Physically-based rendering and algebraic manipulation of volume models.

Abdul Rahman, Al-Fathiatul Habibah

Physically-based Rendering and Algebraic Manipulation of Volume Models

Al-Fathiatul Habibah Abdul Rahman BSc. (Wales)

Submitted to the University of Wales in fulfilment of
the requirements for the Degree of Doctor of Philosophy

Department of Computer Science
Swansea University

September 2006

Summary

Algebraic properties of operations on volume objects may be discovered and reasoned about independently of any specific volume object. In this thesis, the main objective of the study of the algebraic properties of constructive operations in volume graphics is to design more consistent and useful operators and to discover the laws governing these operators. This objective is important as it helps maintain the predictability and consistency in the results of performing a series of operations on volume objects.

This thesis is concerned with the physically-based rendering and algebraic manipulation of volume models. The work that has been undertaken in the thesis includes:

- the development of a novel approach to direct volume rendering based on the Kubelka-Munk theory of diffuse reflectance, which exhibits better algebraic properties in comparison with the conventional volume rendering based on the RGBα model;
- the design of a new set of operations for an algebraic framework of volume-based modelling and the consideration of a more complex example class of volume model that includes physical properties such as absorption and scattering;
- the implementation of an alternative form of colour mixing algorithm that utilises the physical properties of volume objects;
- the design of an algebraic framework for modelling complex deformation in a constructive manner;
- the consideration and implementation of newly improved colour-to-spectrum conversion algorithms.

The work undertaken in the thesis falls neatly under the algebraic framework for modelling complex spatial objects.

The physically-based rendering facilitates a correct spectral volume rendering integral that is suitable for both solid objects and amorphous matter in volume data sets. It can be seen that the physically-based rendering also provides volume visualisation with more accurate optical effects than the traditional volume rendering integral based on RGBα accumulation. The algebraic framework for modelling complex deformation confirms the theoretic feasibility as well as providing a practical implementation for integrating complex volume deformation into volume scene graphs.

Some parts of this thesis have been presented at Eurographics 2005, Dublin, Ireland.

Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed (candidate)

Date 2/4/07

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed (candidate)

Date 2/4/07

Statement 2

I hereby give my consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed (candidate)

Date 2/4/07

Acknowledgements

It is hard to believe that it was seven years ago that I arrived in Swansea. It is even harder to imagine that after three years of undergraduate study, I have somehow managed to pluck enough courage to pursue a PhD and now have approached the end of my PhD. All this would not have been possible without the support and encouragement of my family, friends and colleagues.

I would like to give my deepest thanks and gratitude to my mother for all the great sacrifices that she has made without which it would have been impossible for me to achieve so much in life, and to my sister who always supports and respects the decisions that I have made.

I would like to give my most heart-felt "Thank You" to my supervisor, Prof. Min Chen, for his great patience, confidence in my capabilities and many invaluable ideas throughout my PhD, without which this thesis would not be what it is.

I would also like to express my thanks and gratitude to my proof readers, Will Harwood, Simon Walton and Jon Harvey, for their time and patience in reading and correcting this thesis. The final version of this thesis is solely the author's responsibility.

Last but not least, to all the staff in the department, friends and colleagues who have helped and assisted me in one way or another in the completion of this thesis.

This work was supported by a PhD studentship from Engineering and Physical Sciences Research Council (EPSRC), Overseas Research Student Awards Scheme (ORS) and International Office Postgraduate Scholarship, University of Wales Swansea.

Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Aims and Objectives 3

1.3 Thesis Outline 4

1.3.1 Chapter 2: Volume Rendering and Visualisation 4

1.3.2 Chapter 3: Colour Models in Computer Graphics 4

1.3.3 Chapter 4: Spectral Volume Rendering 5

1.3.4 Chapter 5: Algebraic Properties of Volume-based Modelling and Colour Mixing 5

1.3.5 Chapter 6: Algebra of Spatial Displacement Function 5

1.3.6 Chapter 7: Colour-Spectrum Conversions 6

1.3.7 Chapter 8: Conclusions 6

1.4 Nomenclature 6

1.5 Data Sets 7

2 Volume Rendering and Visualisation **8**

2.1 Introduction 8

2.2 Data Acquisition Techniques 9

2.2.1 Volumetric Data Types 9

2.2.2 Computed Tomography 11

2.2.3 Magnetic Resonance Imaging 12

2.2.4 Ultrasound 13

2.2.5 Computer Simulation 14

2.3 Surface Extraction 14

2.3.1 Marching Cubes 15

2.3.2 Topological Correctness and Robustness 16

2.3.3 Marching Tetrahedra 18

2.4 Modelling Schemes 18

2.4.1 Solid Object Modelling 18

2.4.2 Non-Solid Object Modelling 22

2.5 Constructive Representations of Volume Objects 24

2.5.1 Volumetric CSG 25

2.5.2 Constructive Volume Geometry 25

2.5.3 Constructive Hypervolume Modelling 26

2.5.4 Volume Scene Graphs 27

2.6 Volume Rendering 27

2.6.1 The Volume Rendering Integral 28

2.6.2 Ray Casting 28

2.6.3 MIP Rendering 29

2.6.4 Splatting 29

2.6.5 Spectral Volume Rendering 29

2.7 Optical Models 30

2.7.1 Absorption, Emission and Combined 30

2.7.2 Single Scattering and Shading 31

2.7.3 Shadows 31

2.7.4 Multiple Scattering 32

2.7.5 Refraction 32

2.8 Volume Deformation and Animation 33

2.8.1 Volume Deformation 33

2.8.2 Volume Animation 34

3 Colour Models in Computer Graphics 36

3.1 Introduction 36

3.2 Background on Colour 37

3.3 Concepts 38

3.3.1 Light Interactions with Matter 40

3.4 Human Visual System 41

3.5 CIE System 42

3.6 Colour Models 43

3.7 Colour Mixing 45

3.8 Colour Constancy 46

3.9 Colour-Spectrum Transformations 47

3.9.1 RGB-Spectrum Transformation 47

3.10 Spectral Representation Models 48

4 Spectral Volume Rendering 51

4.1 Introduction 51

4.2 Related Work 52

4.3 The Kubelka-Munk Theory 53

4.4 Spectral Volume Modelling 56

4.5 Spectral Volume Rendering 57

4.6 Sampling Correctness 58

4.7 Visual Effects and Correctness 64

4.8 Post-illumination and Spectral Transfer Functions 68

4.9 iPi – Interactive Post-illumination 78

4.9.1 iPi: Design Objectives 78

4.9.2 iPi: Interface Design 79

4.9.3 iPi: Rendering Attributes 80

4.9.4 iPi: Spectral Transfer Function 81

4.9.5 iPi: Interrogative Light Source 82

4.9.6 iPi: File Structure and Data Management 84

4.9.7 iPi: System Constraints 85

4.10 Summary 86

5 Algebraic Properties of Volume-based Modelling and Colour Mixing 87

5.1 Introduction 87

5.2 Background on Constructive Volume Geometry 88

5.3 Opacity Channel Model 89

5.4 4-Colour Channel Model 90

5.4.1 Union and Intersection Operations 91

5.4.2 **SELECT** Operation 93

5.4.3 **SELECT** Operation Based on Luminance 94

5.4.4 **SELECT** Operation Based on **MAX** 96

5.4.5 **SELECT** Operation Based on **MIN** 97

5.4.6 Comparison of Results 98

5.5 The CVG Framework for the Kubelka-Munk Model 98

5.6 Colour Mixing Laws 102

5.6.1 Grassmann's Laws of Colour Mixing 104

5.6.2 Bouguer-Beer's Law of Colour Mixing 108

5.6.3 Comparisons on the Colour Mixing Laws 109
5.7 Summary 110

6 Algebra of Spatial Displacement Functions **111**
6.1 Introduction 111
6.2 Related Work 112
6.2.1 Volume Deformation 112
6.2.2 Displacement Mapping 113
6.3 Definitions and Concepts 113
6.3.1 Definitions 114
6.3.2 Regularisation 115
6.3.3 Relative Merits 116
6.4 Combinational SDFs and STFs 117
6.4.1 Algebraic Properties of Combinational SDFs and STFs 124
6.5 Hierarchical SDFs and STFs 124
6.6 Identity Problem 127
6.7 Inverse Problem 130
6.8 Direct Rendering Volume Scene Graphs 133
6.8.1 With Spatial Displacement Functions 133
6.8.2 With Spatial Transfer Functions 136
6.9 Summary 140

7 Colour-Spectrum Conversions **141**
7.1 Introduction 141
7.2 Metamerism 142
7.3 Related Work 143
7.3.1 Fourier Basis 145
7.3.2 Gaussian Basis 147
7.4 Reduction of the Negative Energy through Negative Minimisation 148
7.5 Reduction of the Negative Energy through Shifting Fourier Basis 149
7.6 Evaluation of Results 151
7.6.1 Spectra Generated from Reflectance Data Sets 151
7.6.2 Spectra Reconstructed from RGB coordinates 158
7.7 Summary 160

8 Conclusions **162**

8.1 Summary of contributions 163
8.1.1 A New Approach to Direct Volume Rendering 163
8.1.2 Extensive Study on Algebraic Properties of Volume-based Modelling and Physically-based Colour Mixing Techniques 164
8.1.3 Algebraic Properties of Volume Deformation 164
8.1.4 Improvements on Colour-to-Spectrum Algorithms 165
8.2 Future Work 165

Bibliography **166**

List of Figures **187**

List of Tables **192**

Chapter 1

Introduction

Contents

1.1 Motivation 1

1.2 Aims and Objectives 3

1.3 Thesis Outline 4

1.4 Nomenclature 6

1.5 Data Sets 7

There are many advantages in working with the algebraic properties of operations on volume objects. By studying the algebraic properties of constructive operations in volume objects we are able to design more consistent and useful operators and determine the laws governing them. The laws of the operators are important as they aid in the reasoning about the operators, which can lead to greater consistency and predictability in the results of performing a series of operations on volume objects.

The algebraic properties of operations on volume objects can be formulated to define a mathematical framework for volume data types and models. Such a framework can be seen in the algebraic framework for modelling complex spatial objects using combinational operations, which was introduced in 1998 [CT98]. This algebraic framework is a source of inspiration for this thesis as it presents us not only with a precise and structured manner of defining spatial objects and their composition but also a high-level mathematically-based modelling scheme for volume graphics.

1.1 Motivation

This thesis covers a number of different areas of computer graphics and visualisation. The main study is concerned with physically-based rendering and algebraic manipulation of volume models, with the focus on optical realism and volume deformation through the approach of algebraic properties for volume data types. Other aspects of computer graphics and visualisation that support this thesis are colour mixing laws and colour conversion algorithms.

Colour realism plays an important role in computer graphics and visualisation. The traditional volume rendering integral based on Bouguer's or Lambert's law (1727, 1760), deals with the logarithmic transformation of light intensity. Typically, Lambert-Bouguer's law is deployed to model the absorption of light. The usage of such a model can be a limitation as it does not provide us with great accuracy for optical realism. Recently a new rendering technique, called *spectral volume rendering*, has been introduced as a potential solution [NvS00]. The current spectral volume rendering integrals are, however, unable to provide the level of optical realism that is required as they are formed under the basis of the Lambert-Bouguer's law. It is therefore necessary to present a new approach that provides us with a greater accuracy of optical effects through the control of physical properties, such as absorption and scattering. The optical realism that is achieved by this approach can be demonstrated with a combination of several natural and artificial colour spectral data sets.

In 1998, an algebraic framework for the specification, representation, manipulation and rendering of volume data types was introduced [CT98]. One of the advantages of this algebraic framework is that its operations are not only limited to geometrical compositions but can also be extended to include the combination of the physical properties of the objects. Such a capability provides the author with an opportunity to extend the algebraic framework to include more complex models that hold a variety of physical properties and the definitions of an extensive set of operations based on the physical properties of the models. In using the algebraic framework, further understanding of the relationship between algebraic properties and their underlying physical models can be made.

The RGB (Red Green Blue) space is perhaps the most commonly implemented colour representation in rendering software and graphics platforms. However, the usage of the RGB representation presents us with several limitations [Hal89]. In particular, rendering physical optics phenomena, which requires the usage of spectral calculations [Pee93, JF99]. These limitations have motivated the author to investigate alternative forms of colour representation such as *colour spectral representation.* The usage of the colour spectral representation, however, requires some knowledge of colour science and colour-vision models. Therefore, a colour-spectrum conversion algorithm is required which can be implemented at the back-end of a graphics system while still allowing the system to maintain its RGB representation at the front-end. The colour spectral representation has also provided the author with the means to consider colour mixing techniques that take into account the physical properties of the object.

Recently, there has been demand to specify and render deformation in volume visualisation. Previous advances in the areas of volume modelling, rendering and animation have provided this need with a collection of techniques. However, existing research in the area of volume deformation is mainly experimental and often application-specific. It is therefore necessary to formulate a set of generic concepts and algorithms that unifies the various techniques for volume visualisation.

We are partially motivated by previous work on displacement mapping for modelling surface detail. However, the application of displacement mapping in volume modelling and rendering is uncommon and its implementation in both surface and volume domain is restricted to a small scale distortion. Therefore, we find that it would be an interesting proposition to incorporate such a concept into the volume domain for modelling and rendering complex deformations.

Each of the aspects motivating this thesis falls neatly under the algebraic framework for modelling complex spatial objects. Our motivation begins with the investigation into the algebraic properties of operations in the common graphics models. Such investigation has highlighted some of the limitations that can be found in graphics models based upon the RGB space and has provided us with an opportunity to explore a wider area of complex models that hold a variety of physical properties. The introduction of more complex models holding a number of physical properties can be seen as an advantage as they can provide us with a greater accuracy of optical effects and realism. The final motivation of this thesis is formed from the desire to design an algebraic framework for modelling complex deformation, which fits into the algebraic framework in the structure of a generic form of a constructive operator.

1.2 Aims and Objectives

The main objectives of this work are:

1. To develop a new approach for direct volume rendering that can facilitate a correct spectral volume rendering integral suitable for both solid objects and amorphous matter in volume data sets. The purpose is to provide volume visualisation with more accurate optical effects than the traditional volume rendering integral based on RGBα accumulation.

2. To consider an approach for the construction of spectral transfer functions for post-illumination, facilitating an interactive exploration of volume data sets.

3. To conduct further studies on the algebraic framework for modelling complex spatial objects [CT98]. The aims are to perform a comprehensive analysis of the existing example classes based on their algebraic properties and to define an extensive set of operations for the example classes.

4. To consider more complex example classes of the algebraic framework for modelling complex spatial objects [CT98] where the scalar fields can be used to define physical properties, such as absorption and scattering.

5. To conduct a comprehensive study on the different types of colour mixing algorithms. The purpose is to provide the graphical or visualisation systems with a colour mixing algorithm that is able to utilise the physical properties of objects, such as absorption and scattering.

6. To design an algebraic framework that can incorporate deformation as an integral part of a volume scene graph but also facilitate the constructive specification, interoperation and direct rendering of volume deformation.

7. To examine a technique that can be used to model complex volume deformations such as opening, peeling and folding.

8. To design a method for specifying a deformation in the same manner as a field-based volume object. With the proposed method, both procedural and discrete representations can be implemented together with the application of geometric transformation and transfer functions.

9. To consider and implement further modifications and enhancements to the existing colour-to-spectrum algorithms. The purpose is to address the negative energy problem that can result from existing algorithms.

1.3 Thesis Outline

The aspects of research in this thesis are divided into seven chapters. The thesis begins with the literature reviews found in Chapters 2 and 3. Chapter 2 presents investigations and overviews of existing research work in volume rendering and visualisation, while in Chapter 3 a comprehensive review on the subject of colour models in computer graphics is presented. Chapter 4 presents a detailed discussion on the physically-based rendering. Chapter 5 presents the algebraic properties and colour mixing, while Chapter 6 examines the algebraic manipulations of volume models. In Chapter 7, a detailed examination on colour-spectrum conversions is given. Finally the concluding remarks are presented in Chapter 8. A breakdown of this thesis is given as follows.

1.3.1 Chapter 2: Volume Rendering and Visualisation

Volume graphics can be divided into several main functional components [Che01], namely data acquisition, volume modelling, volume rendering and volume deformation and animation. In this chapter, all the functional components of volume graphics are examined in detail as they form an integral part in understanding our scientific investigations. This chapter begins with a brief outline of a collection of data acquisition techniques commonly applied in volume graphics.

The chapter continues with sections on surface extraction techniques, modelling schemes, volume representations, volume rendering, optical models and volume deformation and animation. We discuss the various forms of modelling schemes that are available, and similar discussions are given on volume representations by looking into major frameworks that are implemented in the construction of volume objects. In particular, this chapter will highlight recent advancements that have be made in the field of volume graphics such as refraction rendering and spectral volume rendering.

1.3.2 Chapter 3: Colour Models in Computer Graphics

This chapter presents a comprehensive review on the subject of colour models in computer graphics. The chapter begins by giving a brief background on colour from a historical point of view. In order to understand the subject of colour models in computer graphics, the terminology and concepts that are found in colour are reviewed.

This chapter then discusses the interaction between light and matter, such as emission, scattering and absorption. Further topics that are examined in this chapter include the Human Visual System, colour standardisation and models, as well as colour mixing and constancy. The subject of colour-spectrum transformations is also discussed together with a brief overview of spectral representation models.

1.3.3 Chapter 4: Spectral Volume Rendering

This chapter presents a new approach to direct volume rendering based on the Kubelka-Munk theory of diffuse reflectance. This chapter begins by giving a brief overview of the Kubelka-Munk theory. Then, an outline of the spectral modelling scheme for representing volumetric objects and their absorption and scattering properties is given.

This chapter later proceeds by demonstrating the correctness of the volume integral based on the Kubelka-Munk theory in comparison with existing spectral volume rendering methods. This is followed by a discussion on the design of transfer functions for specifying absorption and scattering coefficients. The use of post-illumination for integrating pre-processed reflectance images in real-time is then demonstrated with a system, called iPi (*interactive post-illumination*). This chapter further demonstrates the optical realism that can be achieved by this approach with a combination of several natural and artificial colour data sets.

1.3.4 Chapter 5: Algebraic Properties of Volume-based Modelling and Colour Mixing

This chapter gives the example graphics classes that have been implemented in the algebraic framework of Constructive Volume Geometry (CVG) [CT98]. The example classes are explored through the examination of their algebraic properties and the introduction of newly defined operations. In addition, the algebraic framework of CVG is also extended to include the notion of vector fields, in order to accommodate the spatial object based on the Kubelka-Munk model of absorption and scattering.

This chapter continues with an exposition of a collection of colour mixing laws. The descriptions include implementations and a comparative study of the different colour mixing laws.

1.3.5 Chapter 6: Algebra of Spatial Displacement Function

The aim of this chapter is to provide an extensive study into the subject of volume deformation through the approach of *Spatial Displacement Functions* (SDFs). This chapter begins with the introduction of SDF as the fundamental building block in the algebraic framework for modelling complex deformation in a constructive manner. Then, the various representations that are available in SDFs are presented together with the analysis of their algebraic properties. A discussion on the algorithms for rendering SDFs is also presented.

The main focus of this chapter is to demonstrate that SDFs are functionally more powerful and algebraically more sophisticated than its predecessors.

1.3.6 Chapter 7: Colour-Spectrum Conversions

This chapter begins with a brief description of colour metamerism. This is followed by a review of the previous developments in colour-to-spectrum conversions. The chapter then proceeds to detail the existing problems that can be found in the conversion algorithms such as the negative energy problem.

The chapter then presents several methods of addressing the negative energy problem. Detailed explanations on the newly presented methods are provided. A comparative study on the newly presented method and its predecessor is also presented.

1.3.7 Chapter 8: Conclusions

This concluding chapter presents an overview of the contributions that this research work has provided to the scientific community. The chapter also provides the author's suggestions for future work.

1.4 Nomenclature

This chapter presents a collection of important mathematical symbols that will be used throughout this thesis. The collection of mathematical symbols can be seen Table 1.1.

Mathematical Term	Definition
V	Volumetric dataset
p_i	A sampling location in 3D Euclidean space
v_i	A scalar value associated with p_i
$\mathbb{R}$	Set of all real numbers
$\mathbb{N}$	Set of all natural numbers
$\mathbb{E}^3$	3D Euclidean space
S_τ	An isosurface for a designated isovalue, τ
λ	The wavelength of a colour
K	The absorption property of a material
S	The scattering property of a material
R	The reflectance of a material
T	The transmittance of a material
RGB	Red, green and blue intensity respectively

Table 1.1: Thesis nomenclature.

1.5 Data Sets

The author wishes to acknowledge the sources of volume data sets that are used throughout this thesis. The volume data sets and their sources are listed as follows:

- Engine block from General Electric, USA;
- Foot from Philips Research, Hamburg, Germany;
- Frog, orange and tomato from Lawrence Berkeley Laboratory, USA;
- Fuel Injection from SFB 382 of the German Research Council (DFG);
- MRI head from University of Erlangen, Germany;
- Sheep heart from Center for Duke University, North Carolina, USA;
- Visible Human from National Library of Medicine, USA; and
- Lobster from VolVis distribution of State University of New York, Stony Brook, USA.

The author is grateful to the Volume Library web site [Roe04] for making these data sets easily accessible.

The author also wishes to acknowledge the sources of spectral colour datasets, including those made available by Glassner, NCSU and Ward [War04].

Chapter 2

Volume Rendering and Visualisation

Contents

2.1 Introduction **8**
2.2 Data Acquisition Techniques **9**
2.3 Surface Extraction **14**
2.4 Modelling Schemes **18**
2.5 Constructive Representations of Volume Objects **24**
2.6 Volume Rendering **27**
2.7 Optical Models **30**
2.8 Volume Deformation and Animation **33**

2.1 Introduction

Surface graphics and volume graphics are the main subfields that can be found in computer graphics. Volume graphics was first outlined as a subfield of computer graphics in 1993 [KCY93] with its first workshop held in 1999 [Jon99]. Since then, the substantial progress that has been made in the field of volume graphics can be seen in four subsequent successful workshops. The majority of techniques that exist in traditional computer graphics are defined as surface-based techniques. One of the main drawbacks of surface-based techniques is their difficulty in handling amorphous phenomena, which can be modelled and rendered by volume-based techniques. With the rapid developments in computer hardware and the capabilities that volume graphics exhibits, volume-based techniques have the potential to compete with and overtake surface-based techniques in computer graphics.

This chapter presents a literature review of the existing research that has been performed in volume graphics, focusing on its main functional components [Che01]. This includes data acquisition techniques, volume modelling, volume rendering and volume deformation and animation.

The organisation of this chapter is as follows:

- In Section 2.2, a brief overview of data acquisition techniques is presented.
- In Section 2.3, the process of surface extraction on volumetric data set is discussed.
- In Section 2.4, the different types of modelling scheme that can be found in computer graphics are examined.
- In Section 2.5, the major frameworks that are utilised in the construction of volume objects are evaluated.
- In Section 2.6, a review on the subject of volume rendering is presented.
- In Section 2.7, a survey into the various types of optical models is presented.
- In Section 2.8, a review on the collection of existing techniques in volume deformation and animation is given.

2.2 Data Acquisition Techniques

Voxels, or volume elements, are the primitive elements that form a volume dataset. A volume data set is often represented through a three-dimensional (3D) array of voxels, commonly called a *volume buffer*, *cubic frame buffer* or *3D raster*. The properties of a volume data set are normally determined by its spatial and intensity resolutions. A volume dataset can be obtained from a range of techniques – this section presents a brief overview of the different volumetric data types and the numerous approaches of volume data acquisition and creation.

2.2.1 Volumetric Data Types

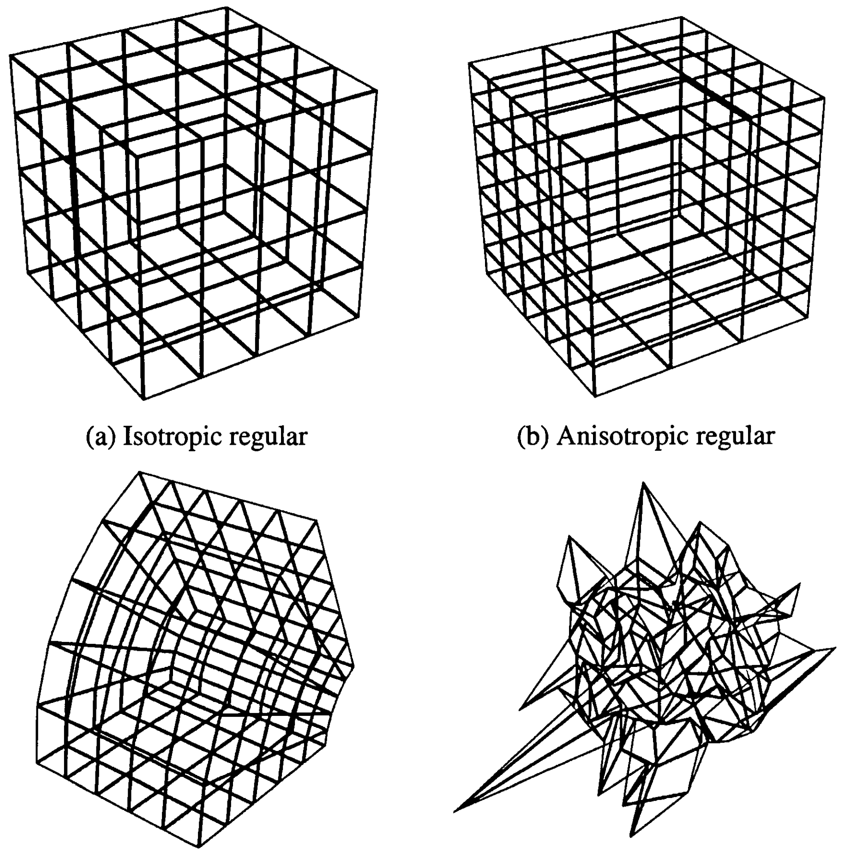
A *volumetric dataset* can be defined as:

$$\mathbf{V} = \{\, (p_i, v_i) \,|\, v_i = F(p_i),\ i = 1, 2, \ldots, n \,\}$$

where element v_i represents either a scalar value or multi-values of vector or tensor and p_i represents the sampling location in 3D Euclidean space, $\mathbb{E}^3$. The function F can be mathematically defined as [Win02]:

- a scalar function, $F : \mathbb{E}^3 \longrightarrow \mathbb{R}$;
- an n-dimensional vector function: $F^n : \mathbb{E}^3 \longrightarrow \mathbb{R}^n$; or
- a t-ranked tensor function: $F^{n^t} : \mathbb{E}^3 \longrightarrow \mathbb{R}^{n^t}$

where $n, t \in \mathbb{N}$ and are often organised as a 3D structure. Scalar and vector functions can be defined as special cases of tensor functions with ranks 0 and 1, respectively. The *topological* and *geometrical* relationships of the structure are determined by the discrete sampling of its continuous functions. The topological relationship defines the connectivities of the element v_i whereas the geometrical relationship is defined by the properties of the element v_i (such as shape and volume).



(a) Isotropic regular (b) Anisotropic regular

(c) Curvilinear (d) Unstructured grid

Figure 2.1: (a)-(c) Structured grids and (d) unstructured grid.

A structure is considered to be a ***structured grid*** when its topological relationship is well defined, and otherwise, an *unstructured grid*. The elements in a structured grid are of hexahedral shape and can be defined to be *rectilinear* or *curvilinear*. A structured rectilinear grid with elements that are of the same dimension and volume with regularly sampling intervals can be distinctly divided into *isotropic regular* for a cuboid shaped element (see Figure 2.1(a)) or *anisotropic regular* for a non-cuboid shaped element (see Figure 2.1(b)). Data sets commonly generated by medical scanners are of anisotropic regular type. Isotropic and anisotropic regular grids have both *implicit* geometry and topology which eliminates the need for *explicit* sampling positions of the elements as well as their connectivity and ordering. Such grids are now commonly used in volume-based applications due to their efficiency in storage and processing, whereas a structured curvilinear may required an explicit specification of the sampling positions and may consists of curved faces (see Figure 2.1(c)). A structured curvilinear is often used in scientific simulation such as *Computational Fluid Dynamics* (CFD) for representation of aircraft wings.

For an unstructured grid, its elements are not necessarily of hexahedral shape and can be defined to be *regular*, *irregular* or *hybrid* (see Figure 2.1(d)). In an unstructured regular

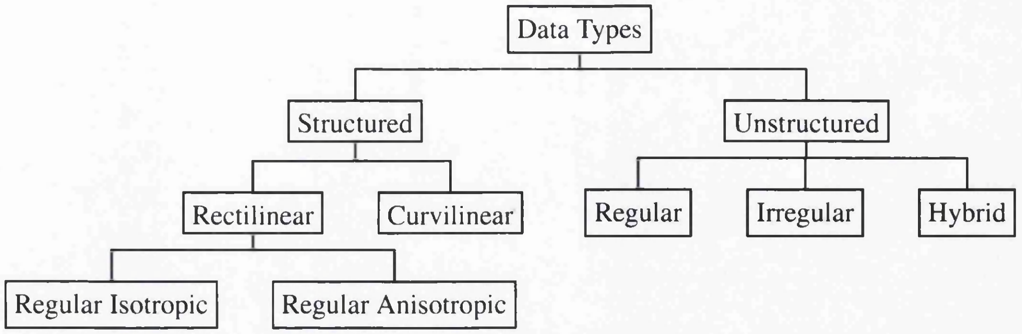


Figure 2.2: Classification of data types.

grid, each dimension and volume of its elements are the same. In an unstructured irregular grid, each element can be of different volumes and dimensions and may not be hexahedral. An unstructured hybrid is defined when all elements have a unique number of faces. An unstructured grid may require an *explicit* specification of the sampling positions or topological connectivities between the samples which can be determined through tetrahedralisation [Nie97]. Figure 2.2 demonstrates the classification of these data types.

2.2.2 Computed Tomography

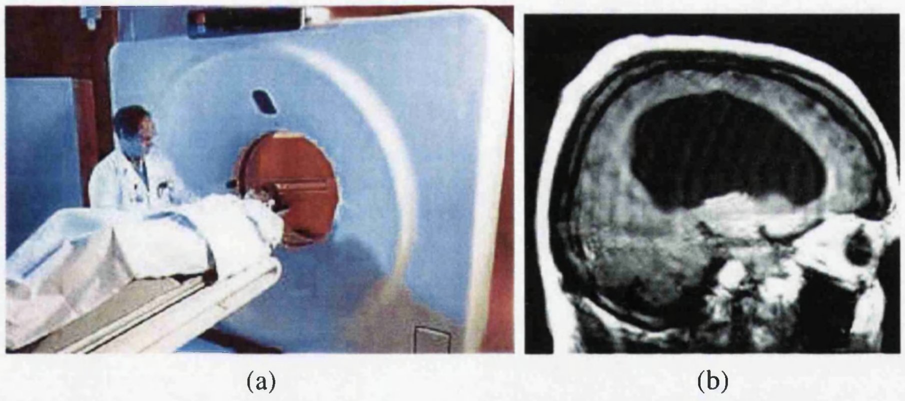


Figure 2.3: (a) A CT equipment and (b) CT scan [The05].

Computed Tomography (CT) [Web98], or *Computed Axial Tomography* (CAT), is a noninvasive medical imaging technique that involves radiation. CT was invented independently in 1972 by Hounsfield [Hou72, Hou73] and Cormack [Cor63, Cor64] for which they were both awarded a Nobel Prize. CT involves the firing of x-rays using a 360° rotating detector through a human body which results in slices (or cross-sectional images) perpendicular to the body. The alignment of these slices creates a 3D dataset. The duration for a CT scan can range from 2 to 20 minutes and an injection of contrast material can be used to emphasise blood vessels.

The intensity of radiation, $I(x)$, that can be detected is defined as [Her80]:

$$I(x) = I_0 e^{\int \mu(x)dx}$$

where I_0 is the attenuation level with absorption μ. This calculation is similar to that of the ray-casting composition and *Bouguer-Beer's law* [Bee52, Bee54]. Although CT can be used in the diagnosis of osteoporosis through the measurement of bone mineral density, a CT scan does not hold as much detail precision in comparison with an MRI scan. CT can now be used to detect symptoms of a stroke through a new method called *Perfusion CT* (PCT). Figures 2.3 (a) and (b) demonstrate a patient being prepared for a CT scan and a CT scan, respectively.

2.2.3 Magnetic Resonance Imaging

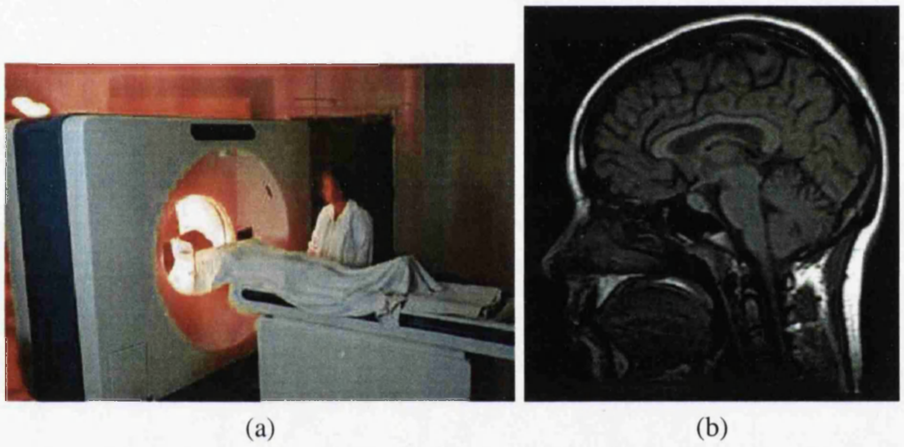


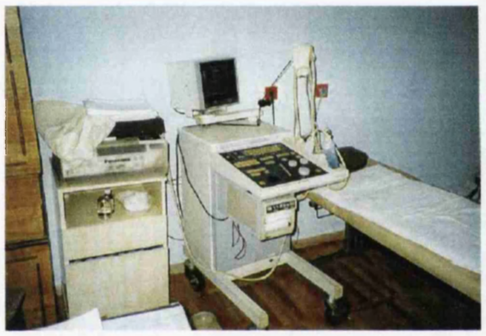
Figure 2.4: (a) An MRI equipment and (b) MRI scan [The05].

Magnetic Resonance Imaging (MRI) [Web98] is used in the medical science for producing high quality images of the human body interior. Although MRI is based on the principles of *Nuclear Magnetic Resonance* (NMR), the word *nuclear* was abandoned during its naming process due to the stigma attached to it during the late 1970s. The history of MRI began with the discovery of the magnetic resonance by Felix Bloch [BHP46] and Edward Purcell [PTP45] and since then it has produced six Nobel Prize winners in the areas of Physics, Chemistry, Physiology and Medicine. Later in 1950, NMR was developed and was used for the next 20 years in the application of chemical and physical molecular analysis. MRI is now a widely used technique for medical imaging with more that 10 000 MRI units located worldwide and more than 75 million MRI scans are being carried out each year [Hor05].

The major component in MRI is the large tube shaped or cylindrical magnet with unit measurement of *Tesla*. MRI is based on the application of phase and frequency encoding and the Fourier Transform [KWE75, EA66]. The images in MRI are created via the absorption and emission of the magnetic energy and radio waves. Clicking noises are often heard during an MRI scan as the magnetic coils are constantly being switched on and off for the measurement of the reflected MR signal out of the body. Currently, there are two types of MRI systems available, the traditional cylindrical MRI and the Open MRI system. The duration for a typical MRI scan can range from ten minutes to an hour. Patients who suffer from claustrophobia are often scanned using the Open MRI system.

MRI helps to reduce the suffering of patients as it is a noninvasive technique which produces no side effects and does not use x-ray radiation. Surgeons are also able to use MRI as a preoperative tool for locating lesions and diagnosing various types of cancer. An advantage of MRI in comparison with CT is of the different possible viewing orientation of the body such as coronal, sagittal and axial orientations. MRI is now not only being used for the examination of the brain and heart but also in the application of diagnostic mammography and treatment of sports injuries. Figures 2.4 (a) and (b) show a patient being prepared for an MRI scan and an MRI result image slice, respectively.

2.2.4 Ultrasound


(a)

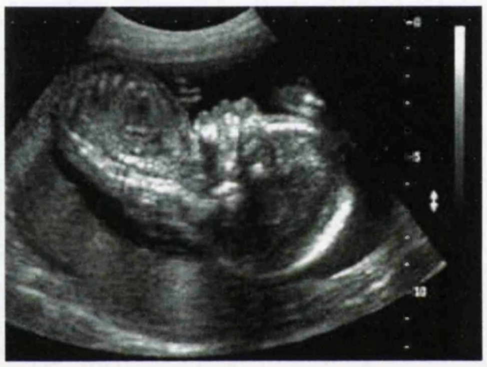
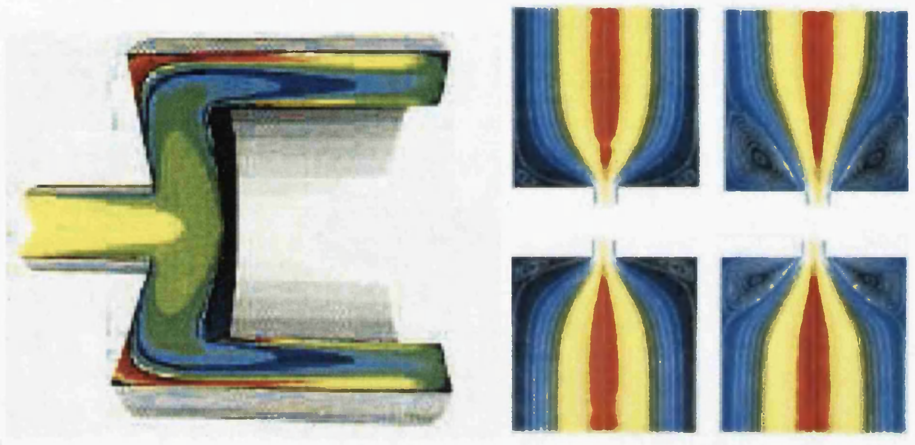

(b)

Figure 2.5: (a) An ultrasound scanner [Fra05] and (b) an ultrasound image [The05].

