–179, 1998.

[WGW94] Orion Wilson, Allen Van Gelder, and Jane Wilhelms. Direct volume rendering via 3D textures. Technical Report UCSC-CRL-94-19, University of California, Santa Cruz, 1994.

[WNDS99] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner. *OpenGL Programming Guide*. Addison-Wesley Publishing Company, Reading, Massachusetts, third edition, 1999.

[Wor96] World Wide Web Consortium. PNG (Portable Network Graphics) Specification. Format specification, Oct 1996.

[WS94] Georges Winkenbach and David H. Salesin. Computer-generated pen-and-ink illustration. In *SIGGRAPH 1994 Conference Proceedings*, pages 91–100, 1994.

[WWG+99] Simon K. Warfield, Carl-Fredrik Westin, C. R. G. Guttmann, M. Albert, Ferenc A. Jolesz, and Ron Kikinis. Fractional segmentation of white matter. *MICCAI 99: Second International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 62–71, Sep 1999.