Ultrasound [Web98], or *sonography*, is a medical imaging technique that uses high-frequency waves for generating images of the human body interior in real-time. Ultrasound is based on the principles of sonar. It is noninvasive and does not use x-ray radiation. Ultrasound is used in obstetrics and gynaecology, cardiology and urology. As ultrasound is able to perform real-time imaging it is now applied in the emergency room as a means for rapid imaging and needle biopsies [SLG$^+$96]. The main equipment in ultrasound is a *transducer*. The transducer has two main purposes: to generate the high frequency waves, and to detect them as they are reflected back by the internal structures or organ in the body. An image is formed by the different sound waves that are reflected by the various tissues. A transducer can also be programmed to act as a sound receiver (*e.g.,* a foetus' heartbeat can be heard during an ultrasound).

Currently, there are two types of ultrasound available: 3D ultrasound imaging and Doppler ultrasound. In 3D ultrasound, 3D images are created from the combination of 2D scan through the application of specialised computer software. These have been described by a collection of landmark papers [FL01, GFWS96, SW95, SO94]. Whereas Doppler ultrasound, which is based on *Doppler effect*, generated an image through the different frequency reflected by the matter depending in its distance, direction and speed from the transducer. The application of 3D ultrasound can be seen when a closer examination of an organ is required while a Doppler ultrasound is used for the examination of blood flow. The drawbacks with ultrasound is the noisy and fuzzy quality image that is produced due to the poor spatial resolution and low ratio of signal to noise. Figures 2.5 (a) and (b) show typical ultrasound equipment and an ultrasound of a foetus, respectively.

2.2.5 Computer Simulation



(a) (b)

Figure 2.6: Simulations of (a) viscoelastic flows and (b) fibre suspensions [Har05].

Scientific modelling and simulation presents us with a modus for generating data which may not be commonly available for explanatory and visualisation purposes. A field that is well-known in scientific simulation is CFD, which it is commonly used to quantitively predict the behaviour and flow processes of fluids. The emergence of CFD began in the 1960s, and since then it has been applied the fields of engineering (*e.g.*, aerospace and automotive), environmental (*e.g.*, atmospheric pollution and fire spread), architecture and building science (*e.g.*, visualisation of the external and internal flows in and around a structure or room). Figure 2.6 shows simulations of (a) viscoelastic flows and (b) fibre suspension where the various colours indicate the various level of compressibility and structures of the materials.

2.3 Surface Extraction

The majority of the images in volume graphics are created from datasets that have been generated through medical scanners such as MRI and CT. In medical science, it is the job of a radiologist to analyse and interpret the results produced by the medical scanners and present them to the physician. It is often difficult for physicians to directly look at the stack of 2D scans and highlight any abnormality that can be seen in scans, as it takes years of experience and training to develop the skills of a radiologist. In volume graphics, the process of surface extraction on volumetric dataset is defined as *isosurfacing*. Other common nomenclatures employed include *surface reconstruction*, *surface tiling* and *surface tracking*. An *isosurface*, or *level surface*, is approximated by extracting all points in a scalar field that match a designate *isovalue* and it can be defined as a function:

$$F : \mathbb{E}^3 \rightarrow \mathbb{R}.$$

Each isosurface, S_τ, for a designate isovalue, τ, is defined, such that:

$$S_\tau = \{ p \in \mathbb{E}^3 \mid F(p) = \tau \}.$$

A popular approach for isosurface extraction is the *marching cubes algorithm* [LC87] which was presented by Lorenson and Cline in 1987. A similar method was also independently developed by Wyvill *et al.* [WMW86] in 1986 for application in modelling soft objects. Since the introduction of the marching cubes algorithm, a significant amount of research effort has been made towards the mechanisms and evolution of the algorithm. This section begins with a brief overview of the marching cubes algorithm. In Section 2.3.2, a review of the topological correctness and accuracy of the marching cubes algorithm is presented. Finally, in Section 2.3.3 an alternative method to the marching cubes algorithm is given.

2.3.1 Marching Cubes

The marching cubes algorithm [LC87] is a well-known method in isosurfacing. In the marching cubes algorithm, a rectilinear grid is divided into cubes where each cube is represented by eight vertices, $v_1, \ldots, v_8$, and twelve edges, $e_1, \ldots, e_{12}$, as shown in Figure 2.7. The marching cubes algorithm is a two phase method where in the first phase the surfaces in each of the cubes are identified and in the second phase surface shading is calculated through the gradient normals obtained from the linear interpolation along the edge.

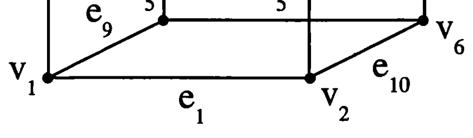


Figure 2.7: Cube indexing.

Using a designated isovalue, τ, an isovalue comparison is performed and can be represented through an 8-bit unsigned vector, $B = b_8b_7b_6b_5\ b_4b_3b_2b_1$, such that:

$$b_i = \begin{cases} 1 & v_i \geq \tau \\ 0 & \text{otherwise} \end{cases}$$

where v_i represents the scalar value of the vertex and 0 or 1 indicates the state of the vertex to be inside or outside, respectively. A cube is defined as *transverse* if and only if $00000000_2 < B < 11111111_2$. A cube is therefore transverse if it contains isosurfaces. There are a possible of $2^8 = 256$ configurations of isosurface-cube intersections as each cube has eight vertices and it can exist in two states, inside and outside. The 256 configurations can be simplified to 14 basic configurations [Lop99] through a specific symmetrical transformation, such as complementary, rotational or reflectional, or a combination of transformations. In the 14 basic cases, there is only one configuration which does not have isosurface-cube intersections. Figure 2.8 shows the 14 basic configurations of the marching cubes algorithm. B can be implemented as an index in a *look-up table.* The look-up table contains all the isosurface-cube intersections configurations and also all edge and surface intersections of the individual configuration.

The look-up table enables for the linear interpolation of edge intersections of an isosurface to be calculated. Using the gradient normals of the isosurface, Gouraud or Phong shading can implemented to provide depth and shading to the isosurface. In order to calculate the gradient vector of the isosurface, estimations of the gradient vectors have to be performed first and then the linear interpolation of the gradient normal is calculated. The gradient calculation is performed through the central differences along each axis, such that [LC87]:

$$G_x = \frac{D(i+1,j,k) - D(i-1,j,k)}{\Delta x}$$
$$G_y = \frac{D(i,j+1,k) - D(i,j-1,k)}{\Delta y}$$
$$G_z = \frac{D(i,j,k+1) - D(i,j,k-1)}{\Delta z}$$

where $D(i,j,k)$ represent the scalar value at position (i,j) in slice k and Δx, Δy, Δz are the distance of the cube edges.

Since the introduction of the marching cubes algorithm as a surface extraction method, a significant amount of research effort has been invested in evaluating and the development of this algorithm as a whole. One of the topics discussed is the correctness and robustness of the marching cubes algorithm, such as the ambiguous configurations in the basic configurations [NH91, Lop99], which is discussed in the next section.

2.3.2 Topological Correctness and Robustness

The marching cubes algorithm has provided a mechanism that not only has simplified surface extraction but has also enabled surface extraction to be performed efficiently and effectively. The surface extraction can be implemented through the use of a look-up table that contains only 14 basic configurations of isosurface-cube intersection. However, this came at a disadvantage at which "holes" can be observed in specific configurations of two adjacent cubes [Dür88]. A possible way of minimising this particular problem is to convert back to the original 256 configurations which has been applied in a *vtk* implementation [SML96] and has also been examined in great detailed by Bartz and Meißner [BM00]. Another drawback that can be noted in the marching cubes algorithm are the ambiguities in the basic configurations, classified as *face ambiguities* and *interior ambiguities* [BW01].

A face ambiguity is defined to be an ambiguity in the exterior topology of the isosurfaces, whereas an interior ambiguity is the ambiguity that occurs in the internal structure of the isosurface. In the time-line of the marching algorithm, much of its early research effort has been concentrated on solving face ambiguities and in the latter stage attention is focused on finding solutions for interior ambiguities or both type of ambiguities. Nielson and Hamann

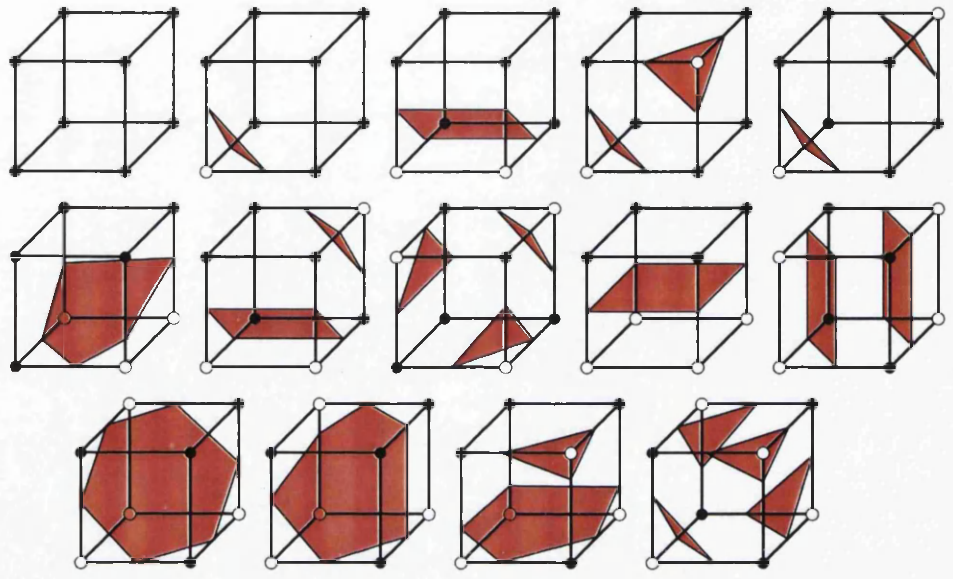


Figure 2.8: The 14 basic configurations of the marching cubes algorithm.

[NH91] presented one of the early works on solving face ambiguity. They demonstrated that six out of 14 basic configurations are ambiguous and that some ambiguous configurations may have as many as nine possible variants. It was also demonstrated that the ambiguous configurations may result in the appearance of "holes" in two adjacent cubes as previously demonstrated by Dürst [Dür88]. Nielson and Hamann introduced a technique called *asymptotic decider* as a solution for face ambiguity where the saddle point of bilinear interpolation is used to indicate the external edges of the isosurface to be constructed. A separate approach was presented by Pasko *et al.* [PPP88] in the aspect of modelling geometric objects and this can be observed in the implementation of *Hyperfun* [Pas05].

Natarajan [Nat94] illustrated the possibility of interior ambiguity in the internal structure of the isosurfaces. Natarajan also demonstrated the use of a body saddle point as an indicator of whether there exist two separate isosurfaces within a cube or only one isosurface which form a shape of a "tunnel". Building on Natarajan's work, Cignoni *et al.* [CGMS00] presented a more recent work on interior ambiguity in which they expanded the 256 configurations to 798 as a method to identify some unique configurations of the marching cubes algorithm. Chernyaev [Che95] presented a detailed review on both face and interior ambiguities. Lopes and Brodlie [LB03] recently presented a paper that amass the researches which have been performed by Nielson and Hamann [NH91], Natarajan [Nat94], Cignoni *et al.* [CGMS00] and Chernyaev [Che95]. In their paper, Lopes and Brodlie also presented a novel approach for improving the topological correctness and robustness of the marching cubes algorithm through the extraction of crucial points in the cube for isosurfaces.

2.3.3 Marching Tetrahedra

The *marching tetrahedra algorithm*, which is an alternative method to the marching cubes algorithm, was put forward by Payne and Toga [PT90] in 1990. The concept of the marching tetrahedra algorithm is similar to that of the marching cubes algorithm, with the exception that tetrahedra are used in the division of the dataset instead of cuboid. In the marching tetrahedra algorithm, a dataset is initially divided into cubes where each cube is further subdivided into five tetrahedra and later isosurfaces in each tetrahedron are identified. Similarly to the marching cubes algorithm, Gouraud and Phong shading can be implemented to provide depth and shading to the isosurfaces.

There are a possible of $2^4 = 16$ configurations of isosurface-tetrahedron intersections as each tetrahedron has four vertices and it can exist in two states, inside and outside. These 16 configurations can be reduced to three basic configurations through a specific symmetrical transformation, such as complementary, rotational or reflectional, or a series of transformations. An advantage of marching tetrahedra in comparison to marching cubes is that not only do the marching tetrahedra present a simpler implementation due to its smaller number of basic configurations, but there is also the smaller probability of ambiguities that could arise in its configurations. Besides that, the marching tetrahedra is not burdened by any patents unlike marching cubes.

A disadvantage of the marching tetrahedra algorithm is the high volume of triangles that would be created, although this issue can be resolved with a post-processing phase where optimisation of the mesh is performed.

2.4 Modelling Schemes

Modelling schemes represent one of the important branches in computer graphics. The field of modelling schemes has provided us with a means of complex object representation through a combination of simple entities. There are two types of modelling scheme and they can be divided into categories of *solid object modelling* and *non-solid object modelling*. Examples of solid object modelling are *boundary representation* [Ago76, Req77, RV83] and *constructive solid geometry* [Req77]. Typical examples of non-solid object modelling are *implicit surfaces* [Bli82a] and *particle systems* [Ree83].

2.4.1 Solid Object Modelling

Solid object modelling is a multi-disciplinary subject that comprises the different fields of mathematics, computer science and engineering. The applications of solid object modelling can be observed in object designing and prototyping, simulation of physical mechanisms and finite-element analysis. In most of these applications, their basic underlying models are constructed from simple primitives and operations. The two most common types of representation in solid object modelling are boundary representation and constructive solid modelling.

The early form of solid object modelling utilised two different categories of representation [Hof92], which were *wire-frame representation* [MW80] and *free-form surface design* [Hof92]. Wire-frame representation provide us with an opportunity to model and represent objects in a quick manner. This form of modelling, however, comes at a disadvantage, as ambiguities could emerge from its method of representation. Free-form surface design, which was also another early form of solid modelling, was however abandoned due to its awkwardness in the evaluation and representation of the object's surface.

Boundary Representation

In *Boundary Representation* (B-Rep), an object is defined in terms of its surface boundaries (*i.e.,* its vertices, edges and faces) [Ago76, Req77, RV83]. The majority of B-Rep systems maintain only objects whose boundaries are 2-manifolds in nature. 2-manifolds can be defined such that there is a continuous one-to-one relationship between the neighbourhood of the points and the disk in the plane. A polyhedron in B-Rep systems is an object that is defined by the following criteria:

- The boundaries of the object are defined by a set of polygons where each of the edges belong to an even number of polygons;
- The edges of the object must each connect two vertices and be shared by exactly two faces;
- A minimum of three edges of the object must meet at each vertex; and
- The faces of the object must not interpenetrate with each other.

The simplest form of a polyhedron is a sphere and the relationship of its vertices, edges and faces can be defined using *Euler's formula.*

A polyhedron can be stated by the Euler formula, such that,

$$V - E + F - 2 = 0$$

where *V*, *E* and *F* represent its number of vertices, edges and faces, respectively. The Euler formula can also be generalised such that it can be applied to polyhedron with holes. This form of Euler formula can be defined as follows:

$$V - E + F - H - 2(C - G) = 0$$

where *H* represent the number of holes in the faces and *G* is the number of holes that pass through the objects whereas *C* can be defined as the number of separate parts of the objects. The polyhedron with holes must also satisfy the series of criteria defined for a polyhedron without holes. For an object that contains a single component, its *G* is defined as *genus*; whereas for an object with multiple components, its *G* is defined by the sum of the genera of its components.

The use of simple representations in B-Rep are however not without its disadvantages, such as the increase in cost for certain computations. In the process of alleviating the increase of computational cost, the architecture of more complex B-Reps such as *winged-edge data* *structure* [Bau72, Bau75], have been introduced. The space-time efficiency of the widely used winged-edge data structure has been investigated by Weiler [Wei85] and Woo [Woo85] together with a collection of alternative B-Rep data structures.

Constructive Solid Geometry

Requicha proposed a method, called *Constructive Solid Geometry* (CSG), used in solid object modelling for creating complex objects using regularised Boolean operations [Req77]. The complex objects are created from a combination of operations on primitive objects. The primitive objects in CSG can be divided into three categories of:

- simple pre-defined primitives such as cuboid and spheres;
- swept primitives, which are discussed in the next section, and
- algebraic half-spaced primitives.

There are three types of regularised Boolean operations in CSG. They can be defined as *regularised union*, *regularised intersection* and *regularised difference*. Regularised Boolean operations are used in CSG instead of the normal set theoretic operations as they are able to provide the closure of the operations on the interior of the two primitives, which can be applied in solving the dangling points problem [Req77].

A tree structure can be used in the representation of CSG in which the primitives are represented by the leaf nodes and the regularised Boolean operators (or transformations such as translations and rotations) are represented by the interior nodes. This tree structure is commonly known as a *CSG tree*. The edges of a CSG tree are ordered, as Boolean operations are generally non-commutative. A CSG tree is processed using a depth-first algorithm where the nodes from the leaves are combined in an upwards manner through the tree. *Cell-decomposition* and *spatial occupancy enumeration* are known as restricted form of CSG [FvDFH95].

Sweeping

Sweeping was introduced by Binford in 1971 as one of the techniques used in solid object modelling [Bin71]. There are three types of sweeping that can be done. They are *translational sweeps*, *rotational sweeps* and *general sweeps*. Translational sweeps are the simplest form of sweeping as an object is built by sweeping a 2D area along a path through space. This way each swept object is the resulting multiplication of its object's area and the distance of the sweep.

In a rotational sweep, an object is constructed by rotating an area about an axis. A general sweep is, however, similar to translational sweep with the exception that the object's area or volume may vary in geometric, size or orientation as it travels along the arbitrary curved path. In computer vision, a general sweep of a 2D cross-section is also known as a *generalised cylinder*. Due to the nature of a general sweep, several problems can observed in its application such as the difficulty to model it efficiently and that its final result may not generally produce a solid.

Although a swept object can be applied in CSG, the applications of regularised Boolean operations on a swept object in its natural form can be complicated, and the resulting operations may not be closed under regularised Boolean operations. Therefore, it is necessary for a swept object to be converted to a different form of representation before performing regularised Boolean operations on it.

Cell Decomposition

Another technique that can be seen in solid object modelling is *cell decomposition* [Ago76] in which complex objects are constructed from a collection of solid primitives such as cubes, spheres, and tetrahedra. As briefly mentioned, cell decomposition is also seen as a restricted form of CSG as its objects are formed through the union of disjoint cells. In the majority of the cases for cell decomposition its object representations are considered to be unambiguous. One of the drawbacks of cell decomposition is that each pair of cells may have to be examined for intersection, which is hard to verify. The application of cell decomposition can be observed in the field of finite element analysis [WM85].

Spatial Occupancy Enumeration

In *spatial occupancy enumeration* [MS74], a space is divided up into cells or voxels (volume elements). The common geometric nature of these cells are often cube-shaped and the arrangements of cube-shaped cells in space are known as cuberille. In spatial occupancy enumeration, an object is constructed by defining which of the cells are occupied and which of them are not. There are several advantages and disadvantages that can be found in spatial occupancy enumeration. One of advantages is that any possibility of ambiguity in object representation can be eliminated as the position of each individual cell can be straightforwardly determined (*i.e.,* whether the cell is in the interior or exterior of the object).

One of the disadvantages which can be observed is that the notion of *partial* occupancy is not supported by spatial occupancy enumeration. With the absence of *partial* occupancy, many of the constructed objects are only approximations of the desired ones. Another disadvantage of spatial occupancy enumeration is that for a reasonable resolution of an object, a high storage space is required. The applications of spatial occupancy enumeration can be seen in the field of medical imaging technique such as CT. A composite scheme of CSG and spatial occupancy enumeration has been presented by Reddy and Rubin [RR78].

Octrees

Octrees [Mea82, RR78], which are a 3D version of the quadtree, were presented in the late 1970s and early 1980s as a method of reducing storage requirements. The octree algorithm operates in a similar manner to that of the quadtree where the bounded volume of an object is recursively subdivided into octants, at which the occupancy of each octant is recorded. The subdivision of the volume is carried out until a suitable balance can be defined between the storage space and complexity.

The implementations of octrees have been observed in the field of volume rendering. Levoy [Lev90] has implemented octrees as *binary pyramid* (or *complete octrees*) which can be used as a method of accelerating ray tracing. Levoy and Whitaker [LW90], later, introduced the use of a pyramid of averaged values for gaze-directed volume rendering. In 1991, Laur and Hanrahan [LH91] applied the technique of binary pyramid for both averaged values and estimated error in a *splatting algorithm*. While recently, Yang *et al.* [YLS00] presented the usage of both pyramids and octrees as a method of enabling interactive volume rendering.

Binary Space-Partitioning Trees

Originally implemented as a method for determining visible surfaces in the rendering process, *binary space-partitioning* (BSP) *trees* [TN87] is another technique that can be seen in solid object modelling. BSP is a dimension-independent algorithm that recursively subdivides space into pairs of subspace. It is being used for representing arbitrary polyhedra. As a BSP tree is dimension independent, it can be used to simulate both quadtrees and octrees. One of the disadvantages of BSP is that an arbitrary BSP tree may not generally represent a bounded solid.

A BSP tree can also be constructed from B-Rep as demonstrated by Thibault and Naylor [TN87]. Thibault and Naylor have also demonstrated the acts of combining both BSP tree and a B-rep through the usage of regularised Boolean operations and also ways of detecting which of the polygonal pieces are on a BSP tree's boundary.

2.4.2 Non-Solid Object Modelling

An object can be constructed either implicitly or explicitly. In the previous section, the explicit manner of constructing objects has been investigated. The focus of this section is on the modelling of non-solid objects or objects that cannot be defined easily through explicit approaches. The introduction of describing objects implicitly has presented us with novel ways of defining objects. In this section, the various ways of modelling non-solid objects, such as *implicit surfaces* [Blo97] and *fractals* [FFC82, Bar93], are examined and they are described as follows:

Implicit Surfaces

Implicit surfaces are typically described either through a mathematically or procedurally defined function. A surface, S, in an implicit surface model is defined through a function, F, such that

$$S = \{\, p \in \mathbb{E}^3 \mid F(p) = 0 \,\}$$

where p represents the sampling location. The use of implicit surface model in computer graphics can be traced back to "blobby models" [Bli82a] which was used to model and display electron density fields. Further applications of implicit surface models have been investigated by Wyvill *et al.* [WMW86] and Nishimura *et al.* [NHK+85] and are more commonly known as "soft objects" and "metaballs", respectively. The "blobby model", "soft objects" and "metaballs" have been each been produced through exponential, sixth-degree polynomial and piecewise quadratics functions, respectively.

In 1992, Duff [Duf92] demonstrated the application of CSG operations on an implicit surface model. Later, Wyvill *et al.* [WGG99] presented a framework, called *BlobTree*, that describes the use of an implicit model with CSG operations, blending and warping together with two rendering algorithms.

Volumetric Representations

Volumetric representation is one of the techniques that can be observed for modelling non-solid objects. The construction of objects through volumetric representations is widely used in the field of medical imaging [SFF91]. The results that are produced by the medical imaging techniques (CT and MRI) are stacks of 2D scans or cross-sectional views of the objects. Examining and visualising these cross-sectional views are not without difficulties in comparison with 3D models of these scans. The examination of these 2D scans in a 3D manner would help the physician to provide a diagnosis for many medical conditions. This type of visualisation is commonly known as volumetric rendering.

Volumetric representations can be classified as an extension of the spatial occupancy enumeration. The attributes of each cell or voxel in volumetric representations are defined by its scalar fields. There are many benefits that can be associated with volumetric representations, amongst them are its capability to handle both solid and amorphous phenomena as well as its ability to define both the interior and exterior of an object. Another advantage of volumetric representations is its simple approach to manipulate an object's data structure.

Particle Systems

In a *particle system* [Ree83], fuzzy phenomena such as fire, fog, grass, clouds and water can be modelled using a collection of particles. Through particle systems, these fuzzy phenomena are modelled easily in comparison with conventional rendering techniques. Each particle in a particle system can be defined by a life cycle that begins with its conception into the system followed by its growth that is defined by its changes and movements within the system, and finally its exit from the system after a certain duration of time. A particle system is often used as a method for modelling fuzzy objects instead of conventional rendering techniques because of its simplicity, procedural definition and also its dynamical nature.

Fractals

Fractals [FFC82, Bar93] are objects that in different levels of magnifications would contain copies of themselves at different scales, in other words they contain the property of self-similarity. It was Mandelbrot who coined the word "fractal" from the Latin verb *frangere* meaning "to break" [Gle88]. There are many different forms of fractals and some of them such as *Mandelbrot set*, *Julia set*, *Cantor set* and *Sierpinski's Triangle* can be seen implemented in computer graphics [Man82]. Due to its time-varying parameters and cost efficiency, fractals are also being used in computer graphics to generate terrains and realistic landscapes [WW92].

Grammar-based Models

A *grammar-based model* [Smi84] is a formal language technique that is used to represent an object together with its topology and geometry. Smith [Smi84] has demonstrated the use of grammar-based models to construct a collection of plant models. Grammar-based models are used to represent objects as they are an efficient method of data representation. Another technique that can be used to model plants is *Lindenmayer systems* or *L-systems* [PLH+90] in which whole plants are constructed using the behaviour of cells of plants.

2.5 Constructive Representations of Volume Objects

Complex objects can be constructed from a recipe of simpler objects together with a combination of transformations and constructive operators. Some of the early work on this technique had been carried out by Ricci [Ric73] and Rvačev [Rva63]. The application of constructive operators in the field of Civil Engineering has been illustrated by Rvačev in 1963. A decade later, Ricci presented a generalised approach to modelling complex objects through the use of constructive operators. Requicha [Req80] introduced a framework, called *Constructive Solid Geometry* (CSG), for modelling complex solid objects in the Boolean domain through the combination of simpler objects, such as bounded primitives or half-spaces, and regularised constructive operators. A complex object in CSG can be represented through an ordered tree which consists of primitives and constructive as well as transformations operators represented by the nodes and leaves, respectively. The constructive operators that are defined in CSG are *union*, *intersection* and *difference*.

Kaufman [Kau00], however, illustrated that as CSG is defined over the Boolean domain, modifications had to be carried out to the pre-existing CSG operators in order for it to be applicable to volume datasets (which are normally defined in the real domain). As well as extending the use of CSG operators in the real domain, these modifications are able to resolve the issue of discontinuity in the pre-existing CSG operators. An alternative extension to the CSG, called *Blobtree*, was introduced by Wyvill *et al.* [WGG99] in the aspect of modelling implicit surfaces. As the BlobTree is applicable in implicit surface modelling, its operators are not only limited to the constructive operators, such as union, difference and intersection, but also deformation operators, such as *blending* and *warping*, for object distortions.

This section presents the major frameworks that can be used in the construction of volume objects. These frameworks can be classified as:

- Volumetric CSG;
- Constructive Volume Geometry;
- Constructive Hypervolume Modelling; and

- Volume Scene Graphs.

In this section, each of these frameworks is reviewed and an evaluation of the framework is then given.

2.5.1 Volumetric CSG

Fang and Srinivasan [FS98] proposed a framework, called *Volumetric Constructive Solid Geometry* (VCSG), which is an extension of CSG, for modelling complex volumetric objects using simple volumetric primitives and operators in the volume domain. The primitives used in VCSG are normally volumetric data sets. As VCSG operates in both Boolean and volume domains, the volumetric dataset can be of binary, integer or real types.

A tree structure can be used in the representation of VCSG in which the volumetric primitives are represented by the leaf nodes and the operators are represented by the intermediate nodes. The operators in VCSG can be classified into two groups of unary operations and binary operations. A unary operation can be formally defined as a function $F : \mathbb{E}^3 \rightarrow \mathbb{E}^3$ and a binary operation can be formally classified as a function $F : \mathbb{E}^3 \times \mathbb{E}^3 \rightarrow \mathbb{E}^3$. Examples of the unary operations include *affine transformations*, such as translation, rotation, scaling, shearing and reflection; *transferring*, (*i.e.,* modification of a scalar through a transfer function); *plane-cutting* and *deformation.* The binary operations, however, consist of union, intersection and difference. The binary operations specified in VCSG are flexible in that their definitions are application-oriented. An example of such flexibility can be seen in the union operation where it can be applied in the retrieval of the maximum parameters or used as a blending application.

Additionally, together with the VCSG framework, a rendering algorithm was also described for visualising VCSG tree. In order to improve the efficiency of the rendering process, Fang and Srinivasan [FS98] also proposed the use of bounding boxes and subdivision methods for the computation of the VCSG term.

2.5.2 Constructive Volume Geometry

An algebraic framework, called *Constructive Volume Geometry* (CVG), for modelling complex spatial objects using combinational operations was proposed by Chen and Tucker [CT00, CT98]. In CVG, a volumetric representation of an object, that is, a *spatial object*, is a tuple

$$\mathbf{o} = (A_0, A_1, \ldots, A_k)$$

of scalar fields defined in $\mathbb{E}^3$ where $k \geq 0$. A scalar field is a function $F : \mathbb{E}^3 \rightarrow \mathbb{R}$.

Typically in CVG, A_0 is a representation of an *opacity* field which determines the visibility of the spatial object and A_i is used to represent other *attribute fields* of the spatial object, such as colours and reflection coefficients. The attribute fields specified are generally bounded by the application and rendering algorithms that are used in the implementation. An opacity field of every $p \in \mathbb{E}^3$ can be formally defined as a function:

$$\mathrm{O} : \mathbb{E}^3 \rightarrow [0, 1]$$

where the values 1 and 0 indicate whether a point p is either to be opaque or transparent, respectively, and translucent for any values in between. Although, scalar fields are used in the demonstration of the CVG framework, this is not an indication of its limitations. A spatial object in CVG framework can be extended to include vector or tensor fields, which will be seen later in Chapter 4.

The combination of a *spatial object class* $\mathbf{O}(\Sigma)$ and a set of operations $\{\Phi_i\}$ can be defined as a *CVG algebra*, where Σ denotes the collection of names for space, attributes and scalar fields. The definitions of the CVG algebra can be specified for various graphics applications and a few of them have been demonstrated by Chen and Tucker. Together with demonstrations, several sets of algebraic laws for them have also been derived. CSG is embedded in the CVG class in the form of $\mathbf{O}(\Sigma_{Boolean-opacity})$. $\mathbf{O}(\Sigma_{Boolean-opacity})$ can be achieved through the substitution of the range $[0,1]$ by the Boolean domain $\mathbb{B} = 0,1 \subset [0,1]$ in $\mathbf{O}(\Sigma_{opacity})$. Basic operations that can be defined in $\mathbf{O}(\Sigma_{opacity})$ include:

$$\begin{aligned} \textit{union}: &\quad \boxed{\cup}(\mathbf{o}_1,\mathbf{o}_2) = \mathbf{MAX}(\mathrm{O}_1,\mathrm{O}_2) \\ \textit{intersection}: &\quad \boxed{\cup}(\mathbf{o}_1,\mathbf{o}_2) = \mathbf{MIN}(\mathrm{O}_1,\mathrm{O}_2) \\ \textit{difference}: &\quad \boxed{-}(\mathbf{o}_1,\mathbf{o}_2) = \mathbf{SUB_{01}}(\mathrm{O}_1,\mathrm{O}_2) \end{aligned}$$

where **MAX**, **MIN** and **SUB**$_{01}$ are the pointwise extensions [Ric73] of scalar operators **max**, **min** and **sub**$_{01}$ and are defined as follows:

$$\begin{aligned} \mathbf{max}(s_1,s_2) &= \begin{cases} s_1 & \text{if } s_1 \geq s_2 \\ s_2 & \text{otherwise} \end{cases} \\ \mathbf{min}(s_1,s_2) &= \begin{cases} s_1 & \text{if } s_1 \leq s_2 \\ s_2 & \text{otherwise} \end{cases} \\ \mathbf{sub}_{01}(s_1,s_2) &= \mathbf{max}(0, s_1 - s_2). \end{aligned}$$

The basic operations in $\mathbf{O}(\Sigma_{opacity})$ can be extended to include operations such as *blend*, *cap* and *trim* which can be found in $\mathbf{O}(\Sigma_{4-colours})$. The extended operations in $\mathbf{O}(\Sigma_{opacity})$ together with their derived laws for them and $\mathbf{O}(\Sigma_{4-colours})$ are demonstrated later in Chapter 5.

2.5.3 Constructive Hypervolume Modelling

A framework, called *Constructive Hypervolume Modelling* (CHM), for modelling implicit objects was introduced by Pasko *et al.* [PASS01] in 2001. Formally, CHM can be illustrated as an algebraic system which consists of a triple set of objects, operations and object relations represented by $(\mathbf{O}, \Sigma, \mathbf{W})$, respectively. Similarly to CSG and CVG, an object in CHM can be represented through recursion of a constructive tree where the primitives are represented by the leaves and the nodes represent the operations and relations. Texturing methods can be applied to a constructive tree, producing a *constructive texturing tree.* An implementation of the CHM can be observed in *Hyperfun* [Pas05].

A spatial object can be defined as a tuple

$$\mathbf{o} = (A_0, A_1, \ldots, A_k)$$

of scalar fields where the individual field is of the form $F : \mathbb{E}^n \longrightarrow \mathbb{R}$. A_0 is defined as the geometry field which indicates the point set of the implicit object, and A_i is used to represent other attribute fields of the implicit object, such as colour, transparency and temperature. The value that is returned by A_0 determines the position of a point. A point is said to be either inside or outside the object depending on the positive or negative returned value, respectively.

There are two types of operation that can be found in a set of operations $\{\Sigma_i\}$; unary and binary. Some of the operators documented for the unary operations include *bijective mapping*, *affine mapping*, *projection*, *offsetting* whereas the binary operations consist of union, intersection, difference, blending, cartesian product and metamorphosis. The operations defined for the union, intersection and difference for CHM is based on the *R-functions* introduced by Rvačev [Rva63].

2.5.4 Volume Scene Graphs

A hierarchical structure called a *volume scene graph*, for defining scenes comprising of volume datasets and space-filling functions was proposed by Nadeau [Nad00]. Volume scene graphs do not only support datasets generated through medical scanners, such as MRI and CT, but also those specified procedurally. A relationship between volume scene graphs and the algebraic framework CVG can be defined such that spatial objects can be constructively incorporated into the scene.

Volume scene graphs can be modelled using a tree structure, where the primitives are represented by the leaf nodes and the operators are represented by the interior. Each node, N, in volume scene graph is defined, such that

$$N = \{\, p \in \mathbb{E}^3 \mid F(p,t) = s \,\}$$

where s is the sampling value of p at time t. The examples of some commonly used functions in the nodes include colour, retrieval of sampling value, affine transformations and composition. In addition to those, other nodes such as *Interpolator*, *Distance* and *Gradient*, can also be implemented in the volume scene graphs.

2.6 Volume Rendering

This section describes the subject of *volume rendering*. Volume rendering is a method that can be utilised to synthesise images of volumetric data, and generally refers to *direct volume rendering*. The direct volume rendering algorithm is the direct rendering of a volumetric data without the production of any intermediate alternative representation, and can be divided into categories of *image-order* or *backward rendering* methods and *object-order* or *forward rendering* methods. A significant amount of research effort has been invested in the developments of volume rendering techniques, which have been highlighted in major surveys by Yagel [Yag96], Chen [Che01] and Brodlie and Wood [BW01]. This section begins with a brief introduction to the volume rendering integral. This is followed by various rendering techniques such as *ray casting*, *MIP* rendering and *splatting*. Finally, this section finishes with a recently introduced rendering technique called *spectral volume rendering*.

2.6.1 The Volume Rendering Integral

The volume rendering integral defines the intensity of colour at a specified pixel along a corresponding ray, and it is the core of the volume rendering algorithm. Kajiya and Von Herzen [KV84] and Max [Max95] have demonstrated the volume rendering integral in its low-albedo structure, which is the foundation of most volume rendering techniques and can be seen in the majority of optical and illumination models.

The volume rendering integral can be derived, such that:

$$I_\lambda = \int_0^T e^{\int_0^t \alpha(v)dv} \alpha(t)\mu_\lambda(t)dt \tag{2.1}$$

where T is the length of the ray and I_λ is the amount of light accumulated at wavelength λ between 0 and T. $\alpha(t)$ is defined as the opacity of some point along the ray where 1 and 0 indicate whether a point is either to be opaque or transparent, respectively, and translucent for any values in between. $\mu_\lambda(t)$ reflects the light emitted at along the ray at wavelength λ. $\mu_\lambda(t)$ can be computed using a standard shading model, such as Phong [Pho75]. Typically, the computation of I is evaluated over three wavelengths, red, green and blue. Further dissection of the volume rendering integral has been presented by Mueller *et al.* [MMC99].

2.6.2 Ray Casting

Ray casting is one of the algorithms that can be implemented in direct volume rendering and is defined as an image order method. A leading paper on ray casting has been published in 1988 by Levoy [Lev88]. Equation 2.1 (in Section 2.6.1) outlines the ray casting integral. In practice, such a continuous integral cannot be implemented and thus has to be evaluated numerically using a Riemann sum approximation. The discrete approximation of the ray casting integral is defined, such that:

$$I_\lambda = \sum_{i=0}^{n} \alpha(i\Delta t)\phi_\lambda(i\Delta t)\Delta t \prod_{j=0}^{i-1} e^{-\alpha(j\Delta t)\Delta t} \tag{2.2}$$

where n is the total number of samples taken along the ray and $\Delta t = \frac{T}{n}$ defines the sampling distance between each point. Equation 2.2 can be further simplified by the replacement of its exponential term with the first two terms of the Taylor expansion. The resulting simplification of Equation 2.2 can then be written as follows:

$$I_\lambda = \sum_{i=0}^{n} \alpha(i\Delta t)\phi_\lambda(i\Delta t)\Delta t \prod_{j=0}^{i-1}(1 - \alpha(j\Delta t)\Delta t). \tag{2.3}$$

The sample points in Equation 2.3 can be calculated by sampling uniformly along the ray.

There are two form of compositing that can be implemented in ray casting. They are *front-to-back* compositing and *back-to-front* compositing. An observation should be made that the order of compositing is associative in nature but not commutative, which is essential in the implementation of parallel rendering applications. The implementation of front-to-back compositing in ray casting can be considered as an advantage as it can help to decrease the computation time through *early ray termination.*

2.6.3 MIP Rendering

The *Maximum Intensity Projection* (MIP) technique was proposed by Avila *et al.* [ASK94] in 1994. MIP was originally utilised as a medical imaging technique called *Magnetic Resonance Angiography* (MRA) in assisting doctors in diagnosing of whether a particular artery is partially blocked which may then required the intervention of a surgery. The MIP technique works by setting each pixel value to the greatest scalar value along the ray. This technique can be seen as a simplified version of the volume rendering integral with its resulting image greatly resembling an X-ray image.

The resulting image of the MIP technique is, however, not synonymous to that of an X-ray image as the former is a non-linear projection of the highest intensity value while the latter is a linear projection of intensity onto an image. Some of the advantages of this technique are that it is computationally fast and is able to highlight regions of highest intensity. A disadvantage of the MIP technique is that it does not provide depth information of the data set. Further enhancements in improving the depth-cued of an MIP image have been presented by Jones [Jon95], in the context of animation.

2.6.4 Splatting

Splatting, or *footprint evaluation*, was initially introduced by Westover [Wes90, Wes91]. The splatting technique works by projecting voxels in discrete volumetric data to the screen and is defined as a forward rendering method. A significant amount of research effort has been invested in the development of the splatting technique. Laur and Hanrahan [LH91] presented a progressive refinement algorithm for the application of real time rendering using Gouraud shaded polygons [Gou71] in their estimations of the splats. Crawfish and Max [CM93] later improved on the performance of the approach by the substitution of several polygons with a texture mapped square. Further developments on splatting have been demonstrated by Mueller and Yagel [MY96] and by Nulkar and Mueller [NM01] where splatting has been extended to include perspective projection and shadows for increased perceived realism, respectively.

2.6.5 Spectral Volume Rendering

The *spectral volume rendering* technique was first introduced by Noordmans *et al.* [NvS00] in 2000. In their approach, the material is separated into an absorbing medium and scattering particles. The spectral volume rendering used by Noordmans *et al.* is based on Kajiya and

Von Herzen's two-pass algorithm [KV84] that involves an illumination phase and a radiation phase. Bergner *et al.* [BMD02, BMDF02, BMD04, BMTD05], later, presented a one-pass spectral volume rendering integral which allows interactive data exploration through the factorisation of the light source during evaluation of the volume rendering integral. Recently, Abdul-Rahman and Chen [AC05] presented a spectral volume rendering integral based on the Kubelka-Munk theory [KM31], which simulates the feature of one dimensional volumetric radiosity [DH96]. Their approach is different from the previous spectral volume rendering techniques in terms of its physical basis of absorption and scattering coefficients. A hardware implementation of the spectral volume rendering based on the Kubelka-Munk theory has also been presented by Strengert *et al.* [SKB$^+$06].

2.7 Optical Models

The physical process of how light interacts with a material is one of the factors that determined the colour that is seen by the eye. The interaction of light with a material, such as absorption, emission and scattering, are processes that are normally observed in nature. These processes have been successfully imitated in the computer graphics and are commonly known as optical and illumination models. There are several major surveys of optical models in the aspect of direct volume rendering. A comprehensive survey of the optical models has been presented by Max in 1995 [Max95] ranked by realism. An extension of this paper has been presented by Max and Chen [MC06] as a response to the recent advancements in optical models for volume rendering, such as refraction rendering and spectral volume rendering. An application-specific survey of the optical models has also been presented by Baranoski and Krishnaswamy [BK04] in 2004.

The optical models presented in this section are arranged by rank of realism and accuracy of the interaction of light with matter, similar to that of Max [Max95] and Max and Chen [MC06]. This section begins with absorption models, emission models and a combination of both models. This is followed by single scattering and shading models and also shadow models. The section finishes with multiple scattering and refraction models, which is one of the more recent advancements in optical models for volume rendering.

2.7.1 Absorption, Emission and Combined

The absorption models, emission models and a combination of both models are the simplest form of optical models that can be implemented, and a notable amount of research effort has been focused on them. The first type of absorption and emission model was introduced by Blinn [Bli82b] in 1982 for modelling the rings of the planet Saturn, a cloudy material. This model is commonly known as *Blinn's model*, and some part of it can be seen in the other cloud models discussed in this section. Blinn's model takes into account the scattering, transparency and shadowing effects of light on the clouds. One of the assumptions made in Blinn's model is that only a single particle is being used to scatter a ray of light and any other scattering is considered to be insignificant. This assumption can only be made if the *albedo* of each particle is small. The brightness of the cloud layers is determined by the shadowing and blocking effect of other particles as a light ray is bounced by a single particle. Blinn's model also takes into account the transparency of the cloud layers by calculating the amount of light that travels from the back of the cloud whose path has not been blocked by the particles. Finally, ray integration is used to compute the brightness of the cloud at each specified pixel. Later, Max [Max86] also presented a method of modelling light diffusion through clouds of constant density using single scattering approximation.

Since then, other models of absorption and emission have emerged such as one presented by Drebin *et al.* [DCH88] for modelling a volume of varied materials. In this model, it is assumed that there is only a single scattering of radiation that travels from the light source to the eye. The resulting light intensity is determined by the properties of the materials in the volume. Shirley and Tuchman [ST90] have also presented an absorption and emission model that uses an algorithm called *Projected Tetrahedra* that uses tetrahedral volume cells for determining the colour and opacity of the volume data set. Max *et al.* [MHC90] presented an illumination model for calculating the density of clouds and contour surfaces, while Westover [Wes90] demonstrated a forward mapping algorithm that defines a final image through a calculated shaded volume. Other models of absorption and emission models have been presented by Wilhelms and Van Gelder [WG91], Williams and Max [WM92], Malzbender [Mal93] and Totsuka and Levoy [TL93].

2.7.2 Single Scattering and Shading

As briefly discussed in the previous section, Blinn's model [Bli82b] also takes into consideration of the single scattering and shading effects of light on the clouds. In 1988 Levoy [Lev88] presented a shading model as a method of directly displaying the smooth surface of a volume without the intermediate construction of surface polygons. The shading at each voxel is computed using contour surface normals. Similar to the model presented by Drebin *et al.* [DCH88], this model can also be used to model a volume of varied materials. The opacity at each voxel is computed so that it contains a specified grey-level value. The opacities at or near a presumed surface are then enhanced by weighting them with a grey-level gradient. Using back-to-front compositing, the resulting opacities and colours are computed to form an image.

In 1990, Tiede *et al.* [THB$^+$90] examined a collection of shading models for visualising datasets that have been generated through medical scanners, such as MRI and CT. The collection of shading models have been examined from various gradient implementations such as z-buffer, grey-level, and adaptive grey-level and two different variations of the marching cubes algorithms.

2.7.3 Shadows

Shadows perform a significant role in the portrayal of realistic rendering of objects that is absent in the previous discussed models. Kajiya and Von Herzen [KV84] presented one of the early models of shadows in computer graphics for modelling clouds, flames and particle systems. Kajiya and Von Herzen's model is an extension of Blinn's model that is a two-pass algorithm computing the light intensity and scattering of every voxel. Kajiya and

Von Herzen's model is different from the rest of the shadow models as it utilises a three-dimensional radiative scattering approximation.

One-dimensional shadows models have been presented by Max [Max86] and Kaneda *at al.* [KONN90]. Kaneda *at al.*'s model uses the concept of radiosity as ambient light to simulate realistic indoor images. Nishita *et al.* [NMN87] have also presented another shadow model which takes multiple lights into consideration in constant opacity value and the presence of other opaque objects. Nishita *et al.*'s model can be used to model scattering and absorption of light caused by particles in the atmosphere. An application-based shadow model for medical visualisation has also been presented by Meinzer *et al.* [MMS$^+$91] in 1991 and is commonly known as the *Heidelberg ray tracing model.*

2.7.4 Multiple Scattering

The models that have been discussed in the previous sections can be classified as single scattering models that consider only either a single ray reflection or scattering. Single scattering models, however, are insufficient to model realistic rendering of high albedo participating media such as atmospheric clouds. These may require multiple scattering models using the concept of radiosity that can be found in thermal radiation heat transport. Rushmeier and Torrance [RT87] presented the *zonal method* which can be applied in the computation of radiative transfer for participating medium. The zonal method can be used in the portrayal of realistic rendering of synthetic images; it takes into account the interactions between light and matter such as emission, scattering and absorption and also the interactions between volume and surface. Other multiple scattering models have also been presented by Bhate [Bha93] and Sobierajski [Sob94]. More recent work demonstrating multiple scattering models can be seen in a paper by Kniss *et al.* [KPHE02] in 2002.

2.7.5 Refraction

One of the recent advancements in optical models for volume rendering is the introduction of refraction rendering in volume graphics. The implementation of refraction in volume rendering is important as it is able to provide us with more realistic representation of objects through improved visual effects of depth and shape that were unavailable in the predecessor optical models. The refraction model was first introduced by Rodgman and Chen in 2001 [RC01], and since then a notable amount of research effort has been invested in it [RC05, LM05a, LM05b, RC06]. Rodgman and Chen [RC01, RC05] presented a refraction model using discrete ray tracing and discussed the important part that normal estimation plays in the refraction rendering of volume data sets. In their work, they have also outlined a collection of key techniques in computing the relative refraction indices of each sampling point. Li and Mueller [LM05a, LM05b] later examined the computational cost of refraction rendering in volume visualisation and presented a framework for implementing the refraction rendering of volume data sets at a near interactive rate.

2.8 Volume Deformation and Animation

One of the main functional components of volume graphics is volume deformation and animation [Che01]. In computer graphics, *deformation* is defined as the change of geometrical shape of an object and its applications can be seen in the areas such as computer animation, scientific visualisation and surgical simulation. However, in the context of computer graphics, *computer animation* is defined as the modelling, control and rendering of temporal behaviour of graphical objects [CIJ+05]. Both motion and deformation play an important role in the animation as they help to define the interaction procedures between the various objects. A number of major surveys have been performed on deformation. These surveys can be divided into applications such as medical image analysis [MT96] and surgical simulation [LTCK03]. The models and methods that can be found in deformation can generally be divided into groups of *functional and procedural models*, *parametric models*, *polygonal meshes* and *physically-based deformable models* [CIJ+05]. These models and methods are discussed in greater detail later in the section. Animation is an active and ongoing research topic, and this can be seen by the various books that have been published on the subject of animation [FvDFH95, Vin92, Par01] and the numerous literature that is written on it annually [The06a, Eur06, IEE06]. The methods that can be seen in animation can also be divided into groups of *articulated models*, *procedural control*, *constraint-based control*, *physically-based control*, *stochastic control* and *behavioural control* [CIJ+05]. These methods are studied in further detail later in the section.

2.8.1 Volume Deformation

Chen *et al.* [CIJ+05] defined deformation as:

> "...the intended change of geometric shape of an object under the control of some external influence such as a force."

Before an object can be deformed, it is generally required that the object holds a form of data representation and a physically- or mathematically-based algorithm [CIJ+05]. These types of object are defined as *deformable models*. There are two types of deformation method that can be implemented on a volume objects: *non-physically-based models* and *physically-based models* [CIJ+05]. In non-physically-based models, their deformation algorithms contain a minimum or a restricted amount of the physical behaviour of deformable objects. However in a physically-based model, their deformation algorithms are designed to mimic the physical behaviour of deformable objects based on physical laws such as Newton's laws of motion and the linear elastic theory [MKN+04, KMH+04].

A number of functional and parametric models in traditional computer graphics can be transferred and implemented in the non-physically-based deformable models. Examples of such functional and parametric models are the implementations of the *global and local deformation* [Bar84, Bar86] on volume objects through ray deflectors [KY95] and spatial transfer functions [CSW+03]. Occasionally, volume objects in the non-physically-based deformable models are defined over surface or solid models as a form of defining deformation. Such form of deformations have been demonstrated by Whitaker [Whi04] and Lefohn *et al.* [LKHW03] in the implementation of level sets for deforming surface-based objects. A more recent development in the non-physically-based deformable models can be seen in the area of point-based representations in the implementation of free-form deformation [PKKG03] and haptic texturing [HBS99].

In the physically-based deformable models, the objects are normally defined together with mesh data representations. Such mesh representations can be observed in the form of triangular or rectangular for surfaces whereas for solid or volumes in the form of tetrahedral or hexahedral. A significant number of surveys have been carried out on the physically-based deformable models; the most recent one has been presented by Nealen *et al.* [NMK+05]. Examples of some of the physically-based deformable models that have been widely implemented in computer graphics include finite element methods [PW84, BC96] and continuum models [TPB87, TW88].

2.8.2 Volume Animation

In comparison with volume deformation, there is a limited amount of literature that can be found on volume animation. The techniques that are used in volume animations can be generally divided into ***blocked-based volume animation*** and ***skeleton-based volume animation*** [CIJ+05]. The research in volume animation is generally aimed at the changing of the posture or the position of a defined volumetric object. Most of the techniques that are generated for volume animation do not define any form of deformation, *i.e.,* the shape of the object is not deformed and only the voxels are moved. There are two types of problem that can occur during the repositioning of a volume object [CIJ+05]. One problem is the breakage of the joints during large movements, while the second problem can be seen in deciding which of the internal structures of the object are attached to which specified joint.

The first blocked-based animation technique was proposed by Wu and Prakash in 2000 [WP00]. In their approach, the volumetric object is first divided into blocks of voxels in order to define the major sections in the character. Then a finite element method is used to calculate the deformed movement of each block structure. Finally, each of the deformed block is re-voxelised through 3D texture mapping. These three steps form the movement of the character at each time interval. Chen *et al.* [CSW+03] later presented the application of spatial transfer functions and constructive volume geometry in block-based animation, which permits block-based animation to be performed at each interval without the need of re-voxelisation. Further work on block-based animation has been presented also by Singh and Silver [SSC03, SS04] and Islam *et al.* [IDSC04].

An alternative approach to blocked-based animation is the skeleton-based volume animation. Skeleton-based volume animation was first proposed by Gagvani and Silver in 1999 [GS99]. Skeleton-based volume animation is similar to the animation process that is applied to surface polygonal models [CIJ+05], and consists of the three steps of skeletonisation, manipulation and reconstruction of the volume and rendering. Further developments on the skeleton-based animation can be found in the later work of Gagvani and Silver [GS00, GS01]. Walton and Jones [WJ06] recently presented a method called *volume wires* which uses curve-skeletons to nonlinearly deform volume objects in an intuitive manner.

The method presented is non-reconstructive, and animation specifications consist only at control points for the wires.

Chapter 3

Colour Models in Computer Graphics

Contents

3.1 Introduction 36
3.2 Background on Colour 37
3.3 Concepts 38
3.4 Human Visual System 41
3.5 CIE System 42
3.6 Colour Models 43
3.7 Colour Mixing 45
3.8 Colour Constancy 46
3.9 Colour-Spectrum Transformations 47
3.10 Spectral Representation Models 48

3.1 Introduction

Colour plays an important role in nature, seen in the reproduction strategies of plants and flowers, *e.g.*, how a flower attracts a bee for pollination; and as defense and offense mechanisms for plants and animals, *e.g.*, how a zebra camouflages itself or how the patterns on a blue butterfly are used to ward off its predators. Humans experience colour from birth and it is often taken for granted. Without colour, the world would be a dull place and the task of recognising and identifying objects would be a difficult exercise for the eye. The effective usage of colour plays an important part in computer graphics [Mac99, Tru81] and several major surveys have been carried out examining the relationship between colour and technology [ST97, San00].

This chapter presents a comprehensive literature review on the subject of colour models in computer graphics, which has been implicitly segregated into three separate aspects of foundation, standardisation and representations of colour. The foundation of colour models in computer graphics begins with a short history of colour research, which consists of a time line that goes back to the ancient Greeks and concludes with the most recent development in computer graphics. To provide further understanding on colour, the terminologies and concepts of colour (such as *wavelength* and *spectra*) are defined. The interactions between light and matter define the physical process of how a colour seen and these interactions includes *emission*, *scattering* and *absorption*, which are examined in detailed in this chapter. The foundations of colour models in computer graphics concludes with image formation and the mechanism of the *Human Visual System.*

The penultimate part of this chapter examines the standardisation of colour specifications and the different colour models in computer graphics and visualisation [FvDFH95]. The standardisation of colour specifications comprises of the *CIE system* and the uniform colour space while the examination of the different colour models includes the listing of the various colour models. The final part of this chapter investigates colour mixing, colour-spectrum transformations and spectral representation models.

The organisation of this chapter is as follows:

- In Section 3.2, a time line of colour research is presented.
- In Section 3.3, a discussion of the concepts and fundamentals of colour models in computer graphics is presented.
- In Section 3.4, the mechanisms of the Human Visual System is explored.
- In Section 3.5, a discussion of the CIE system, which is a proposed method of colour standardisation, is given.
- In Section 3.6, a review into the different types of colour model that are available in computer graphics and visualisation is presented.
- In Section 3.7, a brief review of the different versions of colour mixing techniques that can be found in the computer graphics and industries is presented.
- In Section 3.8, the concept and colour constancy is examined and presented.
- In Section 3.9, an outline of the colour-spectrum transformations is given.
- In Section 3.10, a review of the different categories of spectral representations is presented.

3.2 Background on Colour

The study of how colour is perceived by the eye can be traced back to the ancient Greeks [OW82]. Aristotle proposed an hypothesis which postulated that the eye and light are of the same entity. This hypothesis was upheld for thirteen centuries until in 1000 A.D. Alhazen defined the separation between the eye and light and classed them as two different entities. The major step towards the understanding of colour was later made in 1666 by Newton with his light splitting experiment. In his experiment, a glass prism is used to separate a narrow beam of white light into a spectrum of colour that ranged from violet through blue, green, yellow, orange and red. Using another glass prism Newton then joined the spectrum of colour back into a narrow beam white light. It is from here that the word *spectrum* is coined [OW82].

This breakthrough opened the path for Young and Fresnel to carry out investigations into the concepts of *polarisation* and *diffraction*. Building on these works Maxwell was able to propose the theory of electromagnetism that speculated that light was constantly being emitted by matter. This speculation was later disputed by Planck in 1900 who later described that light is only emitted in small quantities of energy instead of continuous nature [HZ74]. Such view has allowed the development of quantum mechanics where the definition of *photons* [Lew26] was introduced, which later lead to the further comprehension on the interactions between light and matter.

The interactions of light with matter, such as *transmission*, *absorption* and *scattering*, are processes that are normally seen in nature. The amount of light that is transmitted, absorbed and scattered is dependent on the property of the material. For example, the transparency and opaqueness of a material can influence the amount of light that is transmitted and absorbed while the surface of the material can determine how much light is being scattered or reflected. As light is a form of energy, the interactions between light and matter can be mathematically defined as [OW82]:

$$\textit{Light entering} = \textit{Light reflected} + \textit{Light transmitted} + \textit{Light absorbed}$$

The interactions with light and matter have been modelled in computer graphics for some time and are commonly known as optical and illumination models. The first model was implemented by Blinn [Bli82b] in 1982 for modelling the rings of the planet Saturn. Since then, a notable amount of research effort has been in developing models that are able to simulate the interactions between light and matter, and one of the most recent development for the optical and illumination models is the introduction of refraction rendering in volume graphics by Rodgman [RC01] in 2001.

3.3 Concepts

The colour that is seen by the eye is the result of the perception of the visible light. The visible light is just one of the many radiations that can be found in the electromagnetic spectrum, as illustrated in Figure 3.1. On one end of the spectrum are radio waves while at the opposite end of spectrum are gamma rays, with nine orders of magnitude difference in wavelength between them. The electromagnetic spectrum can be represented in terms of *wavelengths*, *frequency* or *energy*. The relationship between the wavelength, λ, and frequency, v, can be mathematically defined as [OW82]:

$$\lambda = \frac{c}{v}$$

where c represents the speed of light ($2.998 \times 10^8 m/s$). Each of the energy components, E, in the electromagnetic spectrum can be defined as:

$$E = hv$$

where h represents the *Planck's constant* [OW82]. The units measurement for wavelength, frequency and energy are *metres* (*m*), *Hertz* (*Hz*) and *electron-volt*, respectively. A more commonly used units measurement of wavelengths are *nanometres* ($10^{-9}nm$).

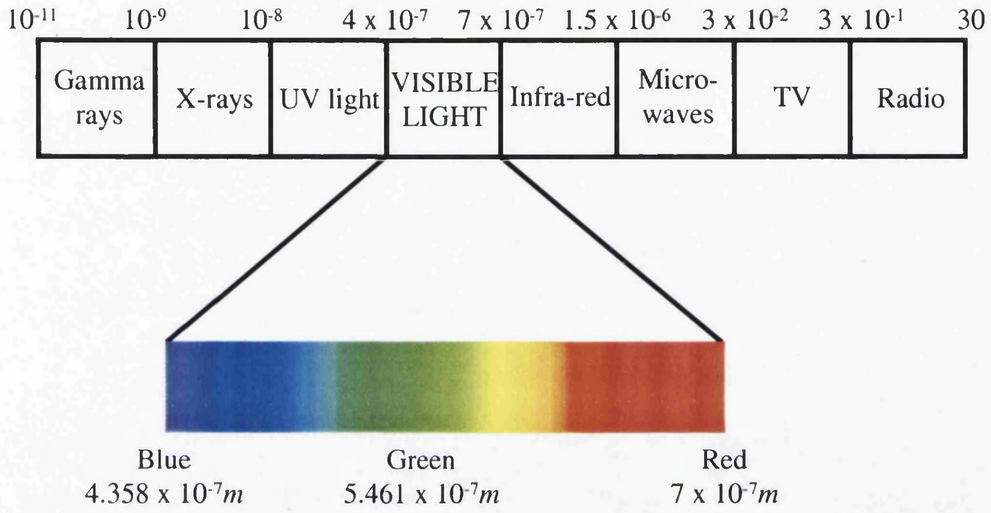


Figure 3.1: The electromagnetic spectrum [McA04].

Visible light is in the range of $380nm$ to $780nm$. However, in colour computation this range is narrowed down to $400nm$ and $700nm$, as the eye is more sensitive within this region and this range is the greatest amount of spectra that can be ignored while retaining acceptable accuracy [SSS92b, SSS92a]. The amount of energy that is detected at each wavelength can be represented in terms of *spectral power distribution* (SPD). A SPD can be directly calculated to its corresponding colour and the relationship between a SPD and colour is a *many-to-one* as several different SPDs can be mapped to a same colour sensation. This type of relationship is commonly known as *metamer* and has been proven by Sharma and Trusell [ST97]. The concept of *metamerism* can be seen in the field of colour reproduction. There are various forms of metamerism but the most common type of metamerism is *illuminant metamerism* where two sample colours match under a specific illuminant but may not match under all illuminant causing a *colour mismatch*. This type of colour matching is known as *metameric match* or *conditional match*. A *spectral match* can, however, be defined when two colour samples with identical SPDs would match under all illuminant.

There are many different colours that can be seen around us and various terminologies have been defined in an effort to classify them. Nassau [Nas01] has identified 15 ways of how a colour can be generated by an object while Foley *et al.* [FvDFH95] has presented two different techniques of classifying colours – *subjective* and *objective*. In subjective classification, the specification of a specific colour is not only determined by its *hue*, *saturation* and *lightness* (or *tints*, *shades* and *tones*) but also by external factors such as viewers' interpretation, sampling size, lighting and surrounding environments. Objective classification is, however, more straightforward as a colour can be specified quantitively or more scientifically known as *colourimetry*. Colourimetry is the science of colour measurement and it is a subject that can be found both in physics and chemistry.

3.3.1 Light Interactions with Matter

There are three main interactions that can occur when light interacts with matter. They are emission, scattering and absorption [OW82, BK05]. A thorough discussion of these interactions is given as follows.

Emission

Emission can be classified into the two different categories of *thermal emission* and *luminescent emission* [Gla95]. Thermal emission occurs when additional heat energy is being radiated by an object in the form of light. The amount of light that is emitted by a thermal source is mainly determined by the type of material it is made of and its temperature. An example of thermal emission is an incandescent light bulb. Luminescent emission is, however, caused by the amount of energy that can be found within the object. Unlike thermal emission, luminescent emission is not dependent on its temperature as its main contributing factors. The difference between thermal emission and luminescent emission is that thermal emission is produced by the object itself while luminescent emission is generated as a reaction to the external light energy.

Scattering

One of the three main interactions that can occur when light interacts with matter is scattering. There are three forms of scattering and they are *Mie scattering*, *Rayleigh scattering* and *reflective-refractive scattering* [BK05]. All of these scatterings can easily be observed in nature. Mie scattering [BW99], or *aerosol scattering*, plays a significant role in the comprehension of the appearance of everyday objects such as milk and can be observed in the lower portion of the atmosphere such as the appearance of white clouds. Mie scattering is the scattering of lights when its wavelength of the radiation and particles are similar in size. Rayleigh scattering, *molecular scattering* or *selective scattering* is the scattering of light when particles are smaller than its wavelength of radiation. Rayleigh scattering can be observed when light travels in transparent liquids and solids and it is the primary reason why the sky is blue. Reflective-refractive scattering is primarily produced by the structure of tissues and the refraction of light between air and cells. Reflective-refractive scattering is a form of internal scattering that can be observed in organic tissue such as human skin and plant leaves [BK05].

Absorption

As light is a form of energy, a light that is shone onto an object that is neither emitted or scattered would be absorbed by the object. The light that has been absorbed by the object is usually transformed into heat energy, which contributes to the heating effects that can be observed in the object. The more light that is emitted or scattered by the object, the less light that is absorbed by the object (and vice versa). Absorption can be classified into two different sub-categories of *general absorption* and *selective absorption* [JW76]. General absorption is when an object reduces the intensity of all wavelengths of radiation equally, while selective absorption occurs when an object reduces only the intensity of certain wavelengths of radiation.

3.4 Human Visual System

This section explores the mechanism of the *Human Visual System* (HVS) [Gla95]. Rods and cones are the two classes of receptors that can be found in the eye and their sensitivities are determined by the presence of light detected. Rods are located across the retinal surface, and there are approximately 75 to 150 million of them [GW92]. Rods become sensitive when it is dark and their sensitivity decreases as the amount of light received by the eye increases. The level of detail recognised by rods is low as they have a large distribution region and several rods are joined together to a single nerve end. Rods are not involved in colour vision and are only able to provide a general image of the field of view. Rod vision is known as *scotopic* or *lim-light vision.*

Cones are found in the central position of the retina, called the *fovea*, and in each eye there is an estimated 6 or 7 million of them [GW92]. They are the receptors that determine the colour that is seen by the eye. As each cone is connected to its own nerve end, the level of detail recognised by cones is high. Cone vision is known as *photopic* or *bright-light vision.* There are three cones that can be found in the eye and their responsiveness varies accordingly to wavelength [KB96]. These three cones have peak sensitives in the long, middle and short wavelength regions and are more commonly known as *SML* cones [SMJ93, SS98], with peaks located roughly at $420nm$, $530nm$ and $560nm$, respectively.

The research into *SML* cones is still an ongoing process. It has been estimated that the number of *L* and *M* cones are more than the *S* cones with the ratio of $6 : 3 : 1$ for *L*:*M*:*S* [CAS$^+$91, HNN98]. Each of the cone signals can be used to form receptive fields either through the arithmetic of addition or subtraction. The receptive fields would have varied properties as each of the *L*, *M* and *S* cones have different spectral sensitivities. A simplified model representing the receptive fields has been proposed by Lee [Lee96]. The simplified model is built based on the *opponent channel theory* and assumes that there exist three types of colour receptive fields. The *black-white channel* or *luminance channel* has the highest spatial resolution and is formed by the addition of the *L* and *M* cones. The *red-green channel*, which has the second highest spatial resolution, is formed by the subtraction of the *L* cone from the *M* cone, while the *yellow-blue channel*, which has the lowest resolution, is formed by the subtraction of the *S* cones from the addition of the *L* and *M* cones. The opponent channel was first proposed by Hering in 1878 [Her78], which assumed there are four individual colour channels: red, yellow, green and blue, that are unique. Each of these channels can be mixed to form other colours and can only exist as opposite pairs. For example, either the colour red or green will be perceived by the eye but never the colour greenish-red.

3.5 CIE System

This section studies the standardisation of colour specifications that have been proposed by *Commission Internationale de l' Éclairage* (CIE) [Ber00, FvDFH95]. The initial standardisation of colour specifications was made in 1931 with the introduction of *colour matching functions*. The first recommended set of colour matching functions is the *CIE RGB*, $\bar{r}(\lambda)$, $\bar{g}(\lambda)$, $\bar{b}(\lambda)$, which is the product of two separate experiments [Wri29, Gui31]. The application of CIE RGB is not, however, without its drawbacks. One of the drawbacks of CIE RGB is the increase in the complexity and computational cost due to negative and positive values that can be found in the colour matching functions. As CIE RGB is the product of two different experiments, concerns were also raised as to the likelihood of discrepancies in the photometric values.

In view of these problems, a second set of colour matching functions, *CIE XYZ*, $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, $\bar{z}(\lambda)$, which covers a 2° angle vision was introduced. CIE XYZ was able to resolve the drawbacks that are found in CIE RGB as all the values in its colour matching functions are positive and its $\bar{y}(\lambda)$ has been set to correspond to luminous efficiency [WS82]. CIE XYZ is often used in computer graphics and is commonly known as *CIE XYZ 1931* to differentiate between *CIE XYZ 1964*, which was introduced later to cover a wider angle of vision. CIE XYZ 1964 is used in the industries for quality control application. The colour matching functions can be found in [WS82].

In a CIE model a colour, C, can be mathematically defined as,

$$C = X\mathbf{X} + Y\mathbf{Y} + Z\mathbf{Z}$$

where $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ represent the variables in the CIE model and X, Y and Z are the units or weights to be applied to the variables. Using the colour matching functions, the X, Y and Z can be specified as:

$$X = \int_\lambda SPD(\lambda)\bar{x}(\lambda)d\lambda$$
$$Y = \int_\lambda SPD(\lambda)\bar{y}(\lambda)d\lambda$$
$$Z = \int_\lambda SPD(\lambda)\bar{z}(\lambda)d\lambda$$

The normalised values of $X + Y + Z$ can be used to generate *chromaticity values* or *chromaticity coordinates* such that:

$$x = \frac{x}{X+Y+Z}; \quad y = \frac{y}{X+Y+Z}; \quad \text{and} \quad z = \frac{z}{X+Y+Z} = 1 - x - y. \qquad (3.1)$$

Equation 3.1 shows that z can be easily determined from the x and y values.

The chromaticity values can be used to generate a two-dimensional map of colour called *chromaticity diagram* that is only determined by dominant wavelength and saturation. One of the drawbacks that can be observed in using the chromaticity diagram is the loss of information as only two variables are used instead of the usual three. The chromaticity values, x and y, and Y can be used to generate a three-dimensional CIE space, *CIE xyY*, where a Y axis is plotted rising from the illuminant point in the chromaticity diagram. The chromaticity values, x and y, and Y can be transformed to the X, Y and Z units such that,

$$X = \frac{x}{y}Y; \quad Y = Y; \quad \text{and} \quad Z = \frac{1-x-y}{y}Y.$$

The chromaticity diagram can be used to specify *colour gamuts* or *colour ranges*. The colour gamuts can be applied to identify the ranges of colour in different colour devices and can be used for accurate colour reproduction between the devices. Hall [Hal89] has presented a review on colour-gamut compression.

One of the disadvantages of CIE XYZ is that it is not visually spaced. To overcome this drawback, CIE LUV was introduced in 1976 as a *perceptually uniform* colour space whereby two colours that are equally apart are perceived to be equally distant by the eye. Transformations between the CIE XYZ and CIE LUV can be found in [FvDFH95].

3.6 Colour Models

There are many different colour models in computer graphics and visualisation [FvDFH95]. Examples of the colour models include RGB, CMY and YIQ colour models. A notable amount of research effort has been made in the reviews of the different colour models [MG80, JG78, SB75] and their effectiveness in various environments [SCB87, DK96]. The majority of the colour models are application-specific and can be easily converted into each other. This section explores some of the extensive colour models that are utilised in the field of computer graphics and visualisation.

RGB Colour Model

The RGB (Red Green Blue) colour model is one of the commonly used colour models in computer graphics and visualisation. The colours in the RGB colour model are specified through a 3D Cartesian coordinate system where red, green and blue are the axes and primary colours. The RGB colour model is an additive colour model and its application can be seen in monitors and televisions. The RGB colour model can be converted to the CIE XYZ system and other colour models. The conversion between the RGB colour model and the XYZ system is significant as the CIE system is a general standard. The conversion between the RGB colour model and the XYZ system is defined such that,

$$[X \quad Y \quad Z] = [R \quad G \quad B]\,\mathbf{M}$$

where M is a 3×3 transformation matrix constructed from the chromaticities of a specific monitor.

A commonly-used colour model for synthesising scenes involving translucent objects in computer graphics was introduced by Porter and Duff in 1984 [PD84]. This colour model is called the RGBα model and consists of four channels; three colour channels, red, green and blue, and an α channel, which indicates the opacity. Oddy and Willis [OW91], later, introduced a 7-colour model as an extension of the RGBα model. The 7-colour model is also known as *Bath colour model.* One of the objectives of the 7-colour model is to improve the optical accuracy of the RGBα by presenting a β channel and separate RGB colour properties for pigments and medium filters. Willis [Wil06] recently presented a paper that is able to unify the RGBα model and the 7-colour model in a single framework. There are also several versions of the RGB colour model such as the sRGB, or standard RGB, colour model which has been proposed by Hewlett-Packard and Microsoft [SACM06].

CMY Colour Model

An alternative colour model to the RGB colour model is the CMY (Cyan Magenta Yellow) colour model. The CMY colour model is a subtractive colour model as its colours are defined through the subtraction of colour components from a white light. The CMY colour model is a hardware-specified model and its application can be seen in the printing procedure. The CMY colour model can be converted to the RGB colour model and this conversion is considered to be important as there is no direct conversion between the CMY colour model and the CIE system. The conversions between the RGB and CMY colour models are defined such that,

$$\begin{bmatrix} C & M & Y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} - \begin{bmatrix} R & G & B \end{bmatrix}$$

and

$$\begin{bmatrix} R & G & B \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} - \begin{bmatrix} C & M & Y \end{bmatrix}.$$

The CMY colour model can be extended to a CMYK colour model, where K represents the colour black. The relationship between the CMY and CMYK colour models can be defined such that,

$$\begin{aligned} K &= \min(C, M, Y) \qquad & C &= C - K \\ M &= M - K & Y &= Y - K. \end{aligned}$$

YIQ Colour Model

One of the most utilised colour models is the YIQ colour model. Similarly to the other two colour models the YIQ is also defined based on a 3D Cartesian coordinate system and is used in the U.S. colour television broadcasting. For European colour television broadcasting an identical model, called the YUV colour model has been defined. In the YIQ colour model, the Y component does not represent the colour yellow but instead luminance and has been specified to be equal to that of the CIE Y primary. The conversion between the RGB colour model and the YIQ colour model is specified such that,

$$[Y \quad I \quad Q] = [R \quad G \quad B] \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix}$$

Other Colour Models

The colour models that have been examined so far are hardware-specified models. Alternative versions of colour models can also be defined for the specification of colours for a particular colour gamut such as intuitive-based or perceptive-based colour models. Examples of intuitive-based colour models are HSV (Hue Saturation Value) and HLS (Hue Lightness Saturation) colour models. The HSV, or HSB (Hue Saturation Brightness) colour model [Smi78] is a user-oriented colour model, which incorporates the concepts of tint, shade and tone. HSV and HLS colour models are both specified based on the cylindrical coordinate system.

Examples of the perceptive-based colour models includes *Munsell Color System* [Mun05, Mun15] and *Natural Color System* [Swe79]. The Munsell Color System was developed as part of a teaching material for art students and it is based on the concept of equally visual spacing. The Munsell Color System is defined as a three-dimensional colour space with hue, lightness and chroma as its dimensions and are referred to as *Munsell hue*, *Munsell value* and *Munsell chroma*, respectively. The Natural Color System (NCS) is also a three-dimensional colour space, which has been built using the opponent channels theory [Her78]. The dimensions of the NCS has been divided into *NCS hue*, *NCS blackness* (or *NCS whiteness*) and *NCS chromaticness*. A revised version of the NCS can be found at [Swe79].

3.7 Colour Mixing

Colour mixing is an art that has been introduced to us from an early age and has been greatly practiced in the field of computer graphics and visualisation. This section presents an overall picture of the various types of colour mixing techniques. Colour mixing is predominantly divided into two categories of *additive colour mixing* and *subtractive colour mixing* [Nas01, Rig97]. Additive colour mixing generally involves the mixture of coloured light, whereas subtractive colour mixing requires the selective subtraction of colour segments from a white light.

There are various types of additive colour mixing technique. The RGB colour model is one of the most commonly used colour models for additive colour mixing. The RGB additive colour mixing technique can be observed in the RGBα model [PD84], which is generally utilised in computer graphics. An alternative version of colour mixing technique has been presented by Grassmann in 1853 [Gra54, Bou47] and is known as *Grassmann's laws of colour mixing*. Additive colour mixing can be seen in televisions and monitors.

Subtractive colour mixing techniques can be divided into *simple subtractive colour mixing* and *complex subtractive colour mixing* [Ber00]. Simple subtractive colour mixing involves only the process of absorption, whereas complex subtractive colour mixing involves both the processes of absorption and scattering. Examples of the simple and complex subtractive colour mixing are Bouguer-Beer's law [Bee52, Bee54] and Kubelka-Munk theory [KM31], respectively. These laws can be seen in the field of photography and painting industries.

3.8 Colour Constancy

Colour constancy [Lan77, MW86, DI93] and metamerism are two subjects that are closely connected to each other. The difference between colour constancy and metamerism is the number of colour samples that are involved in the portrayal of their phenomenon. Briefly, metamerism occurs when two colour samples (which are of different physical properties) are perceived by the eye to be of the same colour. In colour constancy, however, a sample colour would be perceived by the eye to be the same regardless of its changing illumination conditions. Colour constancy can be seen as a form of stability in colour perception and is a characteristic of the human visual system. The eye is able to provide us with the stability in colour perception as it is embedded with a mechanism that enables itself to conform to changing illumination conditions. The capability to conform to changing illuminant is known as *chromatic adaptation* [Rig97]. Colour constancy is also known as the *illuminant estimation problem* and is important as it provides us with the ability to recognise and identify objects without the confusion of colour changing objects.

Colour constancy, however, cannot be observed in all visual systems. An example of such a situation is the digital camera visual system where a similar scene has been lit using the two different illuminant of daylight and tungsten. The objects under the tungsten illuminant would have reddish appearance, whereas the objects under the daylight illuminant would have bluish appearance. The breakdown in colour constancy is known as *colour inconstancy* or *colour constancy failure* [Hil97]. Colour inconstancy can also happen in scenes with objects that exhibit high reflectivity or scenes that use monochromatic illuminant. *Light constancy* [Hil97] is a term that has been used to illustrate the situation where objects are judged by their reflectance comparison rather than their absolute energy levels. Light constancy can only be performed when there is a direct comparison between the two objects, such as how a piece of coal is distinguished as black and a piece of chalk as white.

Colour constancy plays an important role in the aspect of image production. It has been broadly researched in both of the fields of human and computer vision [BF94, DI93, DI94, FHH01, Lan77, MW86] and is still an ongoing research topic. Colour constancy can be achieved through the amendment of the colour responses based on the changing illumination conditions. A possible amendment is to generate a mapping of the sensor responses between different illuminant [The06b].

One of the many algorithms that has been developed in an effort to implement the mapping of the sensor responses is the *diagonal model, gamut mapping* or *von Kries model* [Lan77, For88b, Fin95]. The mapping of the sensor responses under different illuminant can be mathematically defined as [The06b]:

$$\begin{bmatrix} R' & G' & B' \end{bmatrix} = \begin{bmatrix} R & G & B \end{bmatrix} \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix}$$

where R, R', G, G', B' and B represent the red, green and blue sensor responses under two different illuminant, while α, β and γ represent factor changes of the sensor responses. The initial work on gamut mapping was performed by Forsyth in 1988 [For88b, For88a]. In his research, Forsyth demonstrated that it is not possible to achieve a single solution for the mapping and that a mapping has to be chosen out of all possible mappings. Since then, a significant amount of research effort has been made in improving the algorithm [Bar00, Fin96, FH00].

Another algorithm that can be utilised for colour constancy is *colour by correlation* [FHH01]. Colour by correlation is based on gamut mapping whereby a computational framework is utilised in allowing a reformulation of the gamut mapping approach. The performance on colour by correlation can be improved upon by incorporating probability information in its calculation [DI94].

3.9 Colour-Spectrum Transformations

This section outlines the algorithms that can be implemented for the colour-spectrum transformations. The algorithms of colour-spectrum transformations are generally grouped into two separate categories of transformation, namely, RGB-to-spectrum transformation and spectrum-to-RGB transformations. The transformations from colour-to-spectrum and vice versa have always been performed using the RGB tuple because the RGB colour model is a predominant form of colour representation and there is a direct transformation between the RGB colour model and CIE XYZ colour space which is required from a spectrum transformation.

3.9.1 RGB-Spectrum Transformation

RGB-to-spectrum transformations are required as it has been demonstrated that the representations of colour through the RGB tuple is insufficient and presents us with constraints [Hal89], such as the complications of rendering physical optics phenomena. One of the proposed solutions is the implementation of spectral representations as a way for modelling colour. The relationship between an RGB tuple and a spectrum is a *one-to-many*, which can also be known as metamer. The RGB-to-spectrum transformation is still an ongoing research topic and a significant amount of research effort has been invested in the development of the RGB-to-spectrum algorithms. The RGB-to-spectrum transformations can be divided into two classes of algorithm and they are the approximation of spectrum through linearly independent basis functions and the reconstruction of a spectrum using a set of metamer [Smi99] or measured reflectance [WXS04].

The approximation of spectra through linearly independent basis functions ranges from the elementary *delta* [Gla89] and *box functions* [Hal89, HG83] to the more complex combination of linearly independent basis functions of *Fourier* [Hor84] and *Gaussian* [SFCD99] or other *fitting functions* [Mey88]. The investigations of the Fourier functions as basis functions in the reconstruction of spectra have also been carried out by Wandell [Wan87] and Glassner [Gla89]. Drew and Funt [DF92] have outlined the usage of Fourier Basis but through the approach of metamerism. Each of the linearly independent basis functions has pros and cons. One common disadvantage that can be found in all of the linearly independent basis functions is the likelihood of negative energy for RGB tuple belonging to highly saturated colours [SFCD99]. This feature is examined in further detail later on the thesis.

3.10 Spectral Representation Models

There are several major surveys into the different spectral representation models [SFD98, RP98]. Spectral representation models examine alternative methods of how spectral functions can be approximated and represent a vital part in the fields of colour science [WS82], realistic image synthesis [FvDFH95, Gla95] and colour image analysis [BLL96, Kli93]. In the surveys presented by Sun *et al.* [SFD98, SDF99] and Rougeron and Péroche [RP98], the models have been divided into five categories of *sampling method*, *analytical method*, *linear model*, *colour model* and *composite model*. Sun *et al.* [SFD98, SDF99] have proposed five benchmarks at which the models should to be judged on and they are accuracy, compactness, computational efficiency, data portability and flexibility. A discussion of the categories of spectral representation models is given as follows.

Sampling Method

The sampling method [SFD98, RP98] presents us with a direct and easy approach of representing spectral functions and has been applied in the laboratory for recording spectral function measurements [VGI94, NNJ43]. A function in a sampling function is typically defined as a continuous function over discretely sampled points. In a spectral representation model, a spectral function is defined by sampling evenly spaced wavelengths across a specified visible range, $[\lambda_{min}, \lambda_{max}]$. The sampling distance, $\Delta\lambda$, of the spaced wavelengths is defined such that,

$$\Delta\lambda = \frac{\lambda_{max} - \lambda_{min}}{N-1}$$

where N is the total number of sampling points and λ_{min} and λ_{max} are generally specified to be $400nm$ and $700nm$, respectively. Applications of sampling methods have been successfully implemented in realistic image synthesis [GMN94, CT82] and colour science [SSS92b, Mey88].

Advantages of the sampling method are its data portability and flexibility as there are no restrictions on its sampling distance and visible range. Computational efficiencies of spectral representation models are often examined from the cost of multiplying two spectral functions, such as the calculation of reflection and transmission of a specific point [SFD98]. The computational efficiency and accuracy of the sampling method is, however, dependent on the range of its sampling distance and visible extent. As the sampling distance between the points decreases, the level of accuracy of the spectral functions increases. But by decreasing the sampling distance, the computational cost of the sampling method also decreases. A drawback of the sampling method is that it cannot be used to represent spectral functions that contain high frequency elements, such as fluorescent illuminant [SFD98].

Analytical Method

Spectral representations in the analytical method are often approximated using polynomials [Moo45] as their analytical functions. A set of polynomial co-efficients can be used to approximate a spectral function whereby the degree of the polynomials determines the compactness of the function. Applications of polynomials in analytical functions can be observed in the area of image synthesis [RF91, GHJ96]. One of the advantages of applying polynomials in the analytical method is that of its data portability with various situations. Some of the drawbacks that can be found are its inflexibility and computational inefficiency in calculating the product of two spectral functions [SFD98]. Polynomials in the analytical method also have restricted accuracy due to the low polynomial degree needed for computational stability [For57].

Linear Model

One of the other models for spectral representations is the linear model and was introduced as a way of overcoming the problems that were encountered in the sampling and analytical models. Spectral functions in the linear model can be approximated using a linear combination of a set of linearly independent basis functions [DF92, GNHR98]. The set of linearly independent basis functions can be generated from Fourier sinusoids or Chebychev polynomials [GHJ96], but typically they are generated to model possible spectral functions in a particular region [DF92, Pee93]. Some of the advantages of the linear model lie in its accuracy, compactness and flexibility in its approximation of the spectral functions. Some of the drawbacks of the linear model are its data importability and computational inefficiency in calculating the product of two spectral functions [SFD98].

Colour Model

The colour model is one of the spectral representation models that approximate spectral functions and it is a popular option for approximating spectral functions as it is simple and easy to implement. The applications of a colour model for approximating spectral functions can be observed in colour image analysis [Kli93, BLL96] and realistic image synthesis [FvDFH95, Gla95]. Some of the advantages of the colour model are its compactness, computational efficiency and data portability. Some of the drawbacks of colour model are its inflexibility and the inaccuracy that may occur due to the loss of information.

Composite Model

The composite model was proposed by Sun *et al.* [SFD98, SDF99, SFDC01] as a technique of approximating spectral functions that contain both smooth and spiky elements. The smooth spectral functions can be represented using a linear combination of a set of basis functions, whereas in the spiky spectral functions, the sampled spikes are defined using the positions and heights of the spikes with its smooth layout represented using the same method used in the smooth spectral functions. By approximating the spectral functions in these separate manners, the composite model is able to satisfy all five benchmarks proposed: accuracy, compactness, efficiency, portability and flexibility, that the rest of the previous models failed [SFD98]. A drawback of the composite model is that its efficiency can be affected in specified conditions where their reflectance light sources and objects are restricted.

Chapter 4

Spectral Volume Rendering

Contents

4.1 Introduction . . . 51
4.2 Related Work . . . 52
4.3 The Kubelka-Munk Theory . . . 53
4.4 Spectral Volume Modelling . . . 56
4.5 Spectral Volume Rendering . . . 57
4.6 Sampling Correctness . . . 58
4.7 Visual Effects and Correctness . . . 64
4.8 Post-illumination and Spectral Transfer Functions . . . 68
4.9 iPi – Interactive Post-illumination . . . 78
4.10 Summary . . . 86

4.1 Introduction

Our sense of colour is a product of the *physical process* of how light interacts with a material and the *perceptual process* of how the colour is seen. It is important for graphical or visualisation systems to synthesise images in a manner that reflects the correct physical process of colour. Therefore, the focus in this chapter is on the modelling and rendering of volumetric objects based on the Kubelka-Munk theory of diffuse reflectance. The Kubelka-Munk theory defines the volumetric relationship between reflectance and transmittance of a material together with its absorption and scattering properties.

In this chapter, the utilisation of the Kubelka-Munk theory as a spectral volume rendering integral for volume visualisation is presented. The implementation of the Kubelka-Munk theory as a volume rendering integral shows that more accurate optical effects can be achieved through the control of absorption and scattering properties than the direct manipulation of the RGB colour and opacity in a conventional RGBα model. By incorporating spectral volume rendering together with combinations of several natural and artificial colour spectral data sets, a higher level of visual realism of both solid objects and amorphous phenomena can be achieved in volume visualisation. Through the Kubelka-Munk theory as a spectral volume rendering integral, an approach to the design of spectral transfer functions for post-illumination that facilitates interactive exploration of volume data sets can be presented.

The organisation of this chapter is as follows:

- In Section 4.2, a brief overview of the use of different colour models in the fields of graphics and visualisation, and previous work on direct volume rendering and spectral volume rendering is given.
- In Section 4.3, a description of the Kubelka-Munk theory, on which this work is based, is presented.
- In Section 4.4, a spectral modelling scheme for representing volumetric objects and their absorption and scattering properties is outlined.
- In Section 4.5, our spectral volume rendering integral based on the Kubelka-Munk theory is presented.
- In Section 4.6, the correctness of our rendering integral is compared to existing spectral volume rendering methods.
- In Section 4.7, the optical effects of varying the absorption and scattering properties of an object are examined. In this section, the optical correctness of the spectral volume rendering based on the Kubelka-Munk theory in comparison with a conventional volume rendering integral based on the RGBα model is also discussed.
- In Section 4.8, our approach to the design of spectral transfer functions is described together with the use of post-illumination for interactive exploratory visualisation.
- In Section 4.9, a system designed to demonstrate the technical feasibility of post-illumination for interactive exploratory visualisation is presented.

Finally in Section 4.10, our observations and concluding remarks are presented.

4.2 Related Work

There are many different colour models in computer graphics and visualisation [FvDFH95], such as RGB, CMY and YIQ colour models. The RGBα model introduced by Porter and Duff in 1984 [PD84] is perhaps the most commonly-used colour model for synthesising scenes involving translucent objects. Oddy and Willis [OW91] improved the optical accuracy of the RGBα model by introducing a 7-colour model with a β channel and separate RGB colour properties for pigments and medium filters. A notable amount of research effort has been made in modelling colours in spectral representations, including:

- The investigation of the possible errors that can occur when using the RGB colour model in the calculation of colours [Hal89];
- The transformation of an RGB value to a spectrum [Gla89];

- The management of full spectral rendering in a linear manner [Pee93]; and
- The review of alternative colour representation methods other than the RGB colour model for accurate colour image synthesis [JF99].

The Kubelka-Munk theory [Sch05, KM31] was successfully deployed in computer graphics for modelling colour mixing [Fis83, HM92], rendering metallic paints [DH96], simulating watercolour [CAS$^+$97], simulating wax crayon [RMN03, RMN05] and simulating paints in a painting system [BWL04]. Recently, the Kubelka-Munk theory has been applied in frequency space for compositing multiple layers of translucent materials [DJ05]. In addition, other physically-based colour theories were experimented within computer graphics, including rendering diamonds [SFD99] and gemstones [GS04] using a combination of Fresnel's law and Bouguer-Beer's law [WS82].

Since the early development of direct volume rendering [Lev88, DCH88, Sab88], the RGBα model has been the dominant colour model used in volume visualisation for synthesising images involving translucent objects.

Spectral volume rendering was first proposed by Noordmans *et al.* [NvS00], who separated the material into an absorbing medium and scattering particles. They demonstrated that a spectral volume renderer produced more physically realistic images in comparison with the RGBα model. The spectral volume rendering integral used by Noordmans *et al.* involves an *illumination* phase and a *radiation* phase. The former features absorption of illumination spectra by an absorbing medium, while the latter is a spectral extension of the back-to-front RGBα compositing.

Bergner *et al.* presented a spectral volume rendering system that facilitates post-illumination and interactive data exploration [BMD02, BMDF02, BMD04, BMTD05]. In their approach, reflectance is accumulated in a front-to-back manner, and both indirect light absorption and reflectance transmittance are computed using a discretised extinction integral. When illumination is integrated into the accumulation process, the physical model of this rendering integral is fundamentally the same as that of the commonly-known volume rendering integral [KV84, DCH88, Max95]. The post-illumination and interactive data exploration presented by Bergner *et al.* focuses on the theory of colour vision, and the authors have applied the concepts of metamerism and colour constancy in their system design.

The main difference between our work and the previous spectral volume rendering efforts is the use of a volume rendering integral based on the Kubelka-Munk theory, which is widely used in industry to predict colour matches. In this chapter, we will also show that a volume rendering integral based on the Kubelka-Munk theory provides a discrete sampling process with a higher degree of consistency and optical correctness, in comparison to the existing approaches by Noordmans *et al.* and Bergner *et al.*, respectively.

4.3 The Kubelka-Munk Theory

The Kubelka-Munk theory [KM31] is concerned with the total diffuse radiation from a material in terms of its *absorption* (K) and *scattering* (S) properties. The basic considerations underlying this theory were first developed by Schuster in 1905 [Sch05]. This theory is

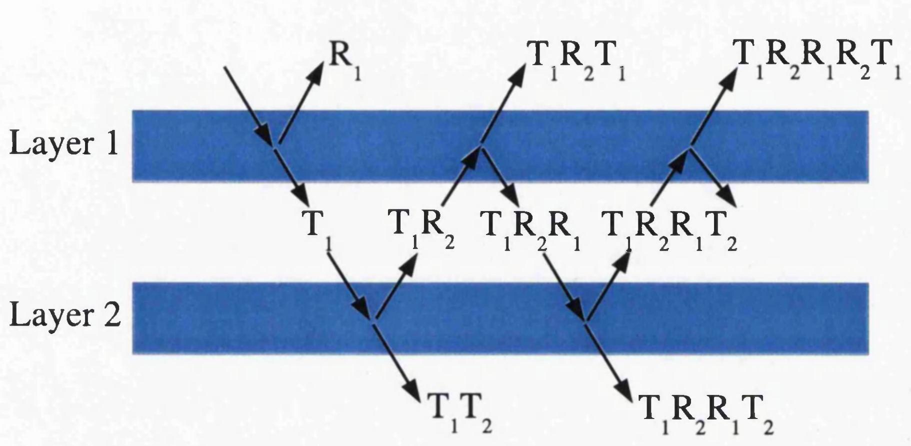


Figure 4.1: The overall reflection and transmittance for two layers, shown in terms of the reflection and transmittance of each individual layer [Kub54].

often used for translucent materials, transparent coloured layers over an opaque, scattering support material and also on opaque materials. The Kubelka-Munk theory is currently used extensively in the printing, coating and textile industries. Fundamentally it is a two-flux theory, which considers two radiation fluxes, namely *reflectance* (R) and *transmittance* (T), that pass through a continuous medium in two opposite directions. The medium is typically composed of multiple layers of paint-like turbid substances, and each layer is assumed to be homogeneous and completely diffuse.

For a single layer [Kub48] of thickness x, R and T relate to K and S as:

$$R = \frac{\sinh(bSx)}{a\sinh(bSx) + b\cosh(bSx)} \tag{4.1}$$

$$T = \frac{b}{a\sinh(bSx) + b\cosh(bSx)} \tag{4.2}$$

where

$$a = \frac{S+K}{S}, \quad b = \sqrt{a^2-1}.$$

Given the reflectance and transmittance of two different layers, R_1, T_1, R_2 and T_2, the composition of the two layers [Kub54] (see Figure 4.1) can be expressed as:

$$R_{1,2} = R_1 + T_1R_2T_1 + T_1R_2R_1R_2T_1 + \dots$$
$$T_{1,2} = T_1T_2 + T_1R_2R_1T_2 + T_1R_2R_1R_2R_1T_2 + \dots.$$

and can be simplified to (since $|R_1R_2| \leq 1$):

$$R_{1,2} = R_1 + \frac{T_1^2R_2}{1-R_1R_2} \tag{4.3}$$

$$T_{1,2} = \frac{T_1T_2}{1-R_1R_2} \tag{4.4}$$

The calculation for the reflectance and transmittance of n layers can be carried out by iterating this compositing operation, sequentially merging two layers into a single layer. The order of composition is irrelevant as it can be performed in a front-to-back or back-to-front manner, such that $R_1R_2R_3 = R_3R_2R_1$ and $T_1T_2T_3 = T_3T_2T_1$ [Kub54].

For an opaque medium, the reflectance is calculated by considering it as an infinitely thick layer [KM31], that is:

$$R_\infty = 1 + \left(\frac{K}{S}\right) - \left[\left(\frac{K}{S}\right)^2 + 2\left(\frac{K}{S}\right)\right]^{\frac{1}{2}}. \tag{4.5}$$

The inverse of the above equation gives:

$$\left(\frac{K}{S}\right) = \frac{(1-R_\infty)^2}{2R_\infty} \tag{4.6}$$

which shows that the reflectance of an opaque medium, R_∞, relates only to the ratio of K and S. Equation 4.6 is commonly used to estimate the absorption or scattering properties of a material from measured reflectance data. An alternative method for estimating the absorption and scattering properties of a material is to determine the reflectance of a thick layer. Using that layer as a background layer, a reflectance layer of known thickness can be estimated. Reviews of different methods for determining the absorption and scattering properties of a material for industry and computer graphics can be found in Nobbs [Nob97] and Haase and Meyer [HM92], respectively.

The Kubelka-Munk model can be applied to multiple layers of translucent and opaque materials. The effect of visually perceived colour that has been mixed using different layers, corresponding to one-dimensional volumetric radiosity [DH96].

In order to produce an accurate approximation of the Kubelka-Munk model, several assumption, such as the type of illumination used, the property of the material and disregarding any interactions at the material's boundary need to be taken into account. Curtis *et al.* [CAS$^+$97] and Baxter *at al.* [BWL04] demonstrated that even though the assumptions are only partially satisfied, the results produced by the Kubelka-Munk model are of close approximation.

A significant amount of research has been carried out on the Kubelka-Munk model, and an example is the use of the Fresnel's and Saunderson's [Sau42] equations in an attempt to take into consideration the partial reflection of light at the air-to-layer boundary. Given Fresnel's equation of reflection, the front-surface reflectance of the material, k_1, is:

$$k_1 = \left(\frac{n_2 - n_1}{n_2 + n_1}\right)^2 \tag{4.7}$$

where n_1 and n_2 are the refraction indexes of the external and internal material, respectively. The reflectance, R_a (*i.e.,* the adjusted reflectance in estimating the absorption and scattering property of the material), is calculated using Saunderson's equation as follows:

$$R_a = k_1 + \frac{R_m - k_1}{1 - k_1 - k_2 + k_2R_m} \tag{4.8}$$

where R_m is the result of a measured reflectance using a spectrophotometer and k_2 is the internal reflectance of the material.

4.4 Spectral Volume Modelling

Let $\mathbb{R}$ denote the set of all real numbers, and $\mathbb{E}^3$ denote 3D Euclidean space. A *scalar field* is a function $F : \mathbb{E}^3 \rightarrow \mathbb{R}$. The value range of a scalar field F is *bounded* if there are $m, n \in \mathbb{R}$, $m \leq n$ such that for all $p \in \mathbb{E}^3$, $m \leq F(p) \leq n$. In other words, $F : \mathbb{E}^3 \rightarrow [m, n]$.

A *volume data set* is a discrete specification of a scalar field, which consists of up to three essential components: a set of samples $\mathbf{V}$, their topological relationships $\mathbf{T}$ and an interpolation function $\mathbf{I}$. The set $\mathbf{V} = \{(p_i, v_i) \mid i = 1, 2, \ldots, n\}$ defines the known value v_i at each sampling location p_i in $\mathbb{E}^3$. Some volume data sets (*e.g.*, regular 3D grids) can be represented simply using a 3D array of values with implicit geometry and topology, whilst others (*e.g.*, curvilinear grids and tetrahedral meshes) may require an explicit specification of sample positions, $\{p_i \mid i = 1, 2, \ldots, n\}$, or topological connectivities between the samples, $\mathbf{T}$. The role of the interpolation function $\mathbf{I}$ is to define the scalar values at all the points in $\mathbb{E}^3$ other than those sample positions $\{p_i \mid i = 1, 2, \ldots, n\}$. Typical interpolation functions include trilinear interpolation for regular grids, and bary-centric interpolation for tetrahedral meshes.

Objects built from volume datasets are a subclass of spatial objects. The concepts introduced in this chapter can be applied to objects defined upon volume datasets, those specified mathematically or procedurally, or a combination of both. Our discussions are supported with objects built from volume datasets as well as those specified procedurally.

Spatial Signature	$\Sigma_{\text{RGB}\alpha}$ = RGBα model
Space	Euclid
Attributes	opacity, red, green, blue
Fields	Opacity: Euclid $\rightarrow$ opacity Red: Euclid $\rightarrow$ red Green: Euclid $\rightarrow$ green Blue: Euclid $\rightarrow$ blue

Table 4.1: Spatial signature of the RGBα model [CT98].

In traditional volume modelling [CT00], a volumetric representation of an object, that is, a *spatial object*, is a tuple, $\mathbf{o} = (A_0, A_1, \ldots, A_k)$, of *attribute fields* defined in $\mathbb{E}^3$. For example, in the RGBα model, an object is represented as $\mathbf{o} = (O, R, G, B)$, where $O, R, G, B : \mathbb{E}^3 \rightarrow [0, 1]$ are scalar fields for specifying opacity and colour attributes of $\mathbf{o}$. Table 4.1 demonstrates the *spatial signature* [CT98] of the RGBα model in the algebraic framework of CVG.

To accommodate the specification of absorption and scattering properties of a spatial object, the above notion is extended to include attribute fields defined in the form of vector fields. A *vector field* is a function $V : \mathbb{E}^3 \rightarrow \mathbb{R}^k$ where $k \in \mathbb{N}$ is the dimension of the vector field. Similarly, the value range of a vector field V can be bounded by $m, n \in \mathbb{R}$, $m \leq n$, such that for all $p \in \mathbb{E}^3$, $m \leq V_i(p) \leq n, \forall i : 1 \leq i \leq k$.

We normally approximate a continuous spectrum by using a vector, typically sampled at a selection of visible wavelengths. A spatial object based on the Kubelka-Munk model of

Spatial Signature	Σ_{km} = Kubelka-Munk model
Space	Euclid
Attributes	reflectance, transmittance, absorption, scattering
Fields	Reflectance: Euclid $\rightarrow$ reflectance Transmittance: Euclid $\rightarrow$ transmittance Absorption: Euclid $\rightarrow$ absorption Scattering: Euclid $\rightarrow$ scattering

Table 4.2: Spatial signature of the Kubelka-Munk model.

absorption and scattering is hence a tuple $\mathbf{o} = (R, T, K, S)$, where $R, T, K, S : \mathbb{E}^3 \rightarrow (0, \infty)^k$ are vector fields for specifying the physical properties of the material, such as the reflectance, transmittance, absorption and scattering attributes of the object. It is worth noting that R and T relate to K and S as demonstrated in Equations 4.1 and 4.2. Table 4.2 defines the spatial signature of the Kubelka-Munk model in the algebraic framework of CVG. In our implementation, we have chosen $k = 31$, allowing each spectrum to be sampled from $400nm$ to $700nm$ at $10nm$ intervals. In some cases, we also involved a refraction index field, R_{fr}, which is a scalar field $R_{fr} : \mathbb{E}^3 \rightarrow [1, 3.5]$ [RC01].

4.5 Spectral Volume Rendering

Ray marching has been used to evaluate fields in previous work on direct volume rendering [Lev88]. The basic mechanism is to sample at regular intervals along each ray cast from the viewing position. At each sampling point q, we determine the absorption and scattering properties of the spatial object concerned.

Given a series of samples, $q_1, q_2, \ldots, q_n$, along a ray cast from the viewing position at a regular interval δ, we approximate the intersection volume between the ray and the spatial object as a series of homogeneous layers of thickness δ. The Kubelka-Munk theory can thereby be extrapolated to multiple homogeneous layers of an inhomogeneous volume. Given the reflectance and transmittance for each layer computed from Equations 4.1 and 4.2 as R_i and T_i, $i = 1, 2, \ldots, n$, we accumulate the sum of reflectance $\mathbf{R}$ and that of transmittance $\mathbf{T}$ in the two respective fluxes as:

$$\begin{aligned}
\mathbf{R}_1(\lambda) &= R_1(\lambda) \\
\mathbf{T}_1(\lambda) &= T_1(\lambda) \\
&\ldots \\
\mathbf{R}_i(\lambda) &= \mathbf{R}_{i-1}(\lambda) + \frac{\mathbf{T}_{i-1}(\lambda)^2 R_i(\lambda)}{1 - \mathbf{R}_{i-1}(\lambda) R_i(\lambda)} \\
\mathbf{T}_i(\lambda) &= \frac{\mathbf{T}_{i-1}(\lambda) T_i(\lambda)}{1 - \mathbf{R}_{i-1}(\lambda) R_i(\lambda)}
\end{aligned}$$

where $i = 2, \ldots, n$ and λ is a wavelength, which is used here to emphasise the discrete vector representation of the spectra. This process of accumulating $\mathbf{R}$ and $\mathbf{T}$ facilitates a *spectral volume rendering integral*, which differs from the conventional opacity-based volume rendering integral in terms of its physical basis.

Assume that we have an opaque background, which is defined as a reflectance map. When a ray exits the spatial object, $\mathbf{R}_n$ and $\mathbf{T}_n$ are further integrated with the background reflectance R_{bg} as:

$$\mathbf{R}_{final}(\lambda) = \mathbf{R}_n(\lambda) + \frac{\mathbf{T}_n(\lambda)^2 R_{bg}(\lambda)}{1 - \mathbf{R}_n(\lambda) R_{bg}(\lambda)}$$

Obviously we can also define the background using an absorption map and a scattering map, and compute the reflectance map using Equation 4.5.

In fact, when the sampling distance $\delta \to 0$, the approximation error, due to the assumption of homogeneous layers, will be alleviated. Hence, we equip our rendering integral with two global parameters, Δ: the *standard thickness* in the world coordinate system based on which K and S are defined; and δ: the *actual sampling distance*, also in the world coordinate system. The calculations of Equations 4.1 and 4.2 are based on $x = \delta/\Delta$. Note that K and S are normally defined upon a unit thickness. Hence a uniform geometrical scaling of a spatial object in all dimensions must be accompanied by a modification of Δ. The change of δ facilitates depth-supersampling in volume rendering as shown in Figure 4.2. The necessary correction for depth-supersampling is in itself less expensive than that in the conventional volume rendering integral [CT00].

Sampling Distance	**Image**	`ppmdiff`
$\Delta = 1, \delta = 0.5$	Figure 4.2(a)	0.085056
$\Delta = 1, \delta = 0.05$	Figure 4.2(b)	0.030219
$\Delta = 1, \delta = 0.005$	Figure 4.2(c)	0.010435

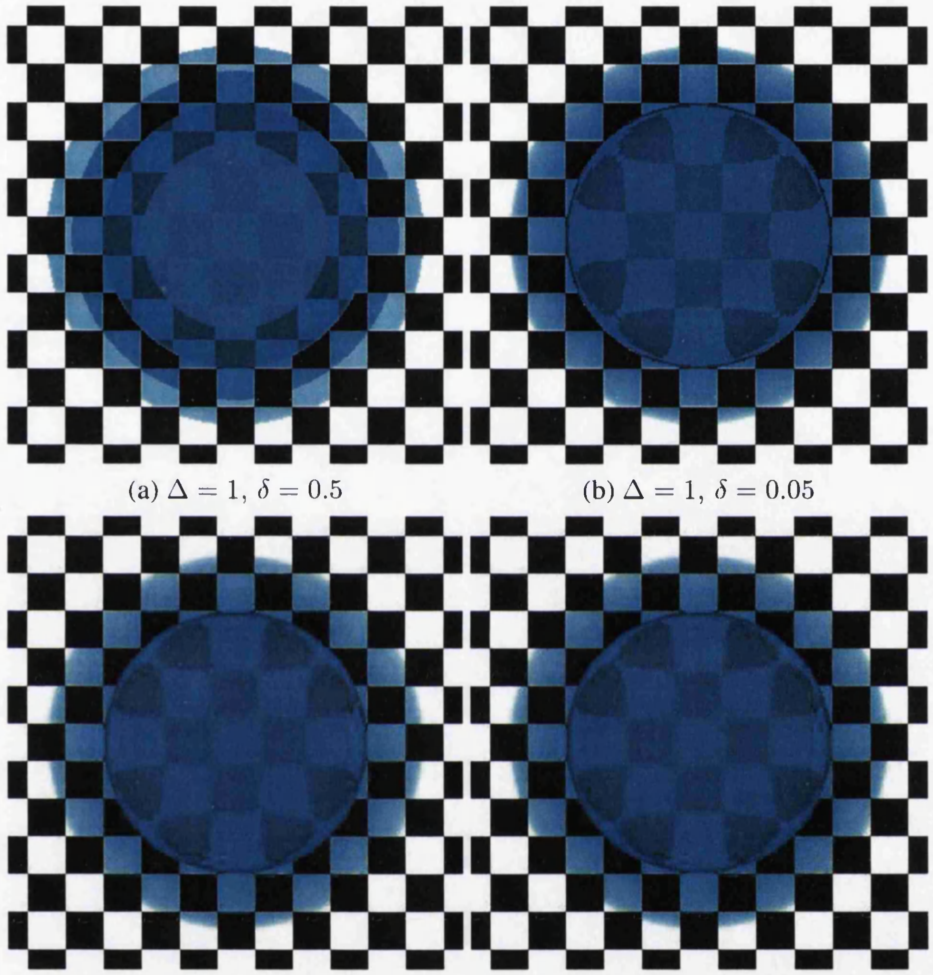
Table 4.3: Quantitative comparison of difference sampling distances.

Note that the spectral volume rendering integral based on the Kubelka-Munk theory produces only an 'image' in the form of a *reflectance map*. To synthesise a visual image, the reflectance map needs to be lit with an appropriate light source, in a *post-illumination* stage, which will be further discussed in Section 4.8.

4.6 Sampling Correctness

Discrete sampling is the most commonly adopted and effective means for implementing a volume rendering integral. The correctness of such a process is a fundamental property of a volume rendering engine.

Consider a very basic case of sampling a homogeneous medium with constant optical properties as illustrated in Figure 4.3. The background is set to black, that is, $R_{bg}(\lambda) = 0$. Given n sampling points, $q_1, q_2, \ldots, q_n$, with a uniform sampling interval $\delta_n = X/(n-1)$, $\mathbf{R}^{<n,\delta_n>}$ denotes the final reflectance obtained from discrete sampling. We thereby have the following theorem, which can be proved by induction.



(a) $\Delta = 1,\ \delta = 0.5$

(b) $\Delta = 1,\ \delta = 0.05$

(c) $\Delta = 1,\ \delta = 0.005$

(d) $\Delta = 1,\ \delta = 0.0005$

Figure 4.2: A translucent sphere (radius = 1) with an amorphous layer (refraction index = 1) and a solid layer (refraction index = 1.03), rendered with different sampling distances. It is nontrivial for the naked eye to distinguish (b), (c) and (d), where most sampling errors are in fact attributed to refraction rather than the accumulation of reflectance and transmittance. Table 4.3 compares the results of applying the different sampling distances to the translucent sphere against the reference image in (d). The quantitative comparisons are generated using a `ppmdiff` program, which compares two images in the ppm format.

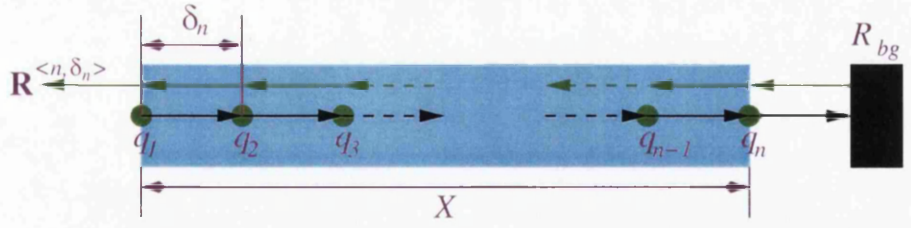


Figure 4.3: Discrete sampling of a uniform medium of homogeneous layers.

Theorem. For any $n \geq 2$ and $\delta_n = X/(n-1)$, the reflectance computed using the volume rendering integral based on the Kubelka-Munk theory satisfies the following equation:

$$\mathbf{R}^{<2,\delta_2>} = \mathbf{R}^{<3,\delta_3>} = \ldots = \mathbf{R}^{<n-1,\delta_{n-1}>} = \mathbf{R}^{<n,\delta_n>}.$$

Proof. By induction on n stacks of thickness X/m, then reflectance

$$R_m^n = \frac{\sinh(nbS\frac{X}{m})}{a\sinh(nbS\frac{X}{m}) + b\cosh(nbS\frac{X}{m})}.$$

Basis. For the base case, it is noted that

$$R_m^1 = \frac{\sinh(bS\frac{X}{m})}{a\sinh(bS\frac{X}{m}) + b\cosh(bS\frac{X}{m})}.$$

Induction. Now let $c_n = \cosh(nbS\frac{X}{m})$ and $s_n = \sinh(nbS\frac{X}{m})$. Assume that the formula for R_m^n (IH) is correct then

$$R^{n+1} = R^1 + \frac{\frac{b^2}{(as_1+bc_1)^2}R^n}{1 - R^1R^n}$$

$$\begin{aligned}
R^{n+1} &= R^1 + \frac{\frac{b^2}{(as_1+bc_1)^2}R^n}{1 - R^1R^n} \\
&= \frac{\frac{s_1}{as_1+bc_1} + \frac{b^2 s_n}{(as_1+bc_1)^2(as_n+bc_n}}{1 - \left(\frac{s_1}{(as_1+bc_1)}\right)\left(\frac{s_n}{(as_n+bc_n)}\right)} \qquad \text{by (IH).} \\
&= \frac{s_1}{as_1+bc_1} + \frac{b^2 s_n}{(as_1+bc_1)((as_1+bc_1)(as_n+bc_n) - s_1s_n)} \\
&= \frac{1}{as_1+bc_1}\left(s_1 + \frac{b^2 s_n}{(a^2s_1s_n + b^2c_1c_n + abc_1s_n + abs_1c_n - s_1s_n)}\right) \\
&= \frac{1}{as_1+bc_1}\left(s_1 + \frac{b^2 s_n}{b^2(s_1s_n + c_1c_n) + ab(c_1s_n + s_1c_n)}\right)
\end{aligned}$$

But $s_1s_n+c_1c_n = c_{n+1}$ by $\cosh(\alpha+\beta) = \cosh\alpha\cosh\beta+\sinh\alpha\sinh\beta$ and $c_1s_n+c_ns_1 = s_{n+1}$ using $\sinh(\alpha+\beta) = \sinh\alpha\cosh\beta+\sinh\beta\cosh\alpha$. So

$$\begin{aligned}R^{n+1} &= \frac{1}{as_1+bc_1}\left(s_1+\frac{bs_n}{as_{n+1}+bc_{n+1}}\right)\\ &= \frac{1}{as_1+bc_1}\left(\frac{as_1s_{n+1}+bs_1c_{n+1}+bs_n}{as_{n+1}+bc_{n+1}}\right)\\ &= \frac{1}{as_1+bc_1}\left(\frac{(as_1+bc_1)s_{n+1}-bc_1s_{n+1}+bs_1c_{n+1}+bs_n}{as_{n+1}+bc_{n+1}}\right)\end{aligned}$$

But $s_{n+1}c_1 - c_{n+1}s_1 = s_n$ since $\sinh(\alpha-\beta) = \sinh\alpha\cosh\beta - \cosh\alpha\sinh\beta$. So

$$\begin{aligned}R^{n+1} &= \left(\frac{1}{as_1+bc_1}\right)\left(\frac{(as_1+bc_1)s_{n+1}}{as_{n+1}+bc_{n+1}}\right)\\ &= \frac{s_{n+1}}{as_{n+1}+bc_{n+1}} \qquad \text{as required.}\end{aligned}$$

□

Corollary. $\mathbf{R}^{<n,\delta_n>} = \mathbf{R}^{<n-1,\delta_{n-1}>} = \frac{\sinh(bSX)}{a\sinh(bSX)+b\cosh(bSX)}$ *as required.*

In contrast, the spectral volume rendering integrals proposed by Noordmans *et al.* and Bergner *et al.* do not possess this useful property. For example, consider a simple front-lit case in the context of Noordmans *et al.* [NvS00]. Assuming that the reflectance R and opacity α are uniformly defined in a medium in a way similar to Figure 4.3, the final colour spectrum $I^{<n,\delta_n>}$ is obtained as follows,

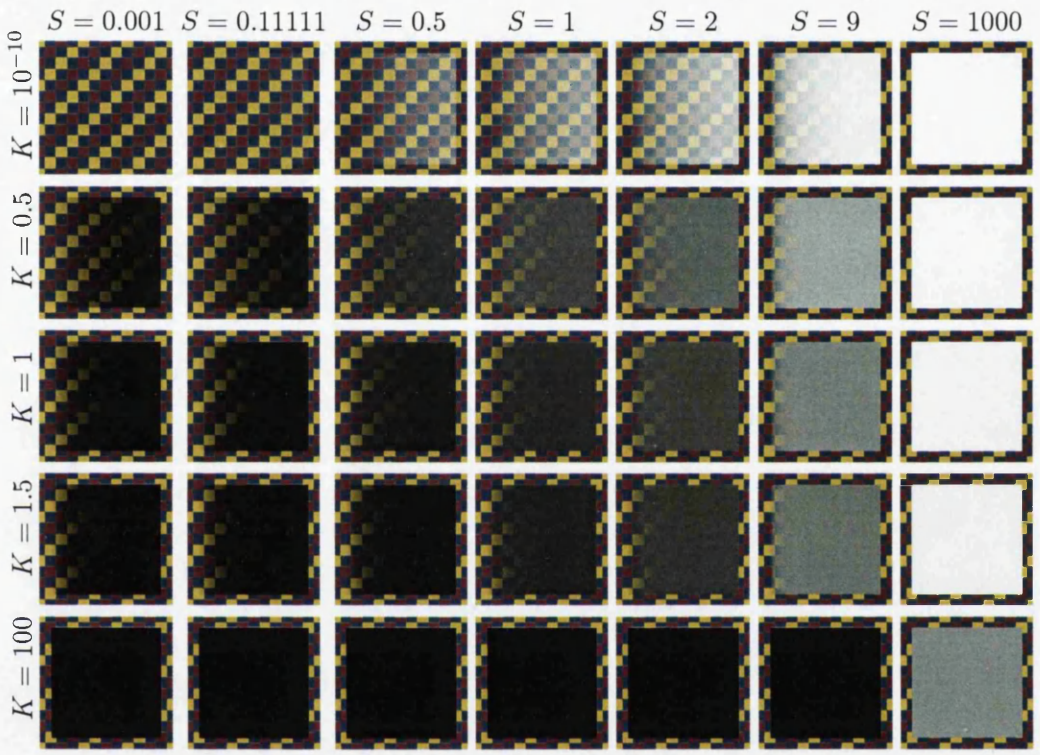
$$\begin{aligned}I(\lambda)^{<n,\delta_n>} &= E_0(\lambda)+E_1(\lambda)T(\lambda)+\ldots+E_n(\lambda)T(\lambda)^n\\ &= R(\lambda)L(\lambda)(1+T(\lambda)^2+T(\lambda)^4+\ldots+T(\lambda)^{2n})\end{aligned}$$

where L is an external light source, local emission $E_i(\lambda) = R(\lambda)L(\lambda)$ and local transmittance $T(\lambda) = (1-\delta_n\alpha(\lambda))$. We can easily confirm the sampling inconsistency by showing that $I(\lambda)^{<n-1,\delta_{n-1}>} \neq I(\lambda)^{<n,\delta_n>}$. We also know that when $n \to \infty$, $I(\lambda)^{<n,\delta_n>}$ is divergent, though this can be corrected by defining $E_i(\lambda) = \delta_nR(\lambda)L(\lambda)$, $i \in [0,n]$ instead.

Similarly, consider a simple local illumination case in the context of Bergner *et al.* [BMDF02]. Assume that the basic reflectance R and local absorptivity (or extinction coefficient) τ are uniformly defined in the medium, and the contribution of indirect lighting $S^L(\lambda)$ is set to 1. The final colour spectrum $C^{<n,\delta_n>}$ is obtained as follows,

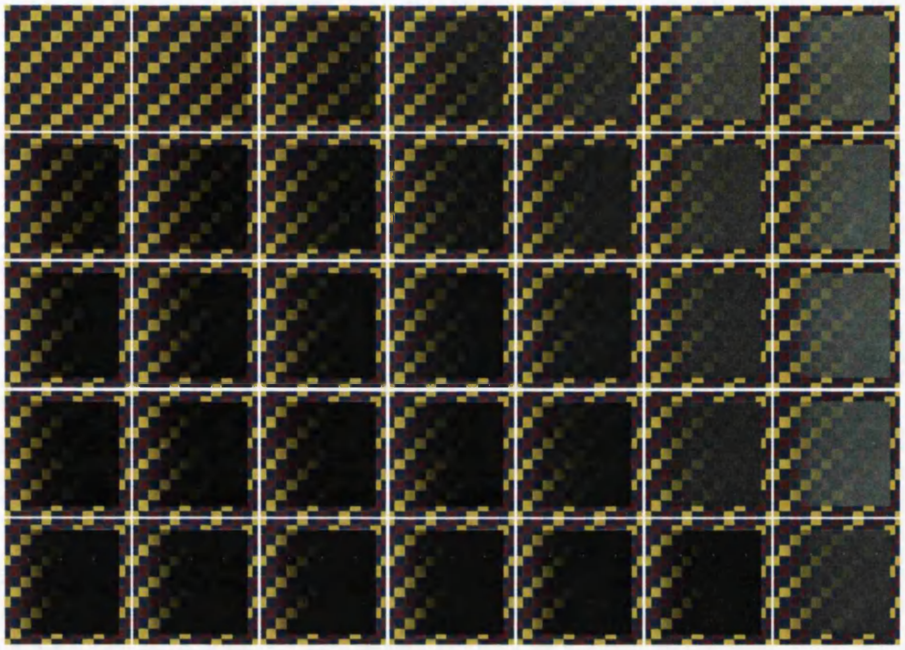
$$\begin{aligned}C(\lambda)^{<n,\delta_n>} &= L(\lambda)R(\lambda)\delta_n(T_0(\lambda)+T_1(\lambda)+\ldots+T_n(\lambda))\\ &= L(\lambda)R(\lambda)\delta_n(1+A(\lambda)+A(\lambda)^2+\ldots+A(\lambda)^n)\end{aligned}$$

where $A(\lambda) = e^{-\delta_n\tau(\lambda)}$. The approach also suffers from inconsistent discrete sampling with a lower sampling rate since $C(\lambda)^{<n-1,\delta_{n-1}>} \neq C(\lambda)^{<n,\delta_n>}$. However, in this case, when $n \to \infty$, $C(\lambda)^{<n,\delta_n>}$ does converge.



Visual effects produced based on the Kubelka-Munk model

Figure 4.4: Effects of varying absorption (K), scattering (S) and thickness (x). In each prism, x changes from 0 (left) to 2 (right). Artificial spectra of K and S with a constant value for all wavelengths are used to minimise the chromatic effects in the visual appearance resulting from different reflectance (R) and transmittance (T). All prisms are lit by the same white light source.


(a) based on the rendering integral by Noordmans *et al.*

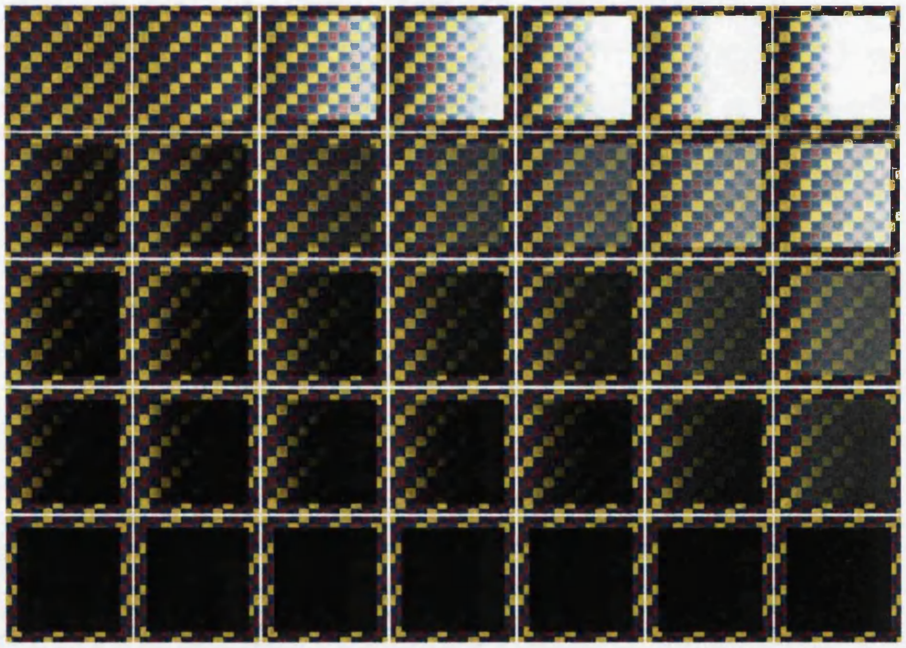
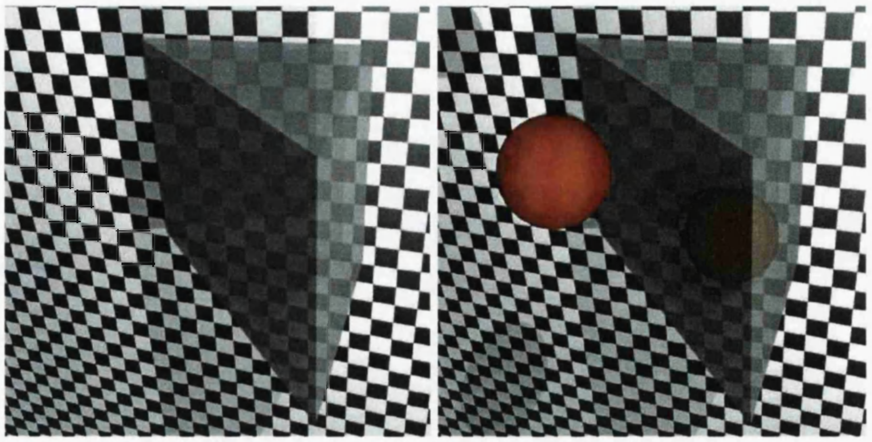

(b) based on the rendering integral by Bergner *et al.*

Figure 4.5: Effects of varying absorption (K), scattering (S) and thickness (x). For (a), we compute the local reflectance R_{KM} and transmittance T_{KM} using K and S in the same way as in the Kubelka-Munk model, and set the local properties defined in Noordmans *et al.* [NvS00] as $R_{NVS} = R_{KM}$ and $\alpha_{NVS} = 1 - T_{KM}$. For (b), we set the basic reflectance defined in Bergner *et al.* [BMDF02] as $R_{BMD} = R_{KM}$, and compute the local absorption by setting $\tau(\lambda) = 0.5K(\lambda)\ln 10$.



(a) a prism (b) a prism and two spheres

Figure 4.6: The geometrical specification of the objects used in Figures 4.4, 4.5, 4.7 and 4.8. To help visualise the geometrical shapes of the objects, the refraction index of the prism is set to 1.03 and that of the spheres to 1.05. The base of the prisms in both (a) and (b) is a right-angle isosceles triangle with adjacent edges of length 2. Both spheres in (b) are of a radius 0.4.

4.7 Visual Effects and Correctness

The conventional volume rendering integral based on the RGBα model provides an empirical mechanism for visualising volume datasets with translucent optical effects. The spectral volume rendering based on the Kubelka-Munk theory determines the opacity and transparency optically according to the internal absorption and scattering properties of the spatial object. The visual effect of opaqueness is caused by a combination of strong scattering and some absorption. The visual effect of translucency results from some scattering and little absorption. Figure 4.4 shows visual effects of a prism controlled by its absorption and scattering properties. With a prism as illustrated in Figure 4.6(a), the effects of changing the thickness x of a medium can be observed.

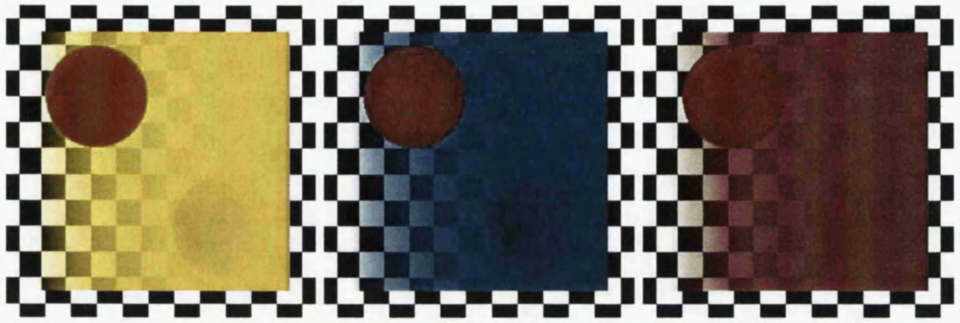
In Figure 4.4, as scattering S increases from 0.001 to 1000, the object becomes more opaque and reflects more light. As absorption K increases from 10^{-10} to 100, the object largely maintains a similar opacity but transmits less light and has less internal reflectance. This in effect provides depth-cueing in an optically correct manner.

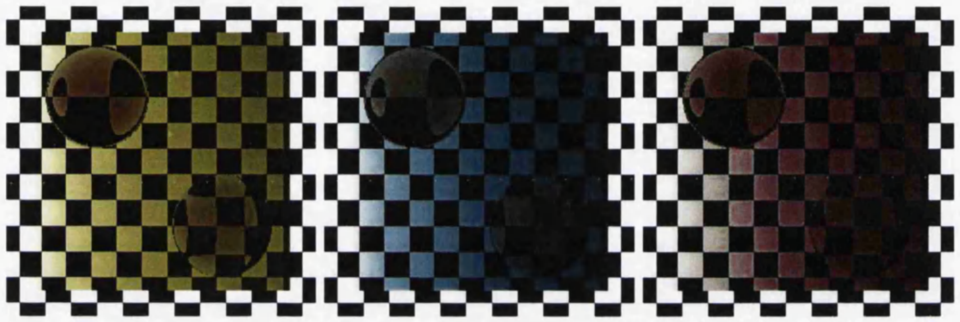
As a comparison, these effects have also been rendered using the spectral volume rendering integrals proposed by Noordmans *et al.* and Bergner *et al.*. In both cases, the prism was lit by a front light source. For the rendering integral by Noordmans *et al.*, we compute the local reflectance and local transmittance from K and S in the same way as in the Kubelka-Munk model, and obtain the local opacity from the local transmittance. For the integral by Bergner *et al.*, we again compute the local reflectance in the same way as in the Kubelka-Munk model, and map the absorption spectra defined in the Kubelka-Munk theory to the absorptivity defined in the Bouguer-Beer's law [WS82] with an appropriate adjustment from base 10 to base e. The results are shown in Figures 4.5(a) and (b) respectively. In comparison with Figure 4.4, we can easily observe the limitations of the two existing rendering integrals. The rendering integral by Noordmans *et al.* seems to have difficulties in handling materials with either high scattering values (*i.e.,* the last column) or high absorption values (*i.e.,* the last row). The rendering integral by Bergner *et al.* seems to have difficulties in handling materials with either high scattering values (*i.e.,* the last column) or low absorption values (*i.e.,* the first row). Both have difficulties in relating opacity to scattering along the direction of S.

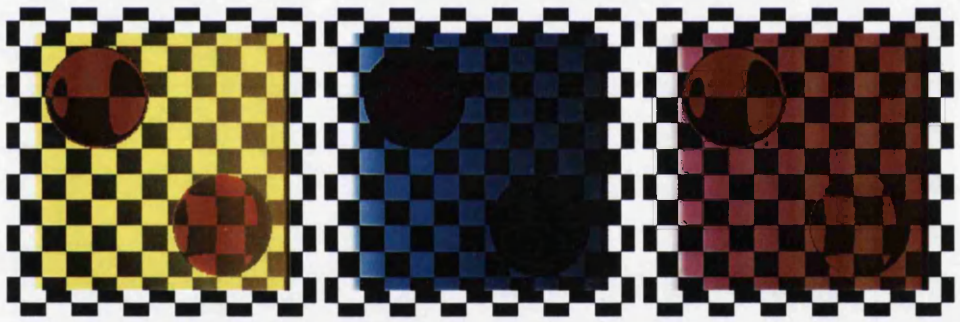
While the conventional volume rendering integral can produce translucent optical effects, it does not result in optically correct visualisation. It simulates a substance featuring *general absorption* [JW76] by reducing the intensity of RGB components of light equally. To demonstrate this, we use a scene consisting of a translucent prism and two translucent spheres as shown in Figure 4.6(b). One sphere is placed in front of the prism and the other is embedded inside the prism. The K and S for the properties of Figures 4.7(d) and 4.8(d) are estimated using Equation 4.6 based on the reflectance spectra provided in the MacBeth Color Checker [War04]. The RGB colours for Figures 4.7(a) and 4.8(a) are calculated based on the reflectance spectra and the opacity of the objects are manually estimated to achieve a reasonably good approximation to Figures 4.7(d) and 4.8(d). As shown in Figures 4.7 and 4.8, with the RGBα model, there is a similar amount of colour accumulated at the pixels which the two spheres were projected to, regardless of the colour of the prism. For example, the cyan prism in Figure 4.7(a) should have blocked more reflectance of the embedded red sphere than the yellow and magenta prisms, and the blue prism in Figure 4.8(a) should have blocked more reflectance of the embedded yellow sphere than the red and green prisms. This is optically incorrect because (i) such a substance can appear to be grey or nearly black under a white light, (ii) no substance is known to absorb all wavelengths equally.

The correct optical effects should feature *selective absorption* [JW76], which can be realised using the spectral volume rendering integral based on the Kubelka-Munk theory as shown in Figures 4.7 and 4.8. For the pixels associated with the front sphere, the spectral volume rendering integral shows correct accumulation of transmittance and reflectance by allowing the absorption of certain wavelengths. In Figure 4.7(d), for example, after passing through the red sphere, much of the remaining transmittance flux continues to pass through the yellow or magenta prism but not the cyan prism. This feature can also be observed in Figure 4.8(d) – after passing through the yellow sphere, much of the remaining transmittance flux continues to pass through the red or green prism but not the blue prism.

As a further comparison, we also have rendered this scene using the spectral volume rendering integrals proposed by Noordmans *et al.* [NvS00] and Bergner *et al.* [BMDF02]. The same method that is used in Figures 4.5(a) and (b) to compute the values for the rendering integrals by Noordmans *et al.* and Bergner *et al.* is applied here. The rendering integral proposed by Noordmans *et al.* exhibit the same general absorption feature as with the RGBα model, where a similar amount of colour is accumulated at the pixels which the two spheres were projected to regardless of the colour of the prism, as shown in Figures 4.7(b) and 4.8(b). Although the rendering integral by Bergner *et al.* does not display the feature of general absorption, it seems to have difficulties in modelling the feature of selective absorption. For example, the red spheres with the cyan prism in Figure 4.7(c) reflect the same amount of colour regardless of their positions in the scene. This feature can also be observed in Figure 4.8(c), for example, the yellow spheres with the red prism reflect almost


(a) RGBα: red spheres with yellow, cyan, magenta prisms


(b) NVS: red spheres with yellow, cyan, magenta prisms


(c) BMD: red spheres with yellow, cyan, magenta prisms

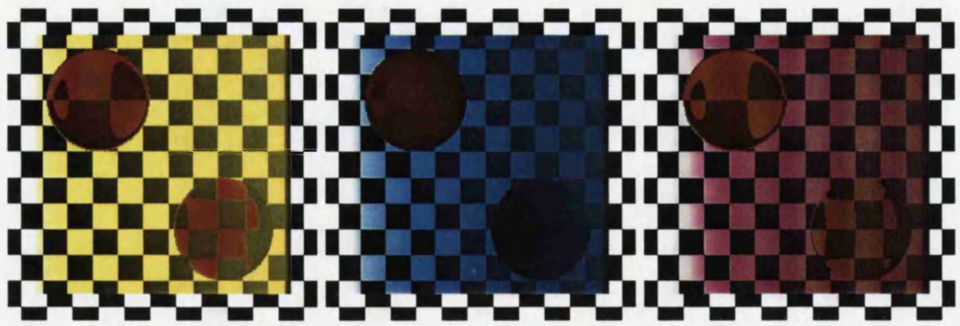
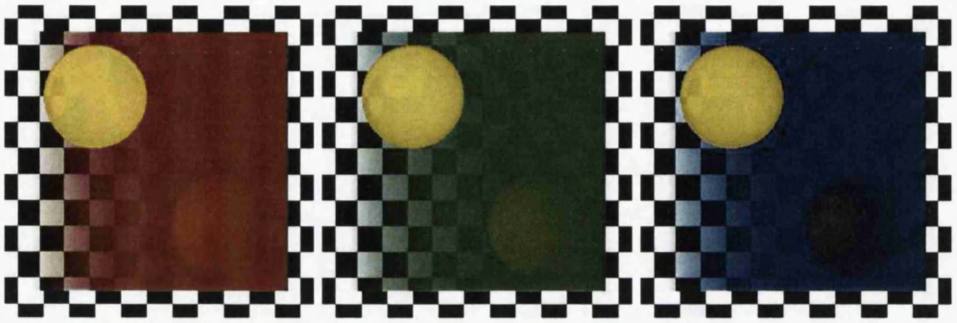
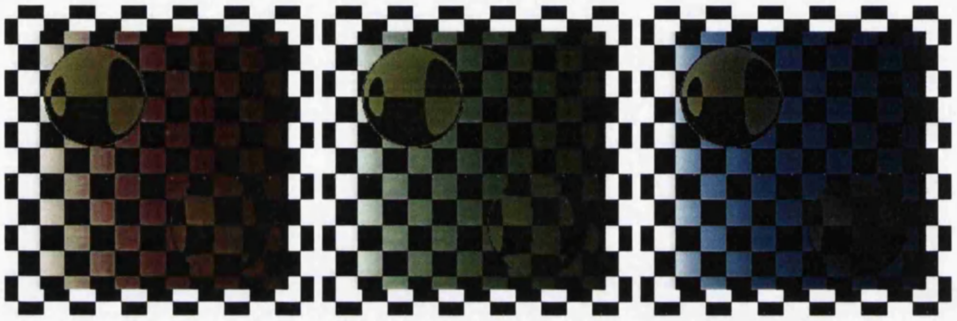

(d) KM: red spheres with yellow, cyan, magenta prisms

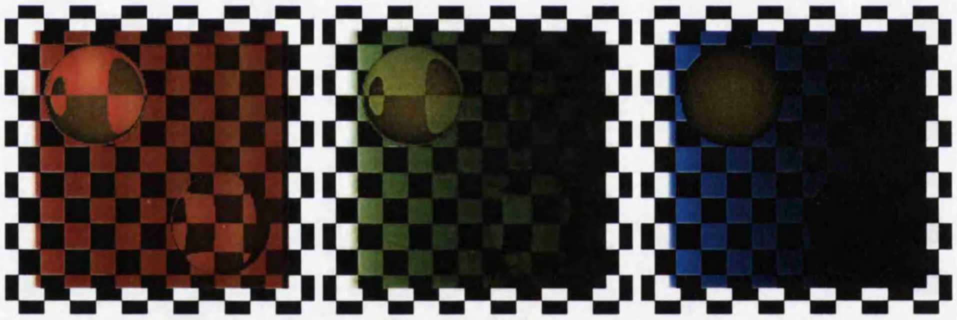
Figure 4.7: Colour accumulation in the RGBα model, the spectral volume rendering integral by Noordmans *et al.* [NvS00] and Bergner *et al.* [BMDF02] and reflectance accumulation in the Kubelka-Munk theory. The latter is shown to be more optically correct.



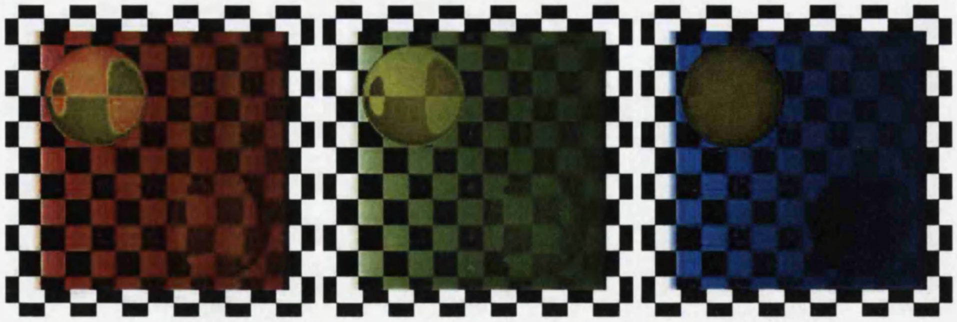
(a) RGBα: yellow spheres with red, green, blue prisms



(b) NVS: yellow spheres with red, green, blue prisms



(c) BMD: yellow spheres with red, green, blue prisms



(d) KM: yellow spheres with red, green, blue prisms

Figure 4.8: Colour accumulation in the RGBα model, the spectral volume rendering integral by Noordmans *et al.* [NvS00] and Bergner *et al.* [BMDF02] and reflectance accumulation in the Kubelka-Munk theory. The latter is shown to be more optically correct.

the same amount of colour regardless of their positions in the scene causing the front sphere to exhibit a more reddish colour than expected.

Figures 4.9 and 4.10 show the application of the spectral volume rendering based on the Kubelka-Munk theory to a number of example volume datasets. In Figures 4.9(a)-(d), the engine block dataset is rendered with K and S spectra estimated from natural reflectance data for aluminium, copper, gold and stainless steel. In Figures 4.9(e) and 4.10(a), the skin of the MRI-head and the foot is rendered with K and S spectra estimated from natural reflectance data for skin, while the interior of the MRI-head in Figure 4.9(f) is rendered with the standard colour spectra defined in the MacBeth Color Checker [War04], and the bone of the foot in Figure 4.10(a) is rendered with measured K and S of a material commonly used for thickening paint [PSS77]. Similarly, images in Figures 4.10(b)-(f) are rendered with a combination of measured and estimated absorption and scattering data. Most estimations were made based on some measured reflectance data using Equation 4.6 by setting scattering to an appropriate constant.

Given absorption and scattering attributes, K and S, of a material, we allow further control of the reflectance and transmittance of the material in the look-up table of our rendering engine. By scaling K and S with the same scaling factor, the principle reflectance of a material R_∞ remains unchanged. A carefully managed scaling factor (*e.g.,* using a fuzzy function [Lev88]) can facilitate the gradual increase and decrease of reflectance and transmittance of a material, which is similar to the design of a smoothly changed opacity transfer function in conventional volume rendering.

4.8 Post-illumination and Spectral Transfer Functions

As mentioned in Section 4.5, the spectral volume rendering integral based on the Kubelka-Munk theory computes an 'image' in the form of a *reflectance map*, which needs to be lit with a light source to produce a final image for display. Given the spectrum of a light source L, for each pixel in the final image, we first compute its colour in a spectral representation as

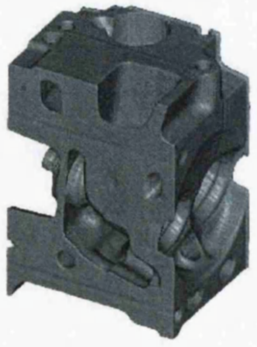
$$C_{sp}(\lambda) = \mathbf{R}_{\mathbf{final}}(\lambda)L(\lambda), \tag{4.9}$$

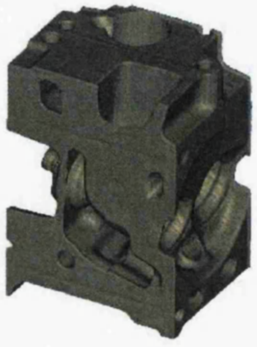
then convert it to an XYZ colour model using the 1931 CIE colour-matching functions, $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, $\bar{z}(\lambda)$, as:

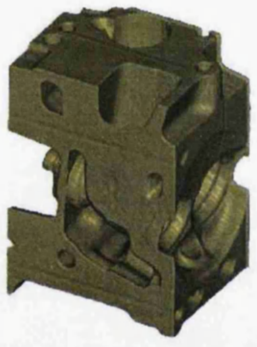
$$C_X = \int_\lambda C_{sp}(\lambda)\bar{x}(\lambda)\,d\lambda$$
$$C_Y = \int_\lambda C_{sp}(\lambda)\bar{y}(\lambda)\,d\lambda$$
$$C_Z = \int_\lambda C_{sp}(\lambda)\bar{z}(\lambda)\,d\lambda$$

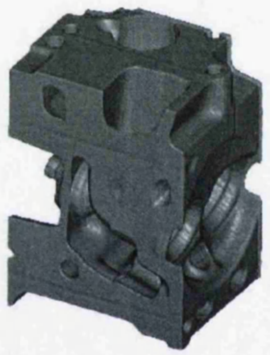
and finally to an RGB colour [Gla89] as:

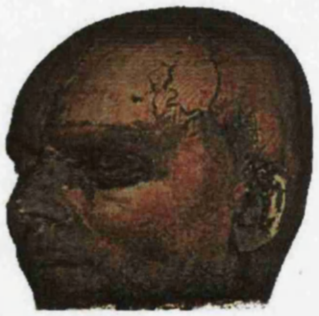
$$[C_R, C_G, C_B] = [C_X, C_Y, C_Z] \begin{bmatrix} 1.967 & -0.955 & 0.064 \\ -0.548 & 1.938 & -0.130 \\ -0.297 & -0.027 & 0.982 \end{bmatrix}$$


(a) aluminium engine


(b) copper engine


(c) gold engine


(d) stainless steel engine


(e) opaque MRI-head

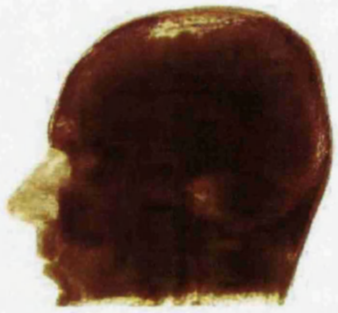
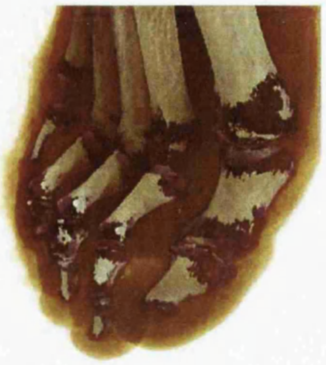
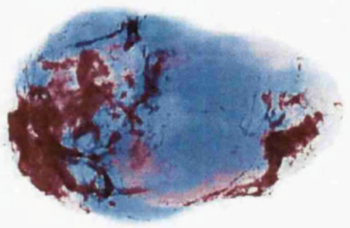
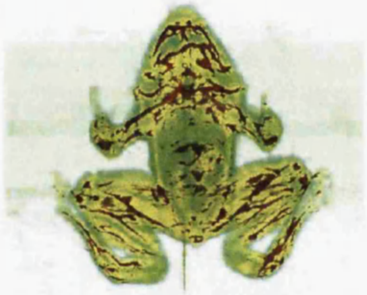
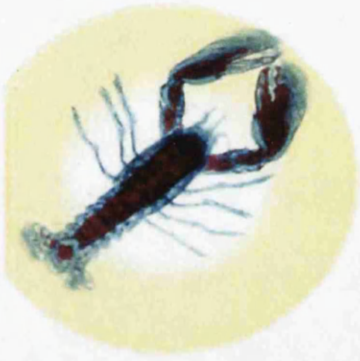

(f) translucent MRI-head

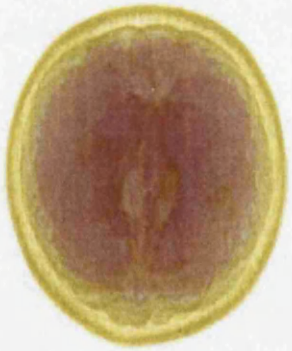
Figure 4.9: Volume datasets rendered using the spectral volume rendering integral based on the Kubelka-Munk theory with measured and estimated natural spectra.


(a) translucent foot


(b) translucent sheep heart


(c) translucent frog


(d) translucent lobster


(e) translucent orange

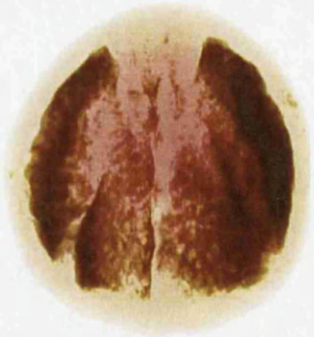

(f) translucent tomato

Figure 4.10: Volume datasets rendered using the spectral volume rendering integral based on the Kubelka-Munk theory with measured and estimated natural spectra.

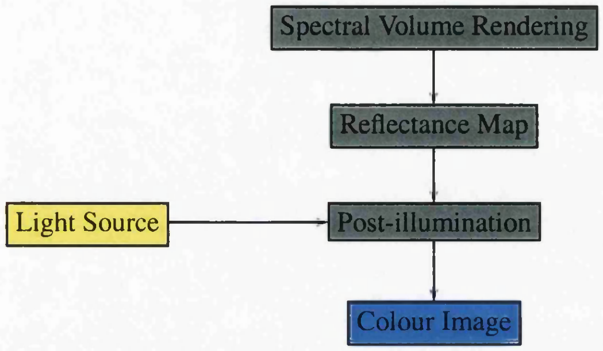


Figure 4.11: The post-illumination pipeline.

The separation of illumination from the volume rendering integral enables the change of light sources to be computed without re-computing the rendering integral, as illustrated in Figure 4.11. For an image with m pixels, and a light source with a k-dimensional vector representation for its spectrum, the post-illumination requires a total of $m(4k + 9)$ multiplications and $3m(k + 1)$ additions. With a modern PC, this can easily be computed at an interactive speed. For example, for a typical image resolution of $m = 400 \times 400$ and a relatively accurate discretisation of spectra with $k = 31$, one can achieve 0.12 seconds per frame on a Pentium 2GHz PC.

This has led to the idea of using post-illumination to interrogate volume datasets with different light sources. For an example, for exploratory visualisation, we can simply use the following continuous single-peak spectral transfer functions for mapping each scalar value v in a volume dataset to K and S. Without losing any generality, let $v \in [0, 1]$, which is mapped onto a visible wavelength λ_v as $\lambda_v = 300v + 400$. We assign spectra K and S as:

$$\phi = \frac{|\lambda_v - \lambda|}{w_{peak}}$$

$$\omega(\lambda) = \begin{cases} 0 & \phi \geq 1 \\ 0.5(1 - \cos(\pi(1 - \phi))) & \phi < 1 \end{cases}$$

$$K(\lambda) = \omega(\lambda)(K_{max} - K_{min}) + K_{min}$$

$$S(\lambda) = \omega(\lambda)(S_{max} - S_{min}) + S_{min}$$

where the peak is a reflected cosine function of a width w_{peak}, and K_{min}, K_{max}, S_{min} and S_{max} are user-definable constants specifying the highest and lowest values of absorption K and scattering S.

Figure 4.12 shows a selection of wavelengths in an example reflectance map computed using such spectral transfer functions. This reflectance map can be integrated interactively using different light sources. For example, using a single-peak light source of a similar shape to the spectral transfer functions for K and S can reveal the corresponding iso-surfaces defined at, and close to, the peak wavelength λ_v, as illustrated in Figure 4.13. Using a more complicated light source, one can visualise more complicated structures of a volume dataset. Figure 4.14 shows four such examples. A white light source in Figure 4.14(a) gives a global impression of the dataset, while light sources with multiple peaks in Figures 4.14(b), (c) and

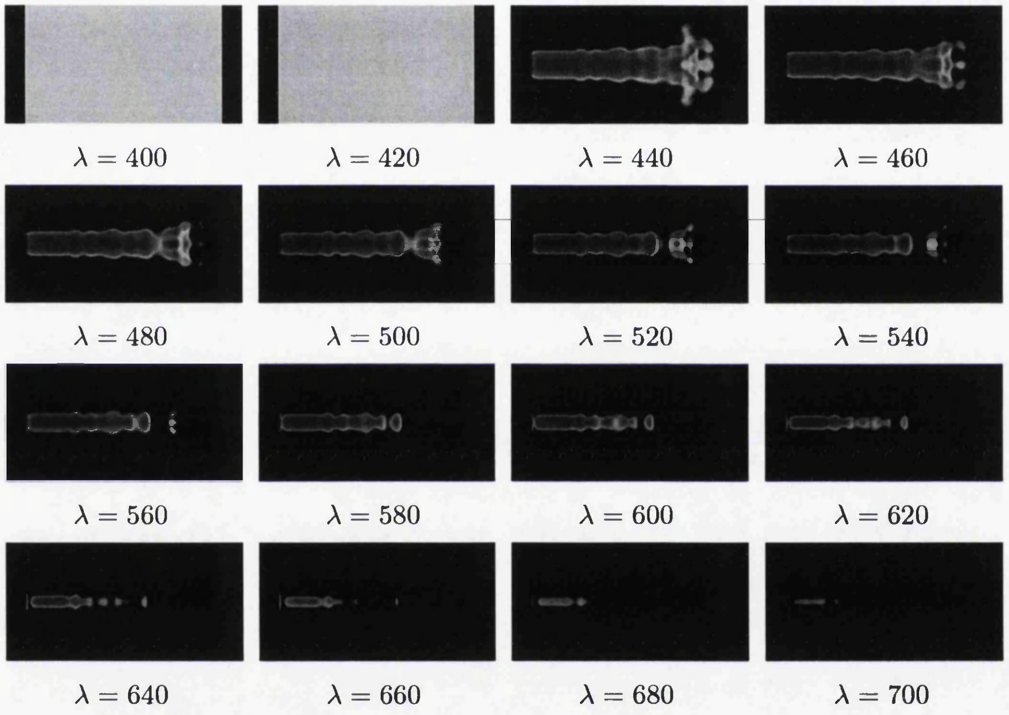


Figure 4.12: Selected wavelengths of a reflectance map, which is computed using single-peak spectral transfer functions for both K and S, with $K_{min} = 0.0001$, $K_{max} = 0.5$, $S_{min} = 0.0001$, $S_{max} = 10$.

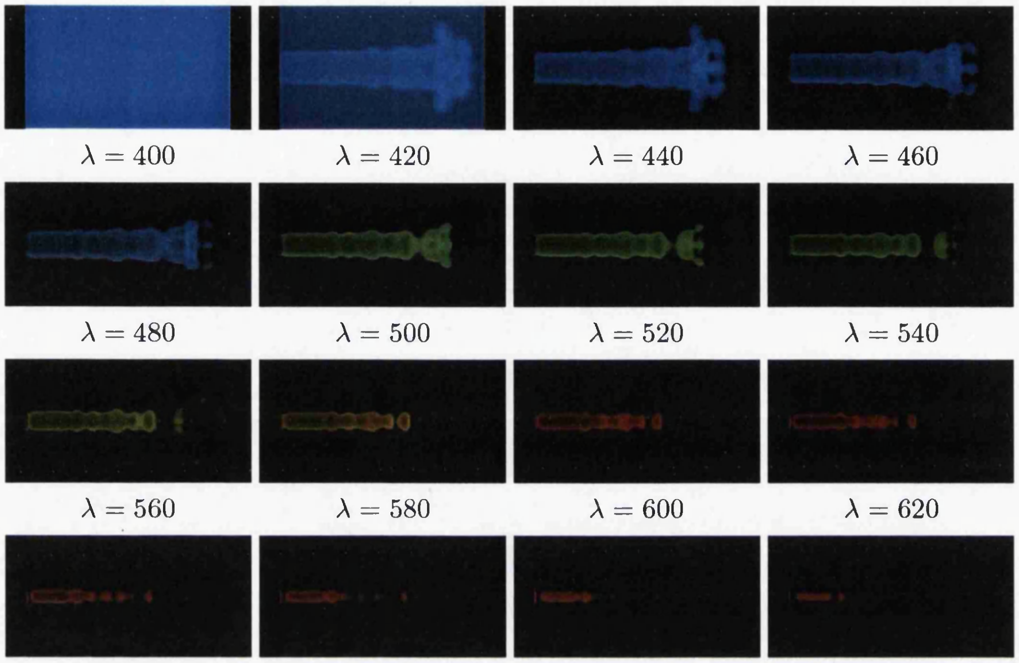
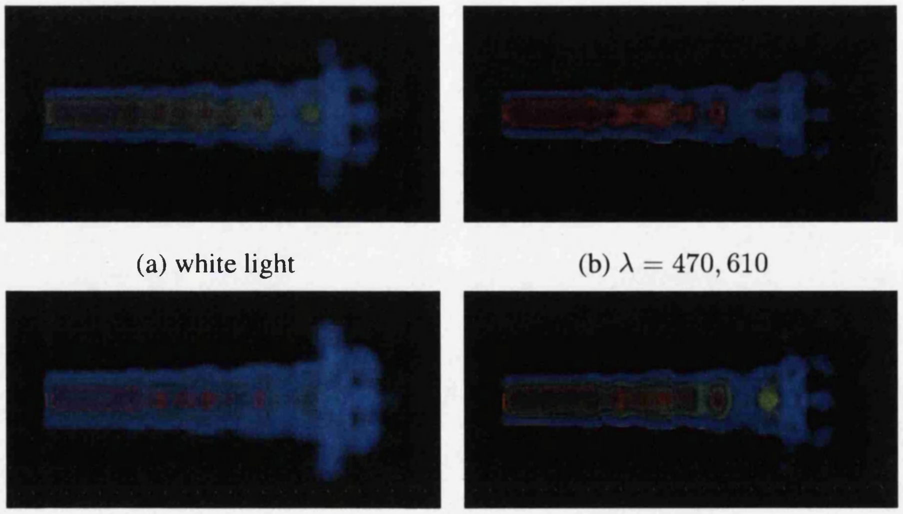


Figure 4.13: Post-illumination with single-peak light source.



(a) white light (b) $\lambda = 470, 610$

(c) $\lambda = 430, 510, 630$ (d) $\lambda = 460, 540, 620, 700$

Figure 4.14: Post-illumination with a white light source and multi-peak light sources.

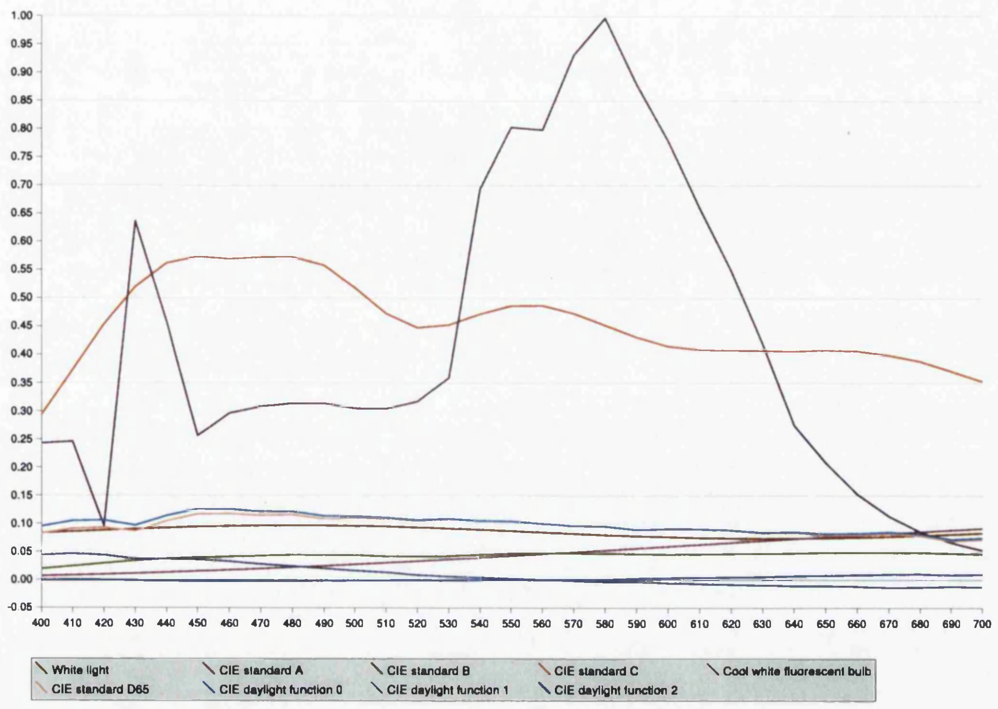


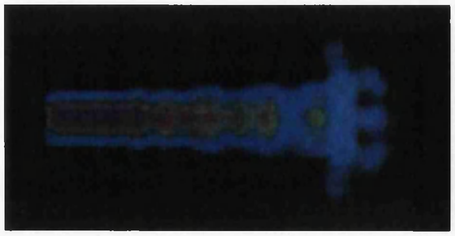
Figure 4.15: Examples of various light sources available for post-illumination.

(d) can depict iso-surfaces defined at, and close to, a few different peak wavelengths.

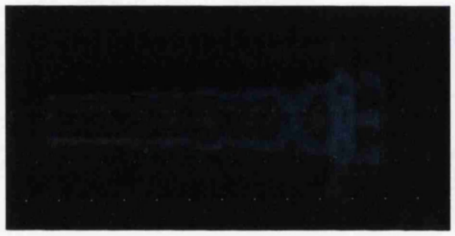
Figures 4.16(a)-(i) shows the results of post-illumination on a volume dataset using various light sources illustrated in Figure 4.15. It is observed that light sources of a near *flat spectra* [Gla95] (*i.e.,* each values of the wavelength is of equal intensity) and of reasonable high intensity such as white light source, CIE standard D65 and CIE daylight function 0 provide the optimal results for viewing the global impression of the dataset, as shown in Figures 4.16(a), (f) and (g). Light sources whose spectra consists of multiple peaks such as CIE standard A, B, C and cool white fluorescent are able to highlight specified features of the dataset corresponding to the peak wavelength, as shown in Figures 4.16(b)-(e), whereas light sources which are of near or below zero such as CIE daylight function 1 and 2, are found to produce a poor display of the dataset as shown in Figures 4.16(h) and (i).

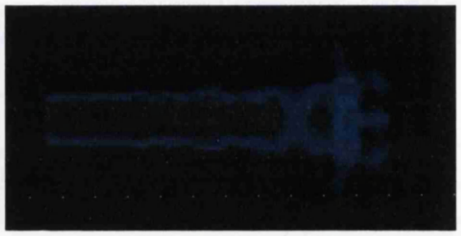
In Figures 4.17 and 4.18, post-illumination is performed at a selected peak wavelength, $\lambda = 450$, to illustrate the setting for the user-definable constants specifying the highest and lowest values of absorption K and scattering S. Figure 4.17 demonstrates the effect of varying the lowest values of absorption, K_{min}, and scattering, S_{min}, with fixed values of 0.5 and 10.0 for the highest absorption, K_{max}, and scattering, S_{max}, respectively. It is found that the results produced are unaffected as the lowest absorption, K_{min}, and scattering, S_{min} are increased from 0.0 to 0.01.

In demonstrating the highest values of absorption K and scattering S, varying values are used for the highest absorption, K_{max}, and scattering, S_{max}, and a fixed value of 0.0001

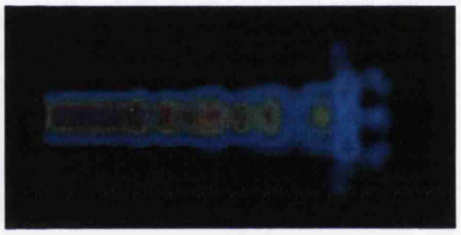

(a) White light

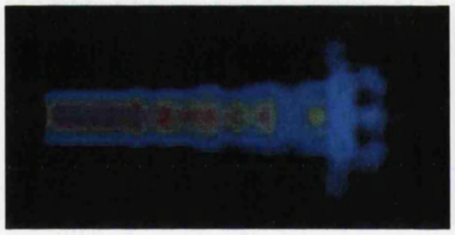
(b) CIE standard A


(c) CIE standard B


(d) CIE standard C

(e) Cool white fluorescent bulb


(f) CIE standard D65


(g) CIE daylight function 0

(h) CIE daylight function 1

(i) CIE daylight function 2

Figure 4.16: Post-illumination with multiple light sources.

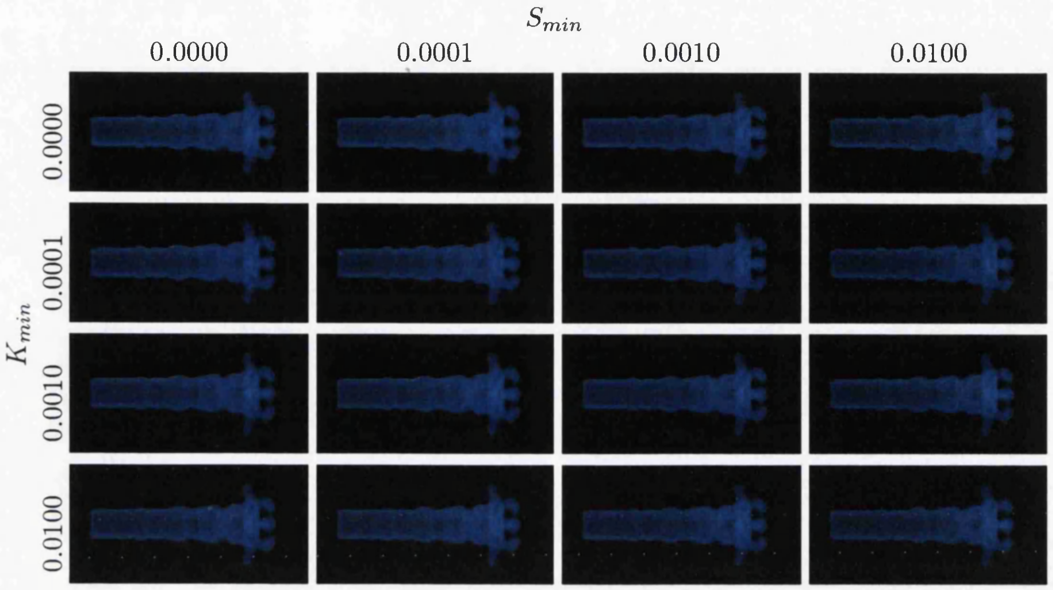


Figure 4.17: Post-illumination at $\lambda = 450$, which is computed using single-peak spectral transfer functions for both K and S, with $K_{max} = 0.5$, $S_{max} = 10.0$ and varying values for K_{min} and S_{min} where K_{min} increases from 0.00 to 0.01 in a top to bottom order and S_{min} increases from 0.00 to 0.01 in a left to right order.

for the lowest absorption, K_{min}, and scattering, S_{min}, as illustrated in Figure 4.18. As the highest scattering S_{max} increases from 1 to 1000, the object reflects more light and becomes more defined. As the highest absorption K_{max} increases from 10^{-10} to 100, the object absorbs more light, but less light is being transmitted. It is observed that even with a low highest absorption K_{max}, a reasonably high value has to be specified for the highest scattering S_{max} for the object to be properly defined.

In general, the design of spectral transfer functions and the control of visual effects with a colour spectrum is more complicated and less intuitive than the conventional RGBα model. However, this does not necessarily undermine the usability of spectral volume rendering. As the spectral colour space is significantly larger than the RGB space, and the Kubelka-Munk model based on absorption K and scattering S is fundamentally more comprehensive than the RGBα model, we believe that it should be possible to identify a collection of spectra for modelling absorption, scattering and illumination that can be effectively deployed in volume visualisation.

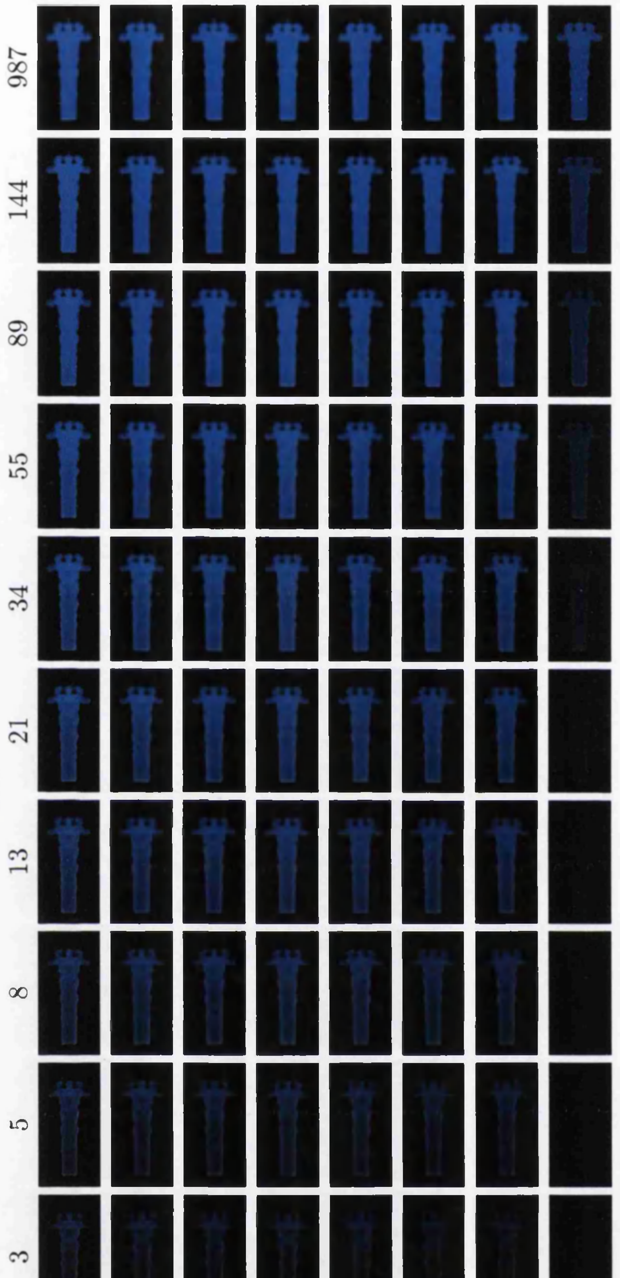


Figure 4.18: Post-illumination at $\lambda = 450$, which is computed using single-peak spectral transfer functions for both K and S, with $K_{min} = 0.0001$, $S_{min} = 0.0001$ and varying values for K_{max} and S_{max} where K_{max} increases from 10^{-10} to 100 in a top to bottom order and S_{max} increases from 1 to 987 in a left to right order.

4.9 iPi – Interactive Post-illumination

Interactive Post-illumination (iPi) is a system designed to demonstrate the technical feasibility of the post-illumination pipeline illustrated in Figure 4.11 in the previous section. The pipeline shows a series of procedures that can implemented in order to generate a system that would enable interactive exploratory visualisation. These procedures can be separated into two engines of post-illumination and rendering. The main motivational aims that lead to the conception of iPi are the desire to provide interactive exploration of volume data sets through the methods of post-illumination and also to present a way of handling spectral data. Figure 4.19 shows an overall view of the user interface that has been developed for the iPi system.

The architecture of iPi consists of two parts, namely a *volume renderer* and a *post-illumination processor*. The volume renderer handles the rendering of the volumetric data sets, which includes calculation of the absorption and scattering properties of the data that can be defined in the spectral transfer functions. The volume renderer additionally generates appropriate outputs for the post-illumination processor in the form of pre-rendered volumetric data sets. The post-illumination processor handles only these pre-rendered data sets, which are then lit (see Equation 4.9) to produce a final image for display using the interrogative light source.

This section begins with a brief overview of the design objectives of iPi. The purpose of the design objectives is to investigate and identify the basic functions that can be achieved by iPi. The interface design of iPi is presented in Section 4.9.2. Later in the section, the rendering attributes of the rendering engine of iPi and its feature of interrogative light sources are discussed. Finally, in Sections 4.9.6 and 4.9.7 the file structures and data managements that are implemented in iPi are presented together with some of the system constraints of iPi.

4.9.1 iPi: Design Objectives

One of the important stages in system development is the investigation and understanding of the requirements and designs of the system. iPi is separated into the two entities of post-illumination and volume rendering. By separating the system into two entities, the main focus of the design objectives can be diverted toward the post-illumination method and the design of the spectral transfer function. The design objectives of iPi are as follows:

- To provide a separation between the rendering entity and the post-illumination entity. By providing this separation, the user is presented with the opportunity to investigate and examine volume data sets without the need of re-computing the rendering integral;
- To present the user with a way of handling spectral data and its visualisation;
- To enable the generation and management of the pre-rendered volumetric data sets for post-illumination through the rendering engine;
- To provide the user with a set of visualisation styles, *i.e.,* setting rendering parameters and refraction indexes; and
- To provide the user with a method for designing a spectral transfer function.

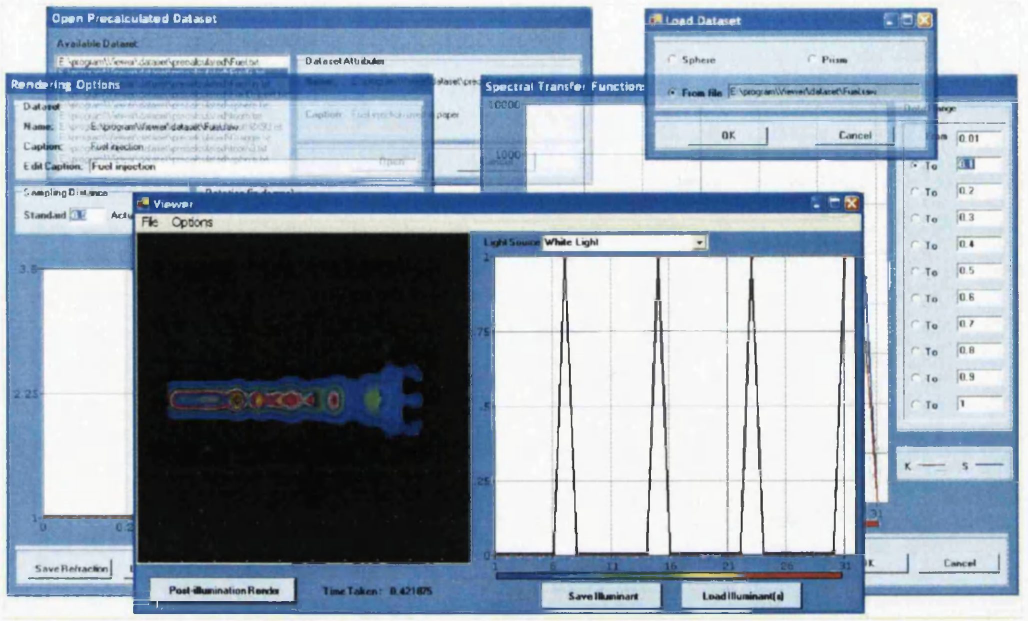


Figure 4.19: The user interface of the iPi system. iPi was initially written in C and tested on a Linux environment. It was later rewritten in Microsoft Visual C# .Net development environment; and this version is shown here.

Figure 4.19 shows the current design of iPi. The current design is the result of several iterations of the designing stage. One of the early designs of iPi presented both rendering and post-illumination options on a single form. The design layout was, however, abandoned as it would not have been able to provide the user with a visual separation of the post-illumination and rendering entities.

4.9.2 iPi: Interface Design

Interface design is one of the criteria that would determine the usability of a system. There are many do's and do not's that have been proposed as guidelines for building a good interface design of a system. Generally, these rules can be catalogued into three design principles and can be defined as follows [Man97]:

- To allow the user to be in charge of the interface;
- To minimise the level of recollection needed by the user; and
- To provide a uniform interface for the user through the system.

Among the other issues that must also be taken consideration during the interface design of a system is the implementation of a design layout that is familiar to the users. By installing familiarity in the system, the users are able to use the system with a minimum amount of training. These design principles and issues are incorporated in the interface design of iPi and can be seen reflected throughout the system.

There are various methods of displaying and accepting input values. One of them would be to provide a visual representation of the input variables as shown in Figures 4.19, 4.20 and 4.21. This type of input entry can be observed in the *Spectral Transfer Functions*, *Rendering Options* and *Interrogative Light Source* functions. The input variables for these functions are spectra, which are being stored as n-arrays of values and are displayed as graphs on the forms. By using a graph to represent the input values, the user is able to have a visual display of the inputted spectra. The user is able to modify the graphs either by clicking and dragging a point or through a form that is assessable by right-clicking a point on the graph. The form would enable the user to specify the position of the graphs with greater precision. These graphs can be saved and loaded for future use.

One of the features of iPi is the ability to handle a collection of pre-rendered volume data sets for post-illumination. As the storage and management of these volume data sets are internally handled by iPi, users would not be encumbered with unnecessary details such as file format and storage. During each of the rendering stages of the volume data sets, the users are able to include brief descriptions of the volume data sets for future references. The descriptions will be displayed alongside the volume data sets during the loading of the pre-calculated volume data sets as references for the users.

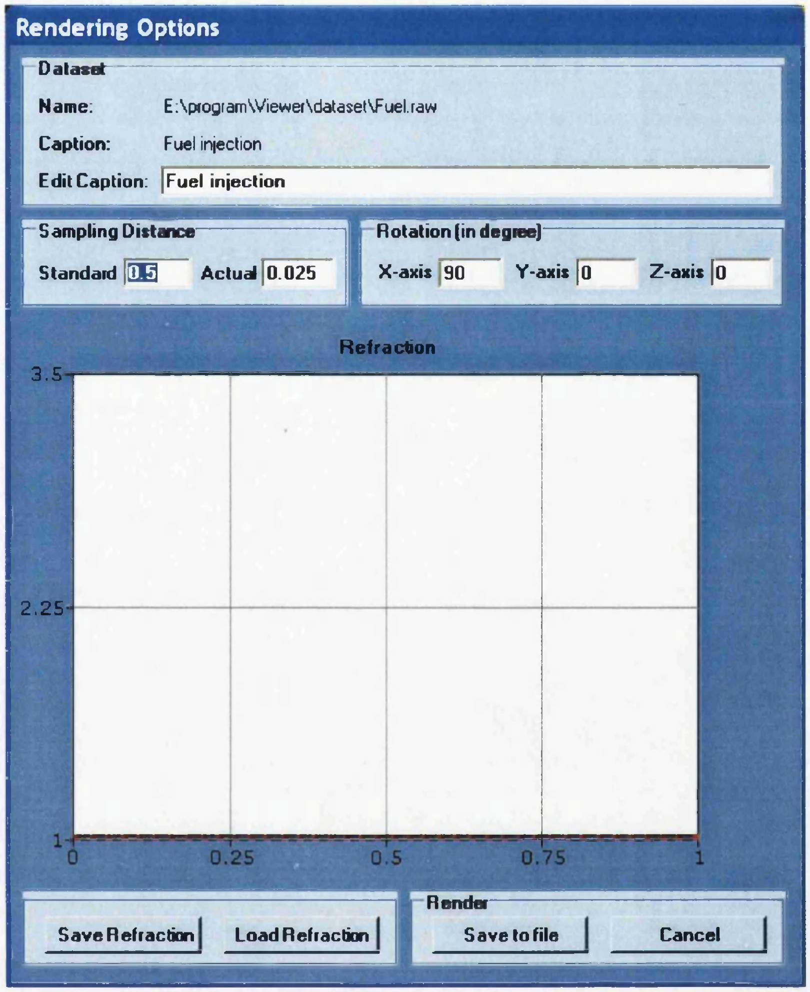
The majority of the parameters in iPi have already been set with pre-defined common values for the convenience of the users. By providing these values, the users are able to immediately use the system.

4.9.3 iPi: Rendering Attributes

As previously mentioned, the processes in iPi have been separated into two engines of post-illumination and rendering. In this section, the rendering engine interface of iPi is presented and examined. Figure 4.20 shows the rendering engine interface of iPi. The interface consists of common rendering attributes that can be found in a typical volume renderer. As iPi has been developed using Microsoft Visual C# .Net, the user is able to access and control the rendering attributes through the graphical user interface of the rendering engine. The rendering attributes that can found in iPi are as follows:

- *Sampling distances*, which is used to specify the standard and actual sampling distance;
- *Rotation*, whose purpose is to define the rotation of the volume data sets (in degrees) for the x-, y- and z-axes; and
- *Refraction*, which is used for specifying the refraction indices for the volume data set. The refraction indexes have been defined to be in the range of 1.00 to 3.5 and are specified as a transfer function. These values are chosen to represent the lowest optical density (*i.e.*, vacuum) and the highest optical density (*i.e.*, gallium). These refraction indexes can be saved and loaded for future use and references.

Due to the nature of its volume rendering integral, iPi does not support any lighting or shading models.


Rendering Options

Dataset

Name: E:\program\Viewer\dataset\Fuel.raw

Caption: Fuel injection

Edit Caption: Fuel injection

Sampling Distance

Standard 0.5 Actual 0.025

Rotation (in degree)

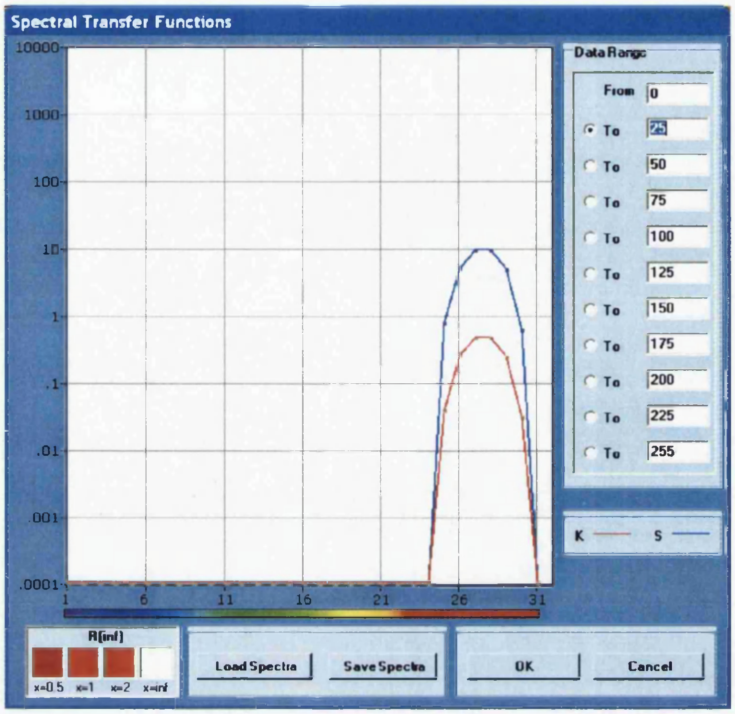
X-axis 90 Y-axis 0 Z-axis 0

Figure 4.20: iPi – rendering engine.

Currently, there are two types of data that are supported by iPi, either volume data sets or objects that are specified procedurally or mathematically, such as a sphere or a prism. The raw volume data sets that can be loaded by iPi are either 8 bit or 16 bit.

4.9.4 iPi: Spectral Transfer Function

The spectral transfer function implemented in iPi is used to represent the absorption and scattering properties of the volume data sets, as demonstrated in Figure 4.21. The absorption and scattering properties are spectra that have been sampled from $400nm$ to $700nm$ at $10nm$ intervals. The spectra are represented as logarithmic graphs in iPi with a maximum and minimum range of 10000 and 0.0001, respectively. As briefly mentioned, the user is able to modify these graphs and save them for future use and references. One of the desired properties of the spectra is that they must follow the physical laws of nature, *e.g.,* the spectra must contain positive values. This property has been incorporated in the construction phase



Data Range
From 0
[✓ To] 25
To 50
To 75
To 100
To 125
To 150
To 175
To 200
To 225
To 255

Figure 4.21: iPi – spectral transfer function.

of the spectra.

4.9.5 iPi: Interrogative Light Source

The post-illumination engine comprises of the interrogative light source as a means for exploring volume data sets. The interrogative light source is set to be between the visible range of $400nm$ to $700nm$, with a $10nm$ interval. The values of the interrogative light source are specified to be between the range of 0 and 1 to ensure that they follow the physical laws of nature. The interrogative light source can be easily modified through its clicking and dragging feature. Similar to the rest of the graphical representation found in iPi, the users are able to save and load the interrogative light for future usage.

There are nine standard light sources that can be chosen for lighting the final image for display. The list of the are displayed in a drop-down menu in iPi and are defined as follows:

- White light;
- CIE standard A;
- CIE standard B;
- CIE standard C;
- Cool white fluorescent bulb;
- CIE standard D65;
- CIE daylight function 0;
- CIE daylight function 1; and
- CIE daylight function 2.

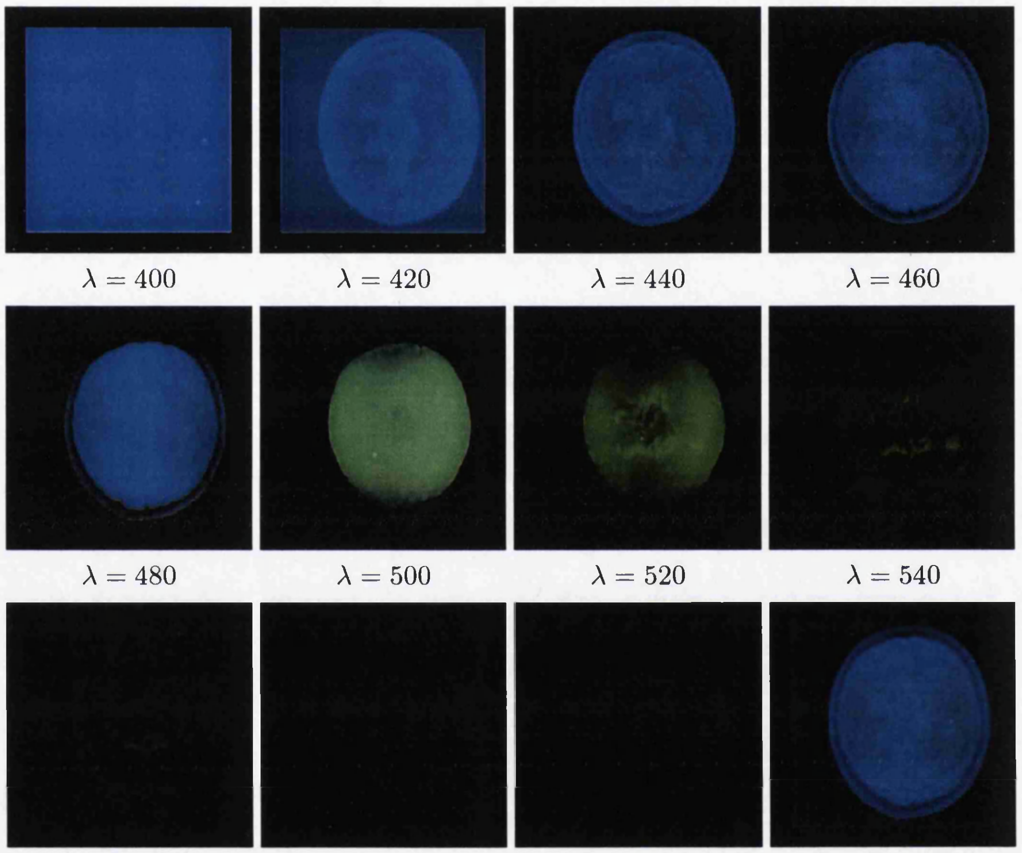


Figure 4.22: Post-illumination of orange data set with single-peak light source and a white light source.

Figure 4.22 shows the resulting images of the orange data set using the interrogative light source.

4.9.6 iPi: File Structure and Data Management

Volume Attribute	**Description**	**Data Type**
Volume Details		
filename	Unique volume identifier	string
caption	Volume dataset summary	string
field	sphere, prism or dataset	integer
xn	x dimension of the dataset	real
yn	y dimension of the dataset	real
zn	z dimension of the dataset	real
bn	Determine whether it is a 8-bit or 16-bit	real
setWidth	Image width	integer
setHeight	Image height	integer
Rendering Attributes		
sd_std	Standard sampling distance	real
sd_act	Actual sampling distance	real
x_axis	x-axis rotation	real
y_axis	y-axis rotation	real
z_axis	z-axis rotation	real
rfr	Tuples (value, data) - refraction	(int, real)
ksSPD	Tuples (value, data) - K, S	(int, real)
xfactor	Normalised x	real
yfactor	Normalised y	real

Table 4.4: Attributes associated with the pre-computed volume data sets.

Attribute	**Description**	**Data Type**
Data range	The data range of the spectral data	real
kSPD	Tuples of (value, data) - K	(int, real)
sSPD	Tuples of (value, data) - S	(int, real)
reflectance	Tuples of (value, data) - r, g, b	(int, real)

Table 4.5: Attributes associated with the spectral transfer function.

File structure and data management are required to ensure the smooth running and the integrity of the system. Table 4.4 describes the attributes that are used in the pre-computation of the volume data sets. During the rendering stage of the volume data sets, the information regarding these data sets will be stored in a file called `pre-info.txt`. This file will be accessed each time the users wish to load a pre-calculated volume data set. Each of the images that are generated during the post-illumination stage can be saved in ppm format.

Table 4.6 presents the attributes that are required in the loading of the pre-calculated volume data set. Whereas Tables 4.5 and 4.7 describe some of the parameters that are used in the spectral transfer function and interrogative light source of iPi.

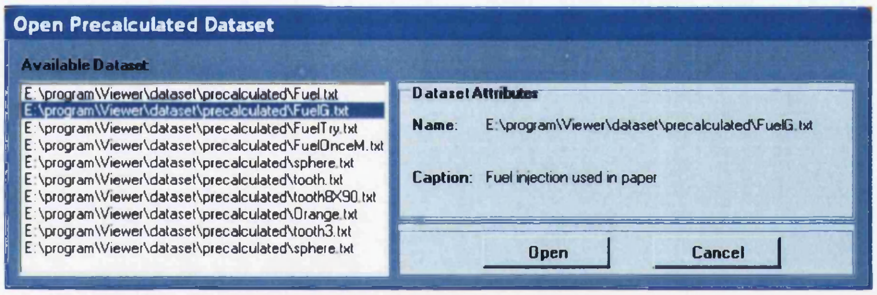
Attribute	Description	Data Type
loadedVolume	The pre-calculated volume data set	(int, real)
width	The width of the data set	int
height	The height of the data set	int

Table 4.6: System environment settings.

Attribute	Description	Data Type
typeLightSource	Type of light source	integer
lset	Tuples of (value, data) - interrogative light	(int, real)
lSPD	Tuples of (value, data) - light source	(int, real)

Table 4.7: Attributes associated with the interrogative light source.

4.9.7 iPi: System Constraints



```
E:\program\Viewer\dataset\precalculated\Fuel.txt
E:\program\Viewer\dataset\precalculated\FuelG.txt
E:\program\Viewer\dataset\precalculated\FuelTry.txt
E:\program\Viewer\dataset\precalculated\FuelOnceM.txt
E:\program\Viewer\dataset\precalculated\sphere.txt
E:\program\Viewer\dataset\precalculated\tooth.txt
E:\program\Viewer\dataset\precalculated\tooth8X90.txt
E:\program\Viewer\dataset\precalculated\Orange.txt
E:\program\Viewer\dataset\precalculated\tooth3.txt
E:\program\Viewer\dataset\precalculated\sphere.txt
```

Name: E:\program\Viewer\dataset\precalculated\FuelG.txt

Caption: Fuel injection used in paper

Figure 4.23: The user interface of the pre-calculated volume data set.

The main objective of this section is to discuss the constraints of iPi in an attempt to provide a greater comprehension of the system. The focus of the discussion is divided into two different areas; post-illumination and rendering. In the rendering engine, only one volume data set can be rendered at a time. The user would specify the required volume data set and the required rendering parameters. The user is also given the opportunity to include a brief description of the volume data set for future reference.

In the post-illumination engine, only one pre-calculated volume data set can be loaded at a time as shown in Figure 4.23. Once the process of loading the pre-calculated volume data set has been completed, the user is able to interrogate the volume data set with the different light sources. The list of the different light sources are those commonly found in colour science and are displayed in the drop-down menu. Currently, only one post-illumination can be viewed at a time. However, this is not seen as a disadvantage in using iPi as the user is able to save each of the resulting post-illuminations.

The interrogative light sources have been defined from $400nm$ to $700nm$, as the human eye is most sensitive within this range and the truncation of the spectra within this range result in minimal errors [SSS92b, SSS92a].

4.10 Summary

In this chapter, a new approach to direct volume rendering has been presented that uses a spectral volume rendering integral based on the Kubelka-Munk theory of diffuse reflectance. It has been shown that the Kubelka-Munk theory not only facilitates a correct spectral volume rendering integral suitable for both solid objects and amorphous matter in volume datasets, but that it also provides volume visualisation with more accurate optical effects than the traditional volume rendering integral based on the RGBα compositing model. We have discussed the design of spectral transfer functions for specifying absorption and scattering coefficients, and the use of post-illumination for integrating pre-processed reflectance images in real-time, which has been demonstrated in the implementation of the iPi system. We have also demonstrated the optical realism in volume visualisation achieved by this approach with a combination of several natural and artificial colour data sets.

While we are enthusiastic about the potential of spectral volume rendering, we have also observed the difficulties in designing spectral transfer functions. This is largely due to the fact that the spectral colour space is much larger than the RGB colour space, and the spectral compositing of reflectance and transmittance based on absorption and scattering can lead to interesting visual effects, which are optically correct but may only be comprehensible to users with adequate knowledge of colour science and colour-vision models. Hence, one of the main scientific challenges in this area is to provide the user with an intuitive means to design spectral transfer functions that can provide a higher degree of freedom in interactive data exploration.

Chapter 5

Algebraic Properties of Volume-based Modelling and Colour Mixing

Contents

5.1 Introduction 87
5.2 Background on Constructive Volume Geometry 88
5.3 Opacity Channel Model 89
5.4 4-Colour Channel Model 90
5.5 The CVG Framework for the Kubelka-Munk Model 98
5.6 Colour Mixing Laws 102
5.7 Summary 110

5.1 Introduction

The topic of discussion in this chapter is divided into two parts. The first part describes an algebraic framework, called *Constructive Volume Geometry* (CVG), used for modelling complex spatial objects using combinational operations [CT98, CT00]. One of the inspirations behind the conception of CVG is the application of a high-level mathematically well-founded modelling scheme for the field of volume graphics. This modelling scheme was demonstrated through two example graphics classes and their algebraic properties. In this chapter, the two example graphics classes are further explored with a new set of scalar operations and their algebraic properties. The algebraic framework of CVG is also extended to include a spatial object based on the Kubelka-Munk model of absorption and scattering together with the notion of vector fields. Through the algebraic framework of CVG, these example graphics classes can be exhibited in a precise, structured manner.

The second part of this chapter investigates the different types of colour mixing laws. Colour mixing laws are widely used in the paint and textile industries. A typical application of colour mixing in the painting industry is the in-store colour mixing machine [Dul06], which can be found in many hardware stores. Given a specified colour sample, the machine is able to mix a container of paint to match the specified colour sample immediately. There are various types of colour mixing laws which can be grouped according to their specified applications. In this chapter, these various types of colour mixing laws are analysed and a review of their implementations and corresponding results are presented.

The organisation of this chapter is as follows:

- In Section 5.2, a brief overview of an algebraic framework, called *Constructive Volume Geometry* (CVG), is presented. This section outlines the concepts and definitions that are utilised throughout the chapter.
- In Sections 5.3 and 5.4, CVG is demonstrated through the applications of the opacity channel and 4-colour channel models, respectively.
- In Section 5.5, a representation of the Kubelka-Munk model based on the CVG algebraic framework is then demonstrated.
- Section 5.6 is an overview of the different types of colour mixing available.

Finally in Section 5.7, our observations and concluding remarks are presented.

5.2 Background on Constructive Volume Geometry

The algebraic framework, called *Constructive Volume Geometry* (CVG), for modelling complex spatial objects using combinational operations was proposed by Chen and Tucker in 1998 [CT98, CT00]. The motivation behind the algebraic framework of CVG is to provide the field of volume graphics with a high-level mathematically well-founded modelling scheme. This scheme is built on the algebraic theory of data types [MT92].

In CVG, a volumetric representation of an object, that is, a *spatial object*, is a tuple

$$\mathbf{o} = (A_0, A_1, \ldots, A_n)$$

of attribute fields defined in $\mathbb{E}^3$. The properties of the spatial object, such as opacity, colours and reflection coefficients, are defined by its attribute fields. Typically in CVG, A_0 is a representation of the opacity field which determines the visibility of the object. An attribute field can be defined mathematically as:

- a scalar field: $F : \mathbb{E}^3 \to \mathbb{R}$;
- an k-dimensional vector function: $F^k : \mathbb{E}^3 \to \mathbb{R}^k$; or
- a t-ranked tensor function: $F^{k^t} : \mathbb{E}^3 \to \mathbb{R}^{k^t}$

where $k, t \in \mathbb{N}$. These attribute fields are demonstrated later in the chapter. The value range of the attribute field can be bounded, and an example of such is shown on the scalar field where $m, n \in \mathbb{R}$, $m \leq n$, such that for all $p \in \mathbb{E}^3$, $m \leq F(p) \leq n$.

In the algebraic framework of CVG, a *spatial object signature* Σ can be denoted as a collection of names for space, attributes and scalar fields, and a *CVG algebra* can be used in classifying a set of operations Φ_i on a spatial object. The set of all spatial objects with signature Σ is known as $\mathbf{O}(\Sigma)$. Later in the chapter, two examples classes that can be found in CVG are presented together with their algebraic properties. A representation of the CVG framework for the Kubelka-Munk model can also be seen later in the chapter.

5.3 Opacity Channel Model

The opacity channel model is one of the most elementary models that can be defined in CVG as it contains only a single attribute of scalar type. The opacity channel model consists of an opacity field, $\mathrm{O} : \mathbb{E}^3 \to [0, 1]$, which defines the visibility of $\mathbb{E}^3$. A spatial object is considered to be *fully opaque* ■, if $\forall p \in \mathbb{E}^3$, $\mathrm{O}(p) = 1$, and *fully transparent* □, if $\forall p \in \mathbb{E}^3$, $\mathrm{O}(p) = 0$ and *translucent* for any values in between [CT00].

In this chapter, three new operations, *blend*, *cap* and *trim*, are introduced in addition to the three original operations. The new operations are constructed through the simplification of their existing operations in the 4-colour channel model. The spatial signature and the CVG algebra of the opacity channel model are as demonstrated in Tables 5.1 and 5.2. The scalar field operations that are defined in Table 5.2 are the point-wise extension [Ric73] of operators on scalars $\in [0, 1]$ and are defined as follows:

$$\mathbf{max}(s_1, s_2) = \begin{cases} s_1 & \text{if } s_1 \geq s_2 \\ s_2 & \text{otherwise} \end{cases}$$

$$\mathbf{min}(s_1, s_2) = \begin{cases} s_1 & \text{if } s_1 \leq s_2 \\ s_2 & \text{otherwise} \end{cases}$$

$$\mathbf{sub}_{01}(s_1, s_2) = \mathbf{max}(0, s_1 - s_2)$$

$$\mathbf{add}_{01}(s_1, s_2) = \mathbf{min}(1, s_1 + s_2)$$

$$\mathbf{cap}_{01}(s_1, s_2) = \begin{cases} s_1 & \text{if } s_1 \leq s_2 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{trim}_{01}(s_1, s_2) = \begin{cases} s_1 & \text{if } s_1 \geq s_2 \\ 0 & \text{otherwise} \end{cases}$$

Spatial Signature	Σ_{op} = opacity channel model
Space	Euclid
Attributes	opacity
Fields	Opacity: Euclid $\to$ opacity

Table 5.1: Spatial signature of the opacity model [CT98].

CVG Algebra	opacity channel model
Spatial Objects	$\mathbf{O}(\Sigma_{\text{op}}) = [\mathbb{E}^3 \rightarrow [0,1]]$
Operations	**union** $\boxed{\cup}$: $\mathbf{O}(\Sigma_{\text{op}}) \times \mathbf{O}(\Sigma_{\text{op}}) \rightarrow \mathbf{O}(\Sigma_{\text{op}})$
	intersection $\boxed{\cap}$: $\mathbf{O}(\Sigma_{\text{op}}) \times \mathbf{O}(\Sigma_{\text{op}}) \rightarrow \mathbf{O}(\Sigma_{\text{op}})$
	difference $\boxed{-}$: $\mathbf{O}(\Sigma_{\text{op}}) \times \mathbf{O}(\Sigma_{\text{op}}) \rightarrow \mathbf{O}(\Sigma_{\text{op}})$
	blend $\boxed{\oplus}$: $\mathbf{O}(\Sigma_{\text{op}}) \times \mathbf{O}(\Sigma_{\text{op}}) \rightarrow \mathbf{O}(\Sigma_{\text{op}})$
	cap ▆: $\mathbf{O}(\Sigma_{\text{op}}) \times \mathbf{O}(\Sigma_{\text{op}}) \rightarrow \mathbf{O}(\Sigma_{\text{op}})$
	trim ▇: $\mathbf{O}(\Sigma_{\text{op}}) \times \mathbf{O}(\Sigma_{\text{op}}) \rightarrow \mathbf{O}(\Sigma_{\text{op}})$
Definitions	$\boxed{\cup}(\mathbf{o}_1, \mathbf{o}_2) = \mathrm{MAX}(\mathrm{O}_1, \mathrm{O}_2)$
	$\boxed{\cap}(\mathbf{o}_1, \mathbf{o}_2) = \mathrm{MIN}(\mathrm{O}_1, \mathrm{O}_2)$
	$\boxed{-}(\mathbf{o}_1, \mathbf{o}_2) = \mathrm{SUB}_{01}(\mathrm{O}_1, \mathrm{O}_2)$
	$\boxed{\oplus}(\mathbf{o}_1, \mathbf{o}_2) = \mathrm{ADD}_{01}(\mathrm{O}_1, \mathrm{O}_2)$
	▆$(\mathbf{o}_1, \mathbf{o}_2) = \mathrm{CAP}_{01}(\mathrm{O}_1, \mathrm{O}_2)$
	▇$(\mathbf{o}_1, \mathbf{o}_2) = \mathrm{TRIM}_{01}(\mathrm{O}_1, \mathrm{O}_2)$

Table 5.2: CVG Algebra of the opacity model with new operations [CT98].

From the definition of operations specified in Table 5.2, a set of algebraic laws [1] for the opacity channel model has been derived and shown in Table 5.3.

5.4 4-Colour Channel Model

The 4-colour channel model is based on the RGBα colour model introduced by Porter and Duff in 1984 [PD84], and is perhaps the most commonly used colour model in the field of computer graphics. The 4-colour channel model consists of four fields; three colour fields, red, green and blue, and an opacity field, which determines the visibility of the object [CT98]. The signature of the 4-colour channel model can be defined as $\mathbf{O}(\Sigma_{4\text{cc}}) = (\text{opacity}, \text{red}, \text{green}, \text{blue})$ and further illustration of it can be observed in Table 4.1 (see Section 4.4, Chapter 4).

The notions of *fully white* and *fully black* for a spatial object are introduced for the 4-colour channel model. A spatial object can be defined to be *fully white* □ if $\forall p \in \mathbb{E}^3$, $\mathrm{O}(p) = 1$ and $\mathrm{R}(p) = 1$ and $\mathrm{G}(p) = 1$ and $\mathrm{B}(p) = 1$, which is $(\mathrm{O}, \mathrm{R}, \mathrm{G}, \mathrm{B}) = (1, 1, 1, 1)$. A spatial object is, however, considered to be *fully black* ■ if $\forall p \in \mathbb{E}^3$, $\mathrm{O}(p) = 0$ and $\mathrm{R}(p) = 0$ and $\mathrm{G}(p) = 0$ and $\mathrm{B}(p) = 0$, which can also be expressed as $(\mathrm{O}, \mathrm{R}, \mathrm{G}, \mathrm{B}) = (0, 0, 0, 0)$. Table 5.4 shows the CVG algebra of the 4-colour channel model. The scalar operations that are defined in Table 5.4 are the point-wise extension [Ric73] of operators on scalars $\in [0, 1]$, which have been defined in the previous section. Additional scalar operations that are used

[1] Some of the laws for the operations, union, intersection and difference, have been proven in [CT98] but are also included for the benefit of the reader.

Commutative laws	$\boxed{\cup}(\mathbf{o}_1,\mathbf{o}_2) = \boxed{\cup}(\mathbf{o}_2,\mathbf{o}_1)$	$\boxed{\cap}(\mathbf{o}_1,\mathbf{o}_2) = \boxed{\cap}(\mathbf{o}_2,\mathbf{o}_1)$
	$\boxed{\oplus}(\mathbf{o}_1,\mathbf{o}_2) = \boxed{\oplus}(\mathbf{o}_2,\mathbf{o}_1)$	
Associative laws	$\boxed{\cup}(\mathbf{o}_1,\boxed{\cup}(\mathbf{o}_2,\mathbf{o}_3)) = \boxed{\cup}(\boxed{\cup}(\mathbf{o}_1,\mathbf{o}_2),\mathbf{o}_3)$	
	$\boxed{\cap}(\mathbf{o}_1,\boxed{\cap}(\mathbf{o}_2,\mathbf{o}_3)) = \boxed{\cap}(\boxed{\cap}(\mathbf{o}_1,\mathbf{o}_2),\mathbf{o}_3)$	
	$\boxed{\oplus}(\mathbf{o}_1,\boxed{\oplus}(\mathbf{o}_2,\mathbf{o}_3)) = \boxed{\oplus}(\boxed{\oplus}(\mathbf{o}_1,\mathbf{o}_2),\mathbf{o}_3)$	
Indempotent laws	$\boxed{\cup}(\mathbf{o},\mathbf{o}) = \mathbf{o}$	$\boxed{\cap}(\mathbf{o},\mathbf{o}) = \mathbf{o}$
	▄$(\mathbf{o},\mathbf{o}) = \mathbf{o}$	▀$(\mathbf{o},\mathbf{o}) = \mathbf{o}$
Identity laws		
Right identity	$\boxed{\cup}(\square,\mathbf{o}) = \mathbf{o}$	$\boxed{\cap}(\blacksquare,\mathbf{o}) = \mathbf{o}$
	$\boxed{\oplus}(\square,\mathbf{o}) = \mathbf{o}$	
Left identity	$\boxed{\cup}(\mathbf{o},\square) = \mathbf{o}$	$\boxed{\cap}(\mathbf{o},\blacksquare) = \mathbf{o}$
	$\boxed{-}(\mathbf{o},\square) = \mathbf{o}$	$\boxed{\oplus}(\mathbf{o},\square) = \mathbf{o}$
	▄$(\mathbf{o},\blacksquare) = \mathbf{o}$	▀$(\mathbf{o},\square) = \mathbf{o}$
Dominance laws	$\boxed{\cup}(\mathbf{o},\blacksquare) = \blacksquare$	$\boxed{\cap}(\mathbf{o},\square) = \square$
	$\boxed{-}(\square,\mathbf{o}) = \square$	$\boxed{\oplus}(\mathbf{o},\blacksquare) = \blacksquare$
	▄$(\square,\mathbf{o}) = \square$	▀$(\blacksquare,\mathbf{o}) = \blacksquare$
Absorption laws	$\boxed{\cup}(\mathbf{o}_1,\boxed{\cap}(\mathbf{o}_1,\mathbf{o}_2)) = \mathbf{o}_1$	$\boxed{\cap}(\mathbf{o}_1,\boxed{\cup}(\mathbf{o}_1,\mathbf{o}_2)) = \mathbf{o}_1$
	$\boxed{\cup}(\mathbf{o}_1,\boxed{-}(\mathbf{o}_1,\mathbf{o}_2)) = \mathbf{o}_1$	$\boxed{\cap}(\mathbf{o}_1,\boxed{\oplus}(\mathbf{o}_1,\mathbf{o}_2)) = \mathbf{o}_1$
	$\boxed{\cup}(\mathbf{o}_1,$▄$(\mathbf{o}_1,\mathbf{o}_2)) = \mathbf{o}_1$	$\boxed{\cup}(\mathbf{o}_1,$▀$(\mathbf{o}_1,\mathbf{o}_2)) = \mathbf{o}_1$
Distributive laws	$\boxed{\cup}(\mathbf{o}_1,\boxed{\cap}(\mathbf{o}_2,\mathbf{o}_3)) = \boxed{\cap}(\boxed{\cup}(\mathbf{o}_1,\mathbf{o}_2),\boxed{\cup}(\mathbf{o}_1,\mathbf{o}_3))$	
	$\boxed{\cap}(\mathbf{o}_1,\boxed{\cup}(\mathbf{o}_2,\mathbf{o}_3)) = \boxed{\cup}(\boxed{\cap}(\mathbf{o}_1,\mathbf{o}_2),\boxed{\cap}(\mathbf{o}_1,\mathbf{o}_3))$	

Table 5.3: A set of algebraic laws for the opacity channel model.

in the 4-colour channel model are defined as follows:

$$\mathbf{select}(s_1,t_1,s_2,t_2) = \begin{cases} t_1 & \text{if } s_1 \geq s_2 \\ t_2 & \text{otherwise} \end{cases} \tag{5.1}$$

$$\mathbf{mix}_{01}(s_1,t_1,s_2,t_2) = \begin{cases} \frac{t_1 s_1 + t_2 s_2}{s_1 + s_2} & \text{if } s_1 + s_2 \neq 0 \\ (t_1 + t_2)0.5 & \text{if } s_1 = s_2 = 0 \end{cases} \tag{5.2}$$

From the definitions of operations summarised in Table 5.4, a set of algebraic laws for the 4-colour channel model has been produced and is shown in Table 5.5.

5.4.1 Union and Intersection Operations

The study of the algebraic properties of constructive operations in volume objects allows us to design more consistent and useful operators, and also allows us to discover the laws governing these operators. The laws of the operators are important as they aid in the reasoning about the operators which can lead to the consistency and predictability in the results of performing a series of operations on volume objects. Tables 5.3 and 5.5 show the algebraic laws which hold for the opacity and 4-colour channels models. In comparison with Table 5.3, it is observed that not all of the laws satisfied in the opacity channel model have been satisfied in the 4-colour channel model shown in Table 5.5. Some of the laws that fail in

CVG Algebra	4 colour channel model
Spatial Objects	$\mathbf{O}(\Sigma_{4cc}) = [\mathbb{E}^3 \to [0,1]]^4$
Operations	**union** $\boxed{\cup}$: $\mathbf{O}(\Sigma_{4cc}) \times \mathbf{O}(\Sigma_{4cc}) \to \mathbf{O}(\Sigma_{4cc})$
	intersection $\boxed{\cap}$: $\mathbf{O}(\Sigma_{4cc}) \times \mathbf{O}(\Sigma_{4cc}) \to \mathbf{O}(\Sigma_{4cc})$
	difference $\boxed{-}$: $\mathbf{O}(\Sigma_{4cc}) \times \mathbf{O}(\Sigma_{4cc}) \to \mathbf{O}(\Sigma_{4cc})$
	blend $\boxed{\oplus}$: $\mathbf{O}(\Sigma_{4cc}) \times \mathbf{O}(\Sigma_{4cc}) \to \mathbf{O}(\Sigma_{4cc})$
	cap ⬓: $\mathbf{O}(\Sigma_{4cc}) \times \mathbf{O}(\Sigma_{4cc}) \to \mathbf{O}(\Sigma_{4cc})$
	trim ⬒: $\mathbf{O}(\Sigma_{4cc}) \times \mathbf{O}(\Sigma_{4cc}) \to \mathbf{O}(\Sigma_{4cc})$
Definitions	$\boxed{\cup}(\mathbf{o}_1,\mathbf{o}_2) = (\mathbf{MAX}(O_1,O_2), \mathbf{SELECT}(O_1,R_1,O_2,R_2), \mathbf{SELECT}(O_1,G_1,O_2,G_2), \mathbf{SELECT}(O_1,B_1,O_2,B_2))$
	$\boxed{\cap}(\mathbf{o}_1,\mathbf{o}_2) = (\mathbf{MIN}(O_1,O_2), \mathbf{SELECT}(O_1,R_1,O_2,R_2), \mathbf{SELECT}(O_1,G_1,O_2,G_2), \mathbf{SELECT}(O_1,B_1,O_2,B_2))$
	$\boxed{-}(\mathbf{o}_1,\mathbf{o}_2) = (\mathbf{SUB}_{01}(O_1,O_2), R_1, G_1, B_1)$
	$\boxed{\oplus}(\mathbf{o}_1,\mathbf{o}_2) = (\mathbf{ADD}_{01}(O_1,O_2), \mathbf{MIX}_{01}(O_1,R_1,O_2,R_2), \mathbf{MIX}_{01}(O_1,G_1,O_2,G_2), \mathbf{MIX}_{01}(O_1,B_1,O_2,B_2))$
	⬓$(\mathbf{o}_1,\mathbf{o}_2) = (\mathbf{CAP}_{01}(O_1,O_2), \mathbf{CAP}_{01}(R_1,R_2), \mathbf{CAP}_{01}(G_1,G_2), \mathbf{CAP}_{01}(B_1,B_2))$
	⬒$(\mathbf{o}_1,\mathbf{o}_2) = \mathbf{TRIM}_{01}(O_1,O_2), \mathbf{TRIM}_{01}(R_1,R_2), \mathbf{TRIM}_{01}(G_1,G_2), \mathbf{TRIM}_{01}(B_1,B_2))$

Table 5.4: CVG Algebra of the 4-colour channel model [CT98, CT00].

the 4-colour channel model are the commutativity of the union and intersections operations, that is $\exists \mathbf{o}_1, \mathbf{o}_2$ such that $\boxed{\cup}(\mathbf{o}_1,\mathbf{o}_2) \neq \boxed{\cup}(\mathbf{o}_2,\mathbf{o}_1)$ and $\boxed{\cap}(\mathbf{o}_1,\mathbf{o}_2) \neq \boxed{\cap}(\mathbf{o}_2,\mathbf{o}_1)$. The commutativity of an operator is often desirable as it helps maintain the predictability and consistency in performing a series of operations. Figure 5.1 shows visual representations of the non-commutativity of the union and intersection operations, and these non-commutativity are also shown in the following arguments.

To show that union is not commutative. Let $\forall p$, $\mathbf{o}_1(p) = (1,0,0,0)$ and $\mathbf{o}_2(p) = (1,0,0,0)$. Then it can be shown that

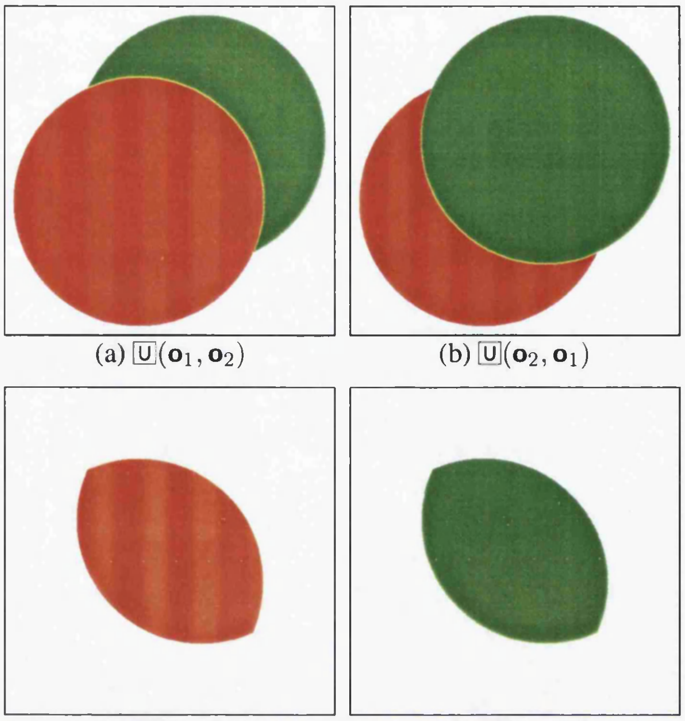
$$\begin{aligned} \boxed{\cup}(\mathbf{o}_1,\mathbf{o}_2)(p) &= (1,1,0,0) \\ &\neq (1,0,0,0) \\ &= \boxed{\cup}(\mathbf{o}_2,\mathbf{o}_1)(p). \end{aligned}$$

Therefore $\boxed{\cup}(\mathbf{o}_1,\mathbf{o}_2) \neq \boxed{\cup}(\mathbf{o}_2,\mathbf{o}_1)$.

To show that intersection is not commutative. Let $\forall p$, $\mathbf{o}_1(p) = (1,0,0,0)$ and $\mathbf{o}_2(p) = (1,0,0,0)$. Then it can be shown that

$$\begin{aligned} \boxed{\cap}(\mathbf{o}_1,\mathbf{o}_2)(p) &= (1,1,0,0) \\ &\neq (1,0,0,0) \\ &= \boxed{\cap}(\mathbf{o}_2,\mathbf{o}_1)(p). \end{aligned}$$

Therefore $\boxed{\cap}(\mathbf{o}_1,\mathbf{o}_2) \neq \boxed{\cap}(\mathbf{o}_2,\mathbf{o}_1)$.



(a) $\boxed{\cup}(\mathbf{o}_1, \mathbf{o}_2)$ (b) $\boxed{\cup}(\mathbf{o}_2, \mathbf{o}_1)$

(c) $\boxed{\cap}(\mathbf{o}_1, \mathbf{o}_2)$ (d) $\boxed{\cap}(\mathbf{o}_2, \mathbf{o}_1)$

Figure 5.1: Results of the union and intersection operations on two spheres. (a) and (b) demonstrate that the union operation of the 4-colour channel model is non-commutative while (c) and (d) demonstrate that the intersection operation of the 4-colour channel model is also non-commutative. The specifications of the objects are as defined in Table 5.4.1.

5.4.2 SELECT Operation

The non-commutativity of the union and intersection operations can be traced back to the **SELECT** operation used in the colour selection of the 4-colour channel model. The definition of the **SELECT** operation on a scalar field is defined in Equation 5.1 and in general its non-commutativity on the scalar field can be shown as follows.

To show that **SELECT** *is not commutative.* Let $\forall p$, $s_1(p) = 1$, $t_1(p) = 1$, $s_2(p) = 1$ and $t_2(p) = 0$.Then it can be shown that

$$\begin{aligned} \mathbf{SELECT}(s_1, t_1, s_2, t_2)(p) &= \mathbf{SELECT}(1, 1, 1, 0) \\ &= 1 \\ &\neq 0 \\ &= \mathbf{SELECT}(1, 0, 1, 1) \\ &= \mathbf{SELECT}(s_2, t_2, s_1, t_1)(p). \end{aligned}$$

Therefore $\mathbf{SELECT}(s_1, t_1, s_2, t_2) \neq \mathbf{SELECT}(s_2, t_2, s_1, t_1)$.

In this section, the existing **SELECT** operation is modified and expanded to include three

Commutative laws	$\boxplus(\mathbf{o}_1,\mathbf{o}_2) = \boxplus(\mathbf{o}_2,\mathbf{o}_1)$	
Associative laws	$\boxed{\cup}(\mathbf{o}_1,\boxed{\cup}(\mathbf{o}_2,\mathbf{o}_3)) = \boxed{\cup}(\boxed{\cup}(\mathbf{o}_1,\mathbf{o}_2),\mathbf{o}_3)$	
Indempotent laws	$\boxed{\cup}(\mathbf{o},\mathbf{o}) = \mathbf{o}$ ▄$(\mathbf{o},\mathbf{o}) = \mathbf{o}$	$\boxed{\cap}(\mathbf{o},\mathbf{o}) = \mathbf{o}$ ▀$(\mathbf{o},\mathbf{o}) = \mathbf{o}$
Identity laws		
Left identity	$\boxed{\cup}(\mathbf{o},$■$) = \mathbf{o}$ ▄$(\mathbf{o},$■$) = \mathbf{o}$ $\boxed{\cap}(\mathbf{o},$□$) = \mathbf{o}$	$\boxminus(\mathbf{o},$□$) = \mathbf{o}$ ▀$(\mathbf{o},$□$) = \mathbf{o}$
Dominance laws	$\boxminus($□$,\mathbf{o}) =$ □ ▀(■$,\mathbf{o}) =$ ■	▄(□$,\mathbf{o}) =$ □
Absorption laws	$\boxed{\cup}(\mathbf{o}_1,\boxed{\cap}(\mathbf{o}_1,\mathbf{o}_2)) = \mathbf{o}_1$ $\boxed{\cup}(\mathbf{o}_1,$▄$(\mathbf{o}_1,\mathbf{o}_2)) = \mathbf{o}_1$	$\boxed{\cup}(\mathbf{o}_1,\boxminus(\mathbf{o}_1,\mathbf{o}_2)) = \mathbf{o}_1$ $\boxed{\cup}(\mathbf{o}_1,$▀$(\mathbf{o}_1,\mathbf{o}_2)) = \mathbf{o}_1$
Distributive laws	$\boxed{\cup}(\mathbf{o}_1,\boxed{\cap}(\mathbf{o}_2,\mathbf{o}_3)) = \boxed{\cap}(\boxed{\cup}(\mathbf{o}_1,\mathbf{o}_2),\boxed{\cup}(\mathbf{o}_1,\mathbf{o}_3))$ $\boxed{\cap}(\mathbf{o}_1,\boxed{\cup}(\mathbf{o}_2,\mathbf{o}_3)) = \boxed{\cup}(\boxed{\cap}(\mathbf{o}_1,\mathbf{o}_2),\boxed{\cap}(\mathbf{o}_1,\mathbf{o}_3))$	

Table 5.5: A set of laws for the 4-colour channel model.

Field	**Sphere 1**	**Sphere 2**
R	1	0
G	0	1
B	0	0
O	1	1

Table 5.6: The attribute specifications of the spatial objects shown in Figures 5.1, 5.2 and 5.3.

new operations and they are classified as follows:

- **SELECT** operation based on luminance;
- **SELECT** operation based on the **MAX** operation; and
- **SELECT** operation based on the **MIN** operation.

A detailed discussion of these operations is given as follows.

5.4.3 SELECT Operation Based on Luminance

The **SELECT** operation based on luminance is calculated using the Y component in the YIQ colour model. The luminance is calculated by substituting the constant colour fields in the objects with field functions. The modified version of the **SELECT** operation on scalars

$\in [0,1]$, where $p \in \mathbb{E}^3$ is defined such that:

$$\mathbf{SELECT}_{lum}(\mathrm{O}_1,\mathrm{R}_1,\mathrm{G}_1,\mathrm{B}_1,\mathrm{O}_2,\mathrm{R}_2,\mathrm{G}_2,\mathrm{B}_2)$$
$$= \begin{cases} (\mathrm{R}_1,\mathrm{G}_1,\mathrm{B}_1) & \text{if } \mathrm{O}_1 > \mathrm{O}_2 \\ (\mathrm{R}_2,\mathrm{G}_2,\mathrm{B}_2) & \text{if } \mathrm{O}_1 < \mathrm{O}_2 \\ (\mathrm{R}_1,\mathrm{G}_1,\mathrm{B}_1) & \text{if } (\mathrm{O}_1 = \mathrm{O}_2) \text{ and} \\ & \quad (\phi((\mathrm{R}_1,\mathrm{G}_1,\mathrm{B}_1),(\mathrm{R}_2,\mathrm{G}_2,\mathrm{B}_2)) = 1) \\ (\mathrm{R}_2,\mathrm{G}_2,\mathrm{B}_2) & \text{if } (\mathrm{O}_1 = \mathrm{O}_2) \text{ and} \\ & \quad (\phi((\mathrm{R}_1,\mathrm{G}_1,\mathrm{B}_1),(\mathrm{R}_2,\mathrm{G}_2,\mathrm{B}_2)) = 2) \end{cases}$$

where

$$\phi((\mathrm{R}_1,\mathrm{G}_1,\mathrm{B}_1),(\mathrm{R}_2,\mathrm{G}_2,\mathrm{B}_2)) = \begin{cases} 1 & \text{if } l_1 > l_2 \\ 2 & \text{if } l_1 < l_2 \\ 1 & \text{if } |l_1 - l_2| < \epsilon \text{ and } (\mathrm{G}_1 > \mathrm{G}_2) \\ 2 & \text{if } |l_1 - l_2| < \epsilon \text{ and } (\mathrm{G}_1 < \mathrm{G}_2) \\ 1 & \text{if } |l_1 - l_2| < \epsilon \text{ and } |\mathrm{G}_1 - \mathrm{G}_2| < \epsilon \\ & \quad \text{and } (\mathrm{R}_1 > \mathrm{R}_2) \\ 2 & \text{if } |l_1 - l_2| < \epsilon \text{ and } |\mathrm{G}_1 - \mathrm{G}_2| < \epsilon \\ & \quad \text{and } (\mathrm{R}_1 < \mathrm{R}_2) \\ 1 & \text{if } |l_1 - l_2| < \epsilon \text{ and } |\mathrm{G}_1 - \mathrm{G}_2| < \epsilon \\ & \quad \text{and } |\mathrm{R}_1 - \mathrm{R}_2| < \epsilon \text{ and } (\mathrm{B}_1 > \mathrm{B}_2) \\ 2 & \text{if } |l_1 - l_2| < \epsilon \text{ and } |\mathrm{G}_1 - \mathrm{G}_2| < \epsilon \\ & \quad \text{and } |\mathrm{R}_1 - \mathrm{R}_2| < \epsilon \text{ and } (\mathrm{B}_1 < \mathrm{B}_2) \\ 1 & \text{otherwise.} \end{cases}$$

where $l_1 = lum(\mathrm{R}_1,\mathrm{G}_1,\mathrm{B}_1)$ and $l_2 = lum(\mathrm{R}_2,\mathrm{G}_2,\mathrm{B}_2)$. lum is specified to calculate the luminance through the Y component of the YIQ colour model and is defined as follows:

$$lum(\mathrm{R}_i,\mathrm{G}_i,\mathrm{B}_i) = 0.299\mathrm{R}_i + 0.587\mathrm{G}_i + 0.114\mathrm{B}_i. \tag{5.3}$$

Figures 5.2(a), (b) and 5.3 (a), (b) show the implementations of the $\mathbf{SELECT}_{lum}$ operation based on luminance for the union and intersection operations. It is observed that union and intersection operations based on luminance are commutative and this is shown by proving the commutativity of the newly defined $\mathbf{SELECT}_{lum}$ operation on scalars:

Proof. Show that $\forall p \in \mathbb{E}^3$, $(\mathbf{SELECT}_{lum}(O_1, R_1, G_1, B_1, O_2, R_2, G_2, B_2) = \mathbf{SELECT}_{lum}(O_2, R_2, G_2, B_2, O_1, R_1, G_1, B_1))$. There are three cases that are needed to be considered and they can be classified into:

- $\mathrm{O}_1 > \mathrm{O}_2$. When $\mathrm{O}_1 > \mathrm{O}_2$, it is shown that

$$\begin{aligned} &\mathbf{SELECT}_{lum}(\mathrm{O}_1,\mathrm{R}_1,\mathrm{G}_1,\mathrm{B}_1,\mathrm{O}_2,\mathrm{R}_2,\mathrm{G}_2,\mathrm{B}_2) \\ &\quad = (\mathrm{R}_1,\mathrm{G}_1,\mathrm{B}_1) \\ &\quad = \mathbf{SELECT}_{lum}(\mathrm{O}_2,\mathrm{R}_2,\mathrm{G}_2,\mathrm{B}_2,\mathrm{O}_1,\mathrm{R}_1,\mathrm{G}_1,\mathrm{B}_1). \end{aligned}$$

- $O_1 < O_2$.When $O_1 < O_2$, it is also shown that

$$\begin{aligned}&\mathbf{SELECT}_{lum}(O_1,R_1,G_1,B_1,O_2,R_2,G_2,B_2)\\ &\qquad = (R_2,G_2,B_2)\\ &\qquad = \mathbf{SELECT}_{lum}(O_2,R_2,G_2,B_2,O_1,R_1,G_1,B_1).\end{aligned}$$

- $O_1 = O_2$. When $O_1 = O_2$, the order of the objects is determined by calculating the luminance of p using the Y component in the YIQ colour model (see Equation 5.3). In calculating the luminance, there are three further cases that are needed to be considered. They are:
  - $l_1 > l_2$. When $l_1 > l_2$, it is shown that

$$\begin{aligned}&\mathbf{SELECT}_{lum}(O_1,R_1,G_1,B_1,O_2,R_2,G_2,B_2)\\ &\qquad = (R_1,G_1,B_1)\\ &\qquad = \mathbf{SELECT}_{lum}(O_2,R_2,G_2,B_2,O_1,R_1,G_1,B_1).\end{aligned}$$

  - $l_1 < l_2$. When $l_1 < l_2$, it is shown that

$$\begin{aligned}&\mathbf{SELECT}_{lum}(O_1,R_1,G_1,B_1,O_2,R_2,G_2,B_2)\\ &\qquad = (R_2,G_2,B_2)\\ &\qquad = \mathbf{SELECT}_{lum}(O_2,R_2,G_2,B_2,O_1,R_1,G_1,B_1).\end{aligned}$$

  - $|l_1 - l_2| < \epsilon$. When $|l_1 - l_2| < \epsilon$, the order of p is then determined by comparing each individual colour channel and it can be seen easily that $\mathbf{SELECT}_{lum}(O_1, R_1, G_1, B_1, O_2, R_2, G_2, B_2) = \mathbf{SELECT}_{lum}(O_2, R_2, G_2, B_2, O_1, R_1, G_1, B_1)$.

From here, it is shown that $\forall p \in \mathbb{E}^3$, $(\mathbf{SELECT}_{lum}(O_1, R_1, G_1, B_1, O_2, R_2, G_2, B_2) = \mathbf{SELECT}_{lum}(O_2, R_2, G_2, B_2, O_1, R_1, G_1, B_1))$. □

5.4.4 SELECT Operation Based on MAX

An alternative version of the **SELECT** operation is the usage of the **MAX** operation when the opacities of both spatial objects are equal. The **SELECT** operation based on the **MAX** operation on scalars $\in [0, 1]$, where $p \in \mathbb{E}^3$ is defined such that:

$$\mathbf{SELECT}_{max}(s_1,t_1,s_2,t_2) = \begin{cases} t_1 & \text{if } s_1 > s_2 \\ t_2 & \text{if } s_1 < s_2 \\ \mathbf{MAX}(t_1,t_2) & \text{otherwise} \end{cases} \tag{5.4}$$

where s_i and t_i are the names of the scalar fields.

Figures 5.2(c), (d) and 5.3(c) and (d) show the implementation of union and intersection operations using Equation 5.4. It can be seen that visually, both of the union and intersection operations based on **MAX** are commutative. Their commutativity can be demonstrated by proving that newly defined $\mathbf{SELECT}_{max}$ operation on scalars is commutative, which is shown in the following proof:

Proof. Show that $\forall p \in \mathbb{E}^3$, $(\mathbf{SELECT}_{max}\ (s_1, t_1, s_2, t_2) = \mathbf{SELECT}_{max}\ (s_2, t_2, s_1, t_1))$. There are three cases that are needed to be considered and they are:

- $s_1 > s_2$. When $s_1 > s_2$, it is shown that

$$\begin{aligned}\mathbf{SELECT}_{max}(s_1, t_1, s_2, t_2) &= t_1 \\ &= \mathbf{SELECT}_{max}(s_2, t_2, s_1, t_1)\end{aligned}$$

- $s_1 < s_2$.When $s_1 < s_2$, it is shown that

$$\begin{aligned}\mathbf{SELECT}_{max}(s_1, t_1, s_2, t_2) &= t_2 \\ &= \mathbf{SELECT}_{max}(s_2, t_2, s_1, t_1)\end{aligned}$$

- $s_1 = s_2$. When $s_1 = s_2$, the resulting colour of the union objects is determined by choosing the maximum value of each individual colour channel. As the **MAX** operation is commutative [CT98], we can easily show that $(\mathbf{SELECT}_{max}\ (s_1, t_1, s_2, t_2)$ $= \mathbf{SELECT}_{max}\ (s_2, t_2, s_1, t_1))$.

From here, it is shown that $\forall p \in \mathbb{E}^3$, $(\mathbf{SELECT}_{max}\ (s_1, t_1, s_2, t_2) = \mathbf{SELECT}_{max}\ (s_2, t_2, s_1, t_1))$. □

5.4.5 SELECT Operation Based on MIN

An alternative version of the **SELECT** operation is the implementation of the **MIN** operation when the opacities of both spatial objects are equal. The **SELECT** operation based on the **MIN** operation on scalars $\in [0, 1]$, where $p \in \mathbb{E}^3$ is defined such that:

$$\mathbf{SELECT}_{min}(s_1, t_1, s_2, t_2) = \begin{cases} t_1 & \text{if } s_1 > s_2 \\ t_2 & \text{if } s_1 < s_2 \\ \text{MIN}(t_1, t_2) & \text{otherwise} \end{cases} \tag{5.5}$$

where s_i and t_i are the names of the scalar fields.

Figures 5.2(e), (f) and 5.3(e), (f) show the implementation of the union and intersection operations using Equation 5.5. It is shown visually that both of the union and intersection operations are commutative. The commutativity of the union and intersection operations can be proved by showing that the **SELECT** operation based on **MIN** is commutative, which is shown in the following proof:

Proof. Show that $\forall p \in \mathbb{E}^3$, $(\mathbf{SELECT}_{min}\ (s_1, t_1, s_2, t_2) = \mathbf{SELECT}_{min}\ (s_2, t_2, s_1, t_1))$. There are three cases that are needed to be considered and they are:

- $s_1 > s_2$. When $s_1 > s_2$, it is shown that

$$\begin{aligned}\mathbf{SELECT}_{min}(s_1, t_1, s_2, t_2) &= t_1 \\ &= \mathbf{SELECT}_{min}(s_2, t_2, s_1, t_1)\end{aligned}$$

- $s_1 < s_2$.When $s_1 < s_2$, it is also shown that

$$\begin{aligned}\mathbf{SELECT}_{min}(s_1, t_1, s_2, t_2) &= t_2 \\ &= \mathbf{SELECT}_{min}(s_2, t_2, s_1, t_1)\end{aligned}$$

- $s_1 = s_2$. When $s_1 = s_2$, the resulting colour of the intersected objects is determined by choosing the minimum value of each individual colour channel. As the **MAX** operation is commutative [CT98], we can easily show that ($\mathbf{SELECT}_{min}$ (s_1, t_1, s_2, t_2) $=$ $\mathbf{SELECT}_{min}$ (s_2, t_2, s_1, t_1)).

From here, it is shown that $\forall p \in \mathbb{E}^3$, ($\mathbf{SELECT}_{min}$ (s_1, t_1, s_2, t_2) $=$ $\mathbf{SELECT}_{min}$ (s_2, t_2, s_1, t_1)). □

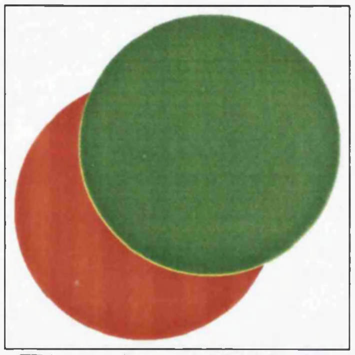
5.4.6 Comparison of Results

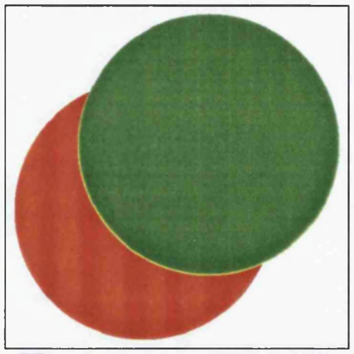
Using the algebraic properties of the objects, we are able to reason and design more consistent union and intersection operators. Figures 5.2 and 5.3 show the results of the newly designed operators, which satisfy the laws of commutativity. Although the results are illustrated in 2D, the operators can easily be implemented and extended to 3D. Even though Figure 5.2 demonstrated that the three modified union operations are all commutative, the question of which of the resulting operations can be perceived to be visually correct arises. In comparisons of all of the operations, it is observed that the **SELECT** operation based on luminance presented a more visually correct result than the **MAX** and **MIN** operations, which show either the colour mixture of the both objects, $\mathbf{O}(\Sigma_{4cc}) = (1, 1, 0, 0)$, or the minimum value of each colour channel, $\mathbf{O}(\Sigma_{4cc}) = (0, 0, 0, 1)$.

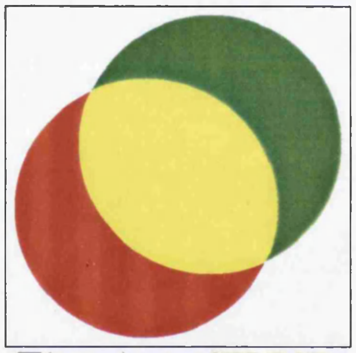
As for the intersection operation, which is shown in Figure 5.3, the **SELECT** operation based on luminance once more presented a more visually correct result in comparison with the other two operations. The **MAX** operation presented a result where the intersected area is the colour mixture of both of the objects that is $\mathbf{O}(\Sigma_{4cc}) = (1, 1, 0, 0)$. Whereas for the **MIN** operation, the intersected area resulted in the colour black as the **MIN** operation takes the minimum value of each individual colour channel, which in this particular situation is $\mathbf{O}(\Sigma_{4cc}) = (0, 0, 0, 1)$.

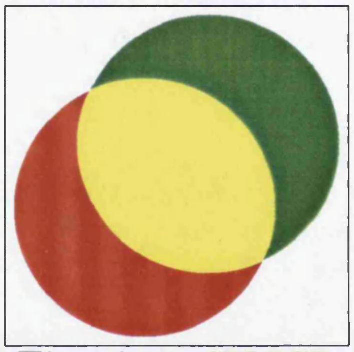
5.5 The CVG Framework for the Kubelka-Munk Model

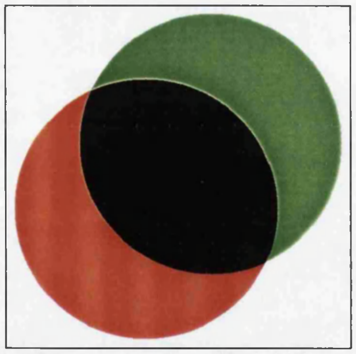
The Kubelka-Munk model [KM31] is concerned with the total diffuse radiation from a material in terms of its *absorption* (K) and *scattering* (S) properties. Fundamentally it is a two-flux model, which considers two radiation fluxes, namely *reflectance* (R) and *transmittance* (T), that pass through a continuous medium in two opposite directions. Figure 5.4


(a) $\boxed{\cup}(\mathbf{o}_1, \mathbf{o}_2)$ using $\mathbf{SELECT}_{lum}$


(b) $\boxed{\cup}(\mathbf{o}_2, \mathbf{o}_1)$ using $\mathbf{SELECT}_{lum}$


(c) $\boxed{\cup}(\mathbf{o}_1, \mathbf{o}_2)$ using $\mathbf{SELECT}_{max}$


(d) $\boxed{\cup}(\mathbf{o}_2, \mathbf{o}_1)$ using $\mathbf{SELECT}_{max}$


(e) $\boxed{\cup}(\mathbf{o}_1, \mathbf{o}_2)$ using $\mathbf{SELECT}_{min}$

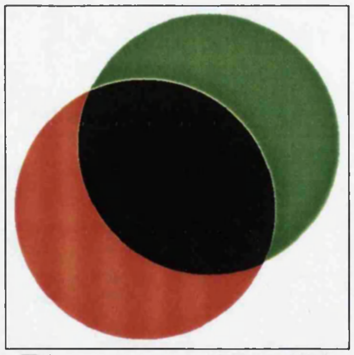
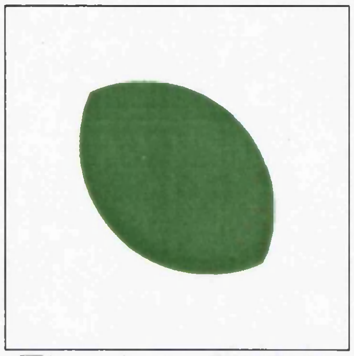
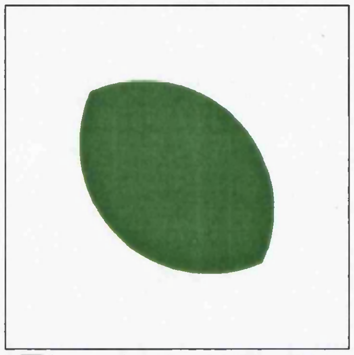
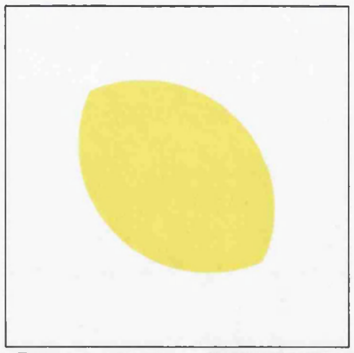
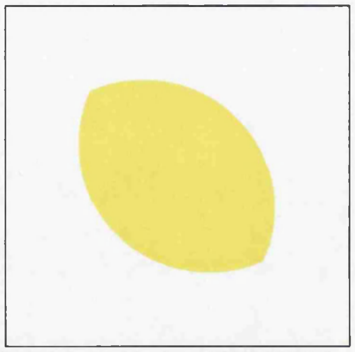

(f) $\boxed{\cup}(\mathbf{o}_2, \mathbf{o}_1)$ using $\mathbf{SELECT}_{min}$

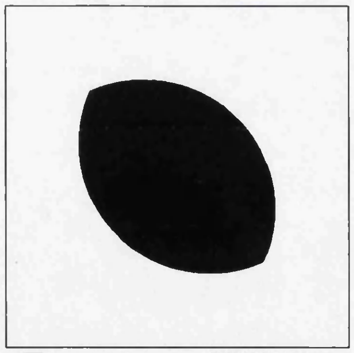
Figure 5.2: Results of the modified union operation on two spheres. The specifications of the objects are as defined in Table 5.4.1.


(a) $\boxed{\cap}(\mathbf{o}_1, \mathbf{o}_2)$ using $\mathbf{SELECT}_{lum}$


(b) $\boxed{\cap}(\mathbf{o}_2, \mathbf{o}_1)$ using $\mathbf{SELECT}_{lum}$


(c) $\boxed{\cap}(\mathbf{o}_1, \mathbf{o}_2)$ using $\mathbf{SELECT}_{max}$


(d) $\boxed{\cap}(\mathbf{o}_2, \mathbf{o}_1)$ using $\mathbf{SELECT}_{max}$


(e) $\boxed{\cap}(\mathbf{o}_1, \mathbf{o}_2)$ using $\mathbf{SELECT}_{min}$

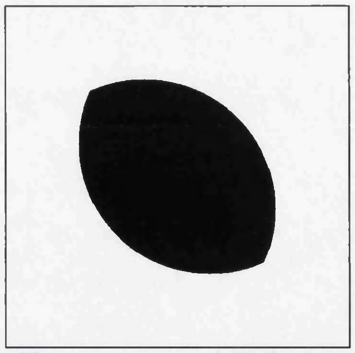

(f) $\boxed{\cap}(\mathbf{o}_2, \mathbf{o}_1)$ using $\mathbf{SELECT}_{min}$

Figure 5.3: Results of the modified intersection operation on two spheres. The specifications of the objects are as defined in Table 5.4.1.

demonstrates the relationships between K, S, R and T in the Kubelka-Munk model. The Kubelka-Munk model is often applied on materials that are translucent, opaque and transparent coloured layers on opaque layer. A detailed discussion on the Kubelka-Munk model can be seen in the previous chapter.

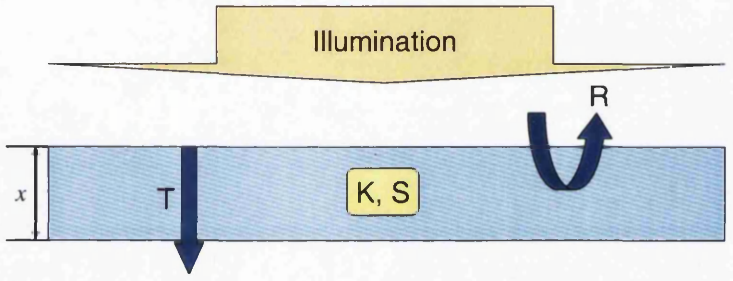


Figure 5.4: K, S, R and T are the four properties that are taken into consideration during the calculation of the Kubelka-Munk model.

In the previous models, each attribute field in the spatial object specification is defined to be a scalar field, $F : \mathbb{E}^3 \to \mathbb{R}$. However, as the Kubelka-Munk model utilises continuous spectra in its implementation, the values of its attributes K, S, R and T are defined through *vector fields*. Hence Kubelka-Munk model based on the CVG framework is a tuple $\mathbf{o} = (\mathrm{K}, \mathrm{S}, \mathrm{R}, \mathrm{T})$ where $\mathrm{K}, \mathrm{S}, \mathrm{R}, \mathrm{T} : \mathbb{E}^3 \to (0, \infty)^k$ [AC05]. In our implementation, we have chosen $k = 31$, allowing each spectrum to be sampled across the visible range of $400nm$ to $700nm$ in $10nm$ intervals. The spatial signature of the Kubelka-Munk model based on the CVG framework is as shown in Table 5.7.

Spatial Signature	Σ_{km} = kubelka-munk model
Space	Euclid
Attributes	reflectance, transmittance, scattering, absorption
Fields	Reflectance: Euclid → reflectance
	Transmittance: Euclid → transmittance
	Scattering: Euclid → scattering
	Absorption: Euclid → absorption

Table 5.7: Spatial signature of the Kubelka-Munk model.

For a single layer [Kub48] of thickness x, R and T relate to K and S as:

$$\mathrm{R}(\lambda) = \frac{\sinh(b(\lambda)\mathrm{S}(\lambda)x)}{a(\lambda)\sinh(b(\lambda)\mathrm{S}(\lambda)x) + b(\lambda)\cosh(b(\lambda)\mathrm{S}(\lambda)x)}$$
$$\mathrm{T}(\lambda) = \frac{b(\lambda)}{a(\lambda)\sinh(b(\lambda)\mathrm{S}(\lambda)x) + b(\lambda)\cosh(b(\lambda)\mathrm{S}(\lambda)x)}$$

where

$$a(\lambda) = \frac{S(\lambda) + K(\lambda)}{S(\lambda)}, \quad b(\lambda) = \sqrt{a(\lambda)^2 - 1}.$$

where λ is the wavelength.

Given the reflectance and transmittance of two layers, R_1, T_1, R_2 and T_2, the composition of the two layers [Kub54] is:

$$\begin{aligned} R(\lambda) &= R_1(\lambda) + \frac{T_1(\lambda)^2 R_2(\lambda)}{1 - R_1(\lambda)R_2(\lambda)} \\ T(\lambda) &= \frac{T_1(\lambda)T_2(\lambda)}{1 - R_1(\lambda)R_2(\lambda)}. \end{aligned}$$

The simplest form of the Kubelka-Munk theory can be observed in the calculation of the reflectance of an opaque medium. Assuming that the opaque medium is an infinitely thick layer [KM31], $R_\infty(\lambda)$ can be defined as:

$$R_\infty(\lambda) = 1 + \left(\frac{K(\lambda)}{S(\lambda)}\right) - \left[\left(\frac{K(\lambda)}{S(\lambda)}\right)^2 + 2\left(\frac{K(\lambda)}{S(\lambda)}\right)\right]^{\frac{1}{2}}.$$

The inverse of the above equation gives:

$$\frac{K(\lambda)}{S(\lambda)} = \frac{(1 - R_\infty(\lambda))^2}{2R_\infty(\lambda)}$$

which shows that the reflectance of an opaque medium, $R_\infty(\lambda)$, relates only to the ratio of $K(\lambda)$ and $S(\lambda)$. The ratio of $K(\lambda)$ and $S(\lambda)$ can be calculated as [All80]:

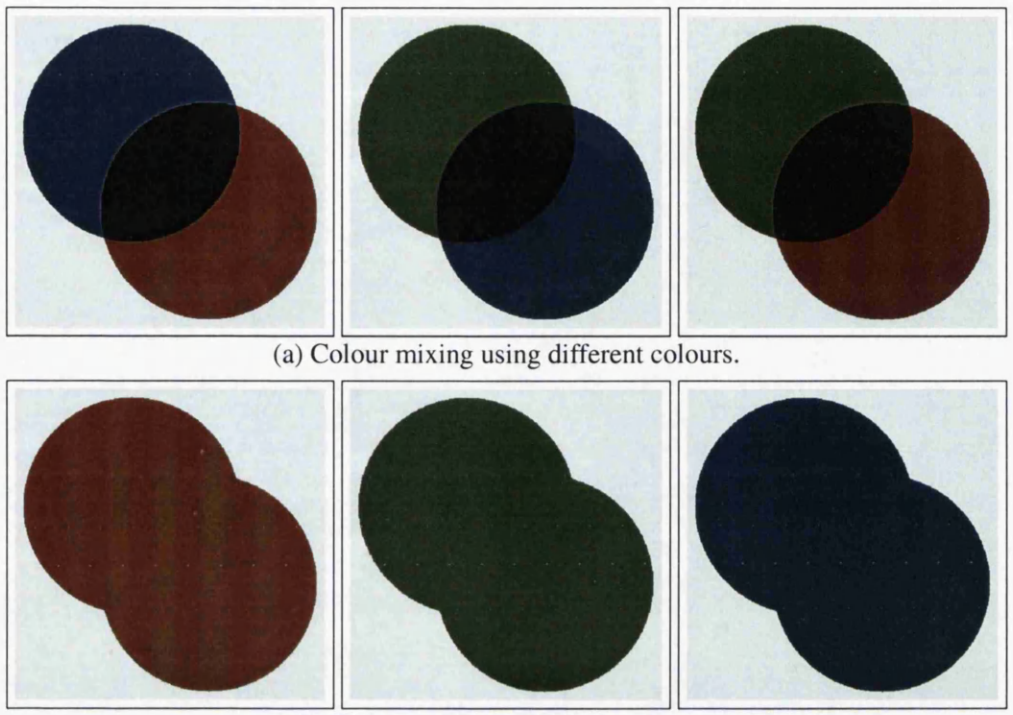
$$\begin{aligned} \frac{K(\lambda)}{S(\lambda)} &= \frac{c_1k_1(\lambda) + c_2k_2(\lambda) + \ldots + c_nk_n(\lambda)}{c_1s_1(\lambda) + c_2s_2(\lambda) + \ldots + c_ns_n(\lambda)} \\ &= \frac{\sum_{i=1}^{n} c_ik_i(\lambda)}{\sum_{i=1}^{n} c_is_i(\lambda)} \end{aligned} \qquad (5.6)$$

where n is the number of pigments and c_i represents the concentration values of pigments. $k_i(\lambda)$ and $s_i(\lambda)$ are used to define the absorption and scattering properties of the pigments. It is worth noting that the total sum of the concentration values is 1 (*i.e.*, $\sum_{i=1}^{n} c_i = 1$) [MB04]. Figure 5.5 shows the implementation of the Kubelka-Munk theory of colour mixing using mixtures of same colour and mixtures of different colour.

Equation 5.6 is generally used as a form of subtractive colour mixing and is known as *complex subtractive mixing* [Ber00]. The Kubelka-Munk theory of colour mixing is used extensively in the industry for simulating the results of mixing paints and inks. The applications of the Kubelka-Munk theory have been successfully implemented in the field of computer graphics for predicting paint mixtures and simulating their appearances [HM92, CAS+97, RMN03, BWL04].

5.6 Colour Mixing Laws

There are extensive surveys into the different types of colour mixing laws. These surveys can be as complex and detailed as the one presented by Duncan [Dun49] or as simple and straightforward as described by Rigg [Rig97] and Nassau [Nas01]. Colour mixing laws can generally be grouped into *additive colour mixing* and *subtractive colour mixing*. Duncan,


(a) Colour mixing using different colours.

(b) Colour mixing with same colours.

Figure 5.5: Kubelka-Munk theory of colour mixing. The colours used in the implementation have been estimated from the MacBeth Color Checker [Pas06].

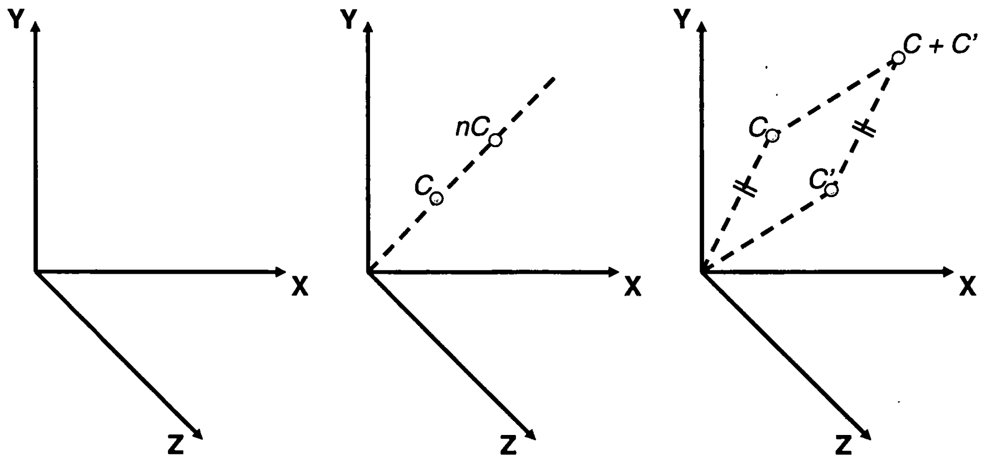
however, argued that the classification of colour mixing laws into two generic types is an oversimplification, and presented eight different categories of colour mixing laws. In this thesis, the colour mixing laws are grouped according to the colour systems in which they operate, that is, *subtractive colour systems* and *additive colour systems*. In additive colour mixing, a colour is generally generated from a mixture of coloured lights. In subtractive colour mixing, however, a colour is produced from the subtraction of colour components from a white light.

The research into the properties of additive colour mixing can be traced back to Grassmann in 1853 [Gra54], and since then, a significant amount of research effort has been made in them [Blo47, Tre53, Tre54]. One of the most common applications of additive colour mixing is within the *Cathode Ray Tube* (CRT) [Poy96, Kel97], which can be found in televisions and monitors. There are three primary colours in additive colour mixing; red, green and blue, and the colour white can be produced when the three primary colours are mixed in the right proportion. One method of demonstrating additive colour mixing would be to shine two or more lights simultaneously onto the same location. As there is no interaction between the lights, the eye would receive the combination of all the lights. Another method of demonstrating additive colour mixing is to rotate a *Maxwell disc*, which is made of segments of different colours. When the rotating disc reaches a certain speed, the segments of colours would be seen to be blended together. One of the most commonly used colour models in computer graphics that uses additive colour mixing is the RGBα model [PD84].

Subtractive colour mixing is, however, different from additive colour mixing as its resulting colour mixture is produced from the selective subtraction of colour components from a white light. The mechanisms of selective subtractions can be observed in nature, in processes such as emission, absorption, refraction, interference and scattering. There are three primary colours in subtractive colour mixing; cyan, magenta and yellow, although black is occasionally used as a way of improving the colour contrast. Opposite to additive colour mixing, the colour black is produced when the three primary colours are mixed in the proper proportions. A simple demonstration of subtractive colour mixing can be performed by shining a white light source through a yellow and a cyan piece of glass filters. After the first glass filter, only 400nm to 450nm of the white light source remains. This is because the light source between 450nm and 700nm would have been absorbed by the yellow glass filter. The resulting colour that could be seen after the second glass filter would be black as the remaining light source of 400nm to 450nm would have been absorbed by the cyan glass filter. The application of subtractive colour mixing can be observed in the fields of photography and printing [Oht71]. CMY colour model is an example of a colour model in computer graphics that uses subtractive colour mixing in its colour representation [FvDFH95].

The majority of existing colour mixing laws have been implemented in the areas of computer graphics. This section begins by examining a form of additive colour mixing laws called *Grassmann's laws*. In Section 5.6.2, a type of subtractive colour mixing laws is investigated. Finally in Section 5.6.3, the comparisons derived on the implemented colour mixing laws are presented.

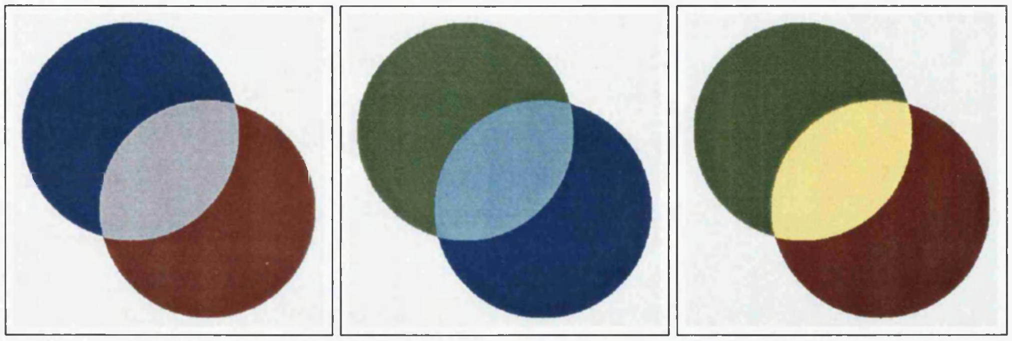
5.6.1 Grassmann's Laws of Colour Mixing



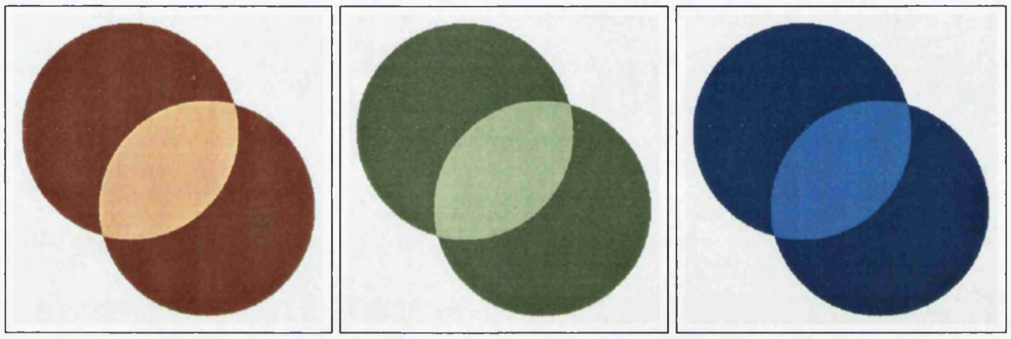
(a) Grassmann's first law (b) Grassmann's second law (c) Grassmann's third law

Figure 5.6: The visual representation of Grassmann's laws of colour mixing [Bou47].

Grassmann's laws of colour mixing is an additive colour mixing law that has existed for more than a century, and can be applied both in the RGB and XYZ colour models. The Grassmann's law of colour mixing is commonly defined by its three laws [Gra54, Bou47] and they can be described as follows.



(a) Grassmann's third law of colour mixing using different colours.



(b) Grassmann's third law of colour mixing with same colours.

Figure 5.7: Grassmann's third laws of colour mixing in the RGB colour space. The input colours have been estimated using the MacBeth Color Checker [Pas06].

Grassmann's First Law.

The arguments regarding the representation of colours in a three-dimensional space is presented in Grassmann's first law. It is argued that by utilising a geometrical representation of colour, we are able to illustrate the properties and concepts of colour with more flexibility and ease in comparison with the colour triangle, *i.e.*, a two-dimensional space. By using a three-dimensional colour space, each colour is specified in its own coordinates without the overlapping with any other colours. A more detailed discussion on these has been discussed by Bouma in 1947 [Bou47].

A three-dimensional colour space can be represented by three axes, where each of the axes represents one of the three primary colours. The primary colours are specified such that each of them cannot be reproduced by the other two primary colours. Grassmann's first law of colour mixing can be formally stated as follows [Bou47]:

Law 1. *A colour, C, can be represented by:*

$$C = x\mathbf{X} + y\mathbf{Y} + z\mathbf{Z} \tag{5.7}$$

where x, y and z represent the units of the primary colours and $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$ are the primary colours of the three-dimensional colour space.

A visual representation of Equation 5.7 can be seen in Figure 5.6(a).

Grassmann's Second Law.

Grassmann's second law of colour mixing examines the coordinates of colours in a three-dimensional colour space. The law states that if a colour mixture matches a specified sample colour, the match will still hold even when the colour mixture and sample colour are both increased by the same amount of brightness. Grassmann's second law can be formally stated as follows [Bou47]:

Law 2. *A colour, C, is a match for a colour mixture, such that,*

$$C = x\mathbf{X} + y\mathbf{Y} + z\mathbf{Z}$$

and this match is still satisfied even though

$$nC = n(x\mathbf{X}) + n(y\mathbf{Y}) + n(z\mathbf{Z})$$

where n represents the amount of brightness.

Figure 5.6(b) shows the visual representation of Grassmann's second law. In Figure 5.6(b), it can be observed that a straight line can be drawn from the *origin* towards the coordinates of nC.

Grassmann's Third Law.

The representation of a colour mixture in a three-dimensional space is discussed in Grassmann's third law of colour mixing. Figure 5.7 shows the results of the implementation of Grassmann's third law of colour mixing in the RGB colour space using mixtures of the same and different colours. The law defines that, given the coordinates of the colours C and C', the coordinates of its mixture $C + C'$ can be determined through coordinate addition. Grassmann's third law can be formally stated as follows [Bou47]:

Law 3. *A colour mixture, $C + C'$, can be defined such that,*

$$C + C' = (x + x')\mathbf{X} + (y + y')\mathbf{Y} + (z + z')\mathbf{Z}$$

where the coordinates of C is

$$C = x\mathbf{X} + y\mathbf{Y} + z\mathbf{Z}$$

and C' is

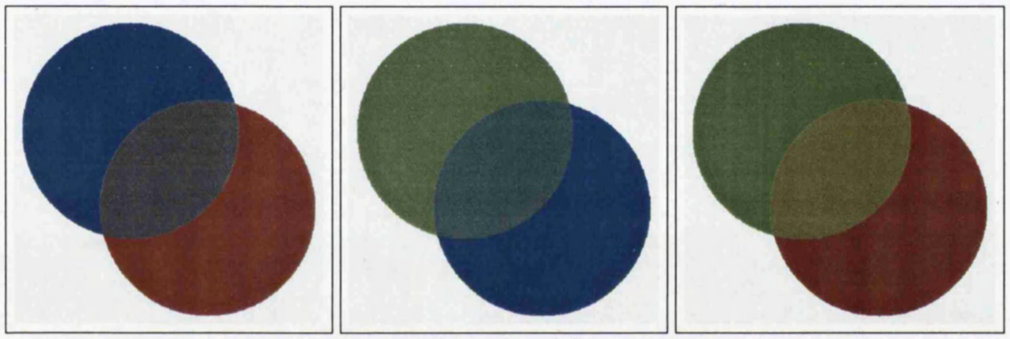
$$C' = x'\mathbf{X} + y'\mathbf{Y} + z'\mathbf{Z}$$

are known. Figure 5.6(c) illustrates the visual representation of Grassmann's third law. In Figure 5.6(c), a parallelogram can be formed when the coordinates of the origin, *C, C' and $C + C'$ are joined together[Bou47].*

Further examination of the Grassmann's laws of colour mixing shows similarity with the blend operation in the 4-colour channel model, which is a form of additive colour mixing. The colour mixing law has been briefly discussed in the 4-colour channel model, which is based on the RGBα colour model [PD84]. The 4-colour channel model comprises of four fields; three colour fields, red, green and blue, and an opacity field, which determines the visibility of the object. The blend operation in the 4-colour channel model is defined such that [CT98, CT00]:

$$\begin{aligned}\boxplus(\mathbf{o}_1,\mathbf{o}_2) = (&\mathbf{ADD}_{01}(\mathrm{O}_1,\mathrm{O}_2),\mathbf{MIX}_{01}(\mathrm{O}_1,\mathrm{R}_1,\mathrm{O}_2,\mathrm{R}_2),\\ &\mathbf{MIX}_{01}(\mathrm{O}_1,\mathrm{G}_1,\mathrm{O}_2,\mathrm{G}_2),\mathbf{MIX}_{01}(\mathrm{O}_1,\mathrm{B}_1,\mathrm{O}_2,\mathrm{B}_2)).\end{aligned}$$

The definitions of $\mathbf{ADD}_{01}$ and $\mathbf{MIX}_{01}$ can be found in Sections 5.3 and 5.4, respectively. Figure 5.8 shows the results of the implementation of this blend operation in the 4-colour channel model.


(a) Colour mixing using different colours.

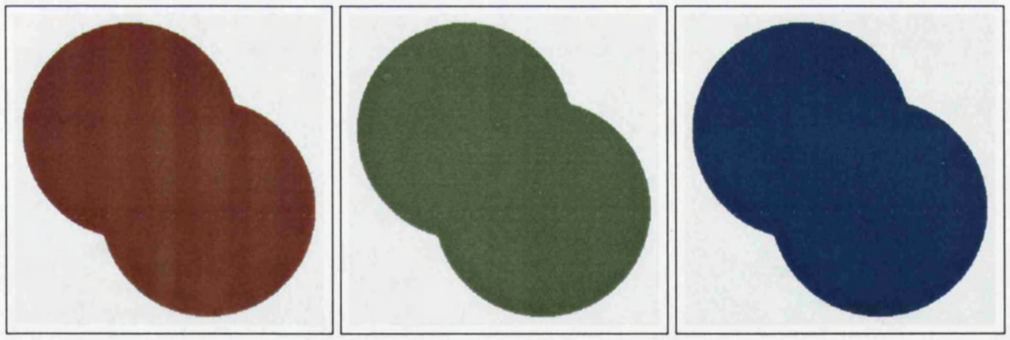
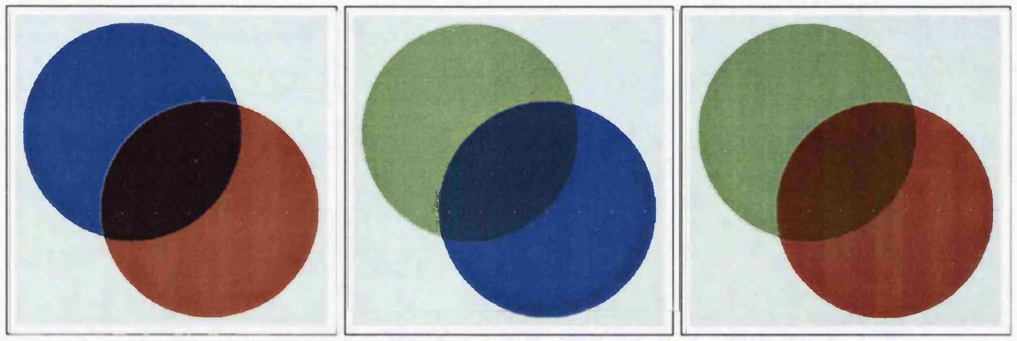

(b) Colour mixing with same colours

Figure 5.8: Colour blending of the 4-colour channel model. The input colours have been estimated using the MacBeth Color Checker [Pas06].

A comparison between Grassmann's third law of colour mixing and colour blending of the 4-colour channel model shows several similarities and differences between the two form of colour mixing. One of the similarities between them is that both of them are based on the additive nature of colour mixing. Despite this similarity, there is one difference that can be observed between the two forms of colour mixing; in the colour blending of the 4-colour channel model, the resulting colour mixture is influenced by the opacity weightings of the objects, whereas in the Grassmann's third law, the resulting colour mixture is not determined by any opacity or density. Another difference that can seen between the two forms of colour mixing is that the colour blending of the 4-colour channel model operates in the bounded region of between 0 and 1, whereas Grassmann's third law operates in an unbounded and infinite range. Operating in a bounded region can be considered as an advantage as the colour mixture can be made to be always in the physically visible range and can be displayed on a viewing display.

5.6.2 Bouguer-Beer's Law of Colour Mixing


(a) Colour mixing using different colours.

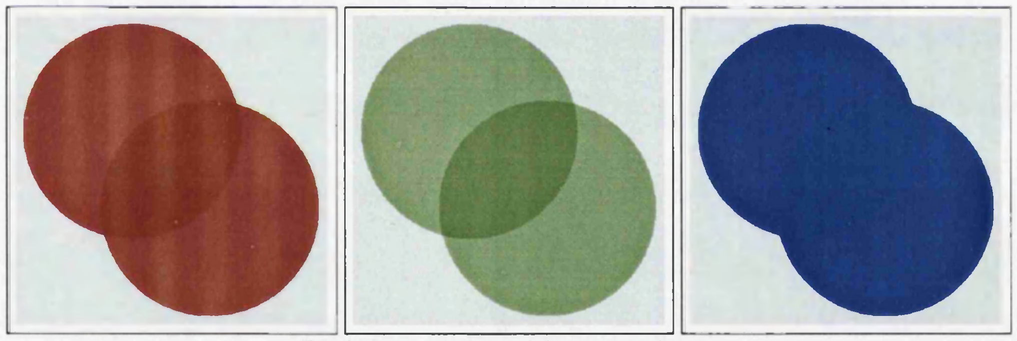

(b) Colour mixing with same colours.

Figure 5.9: Bouguer-Beer's law of colour mixing. The colours used in the implementation have been estimated from the MacBeth Color Checker [Pas06].

There are two different categories of subtractive colour mixing law, and they are divided into *simple subtractive colour mixing* and *complex subtractive colour mixing* [Ber00]. Simple subtractive mixing can be observed in materials such as coloured filters and liquids. It has been used in areas of colour photography [Oht71] and dyeing of transparent plastics [Bil63]. *Bouguer-Beer's law* is known as one of the simple subtractive colour mixing laws as it only involves the process of absorption.

Bouguer-Beer's law defines the logarithmic transformation of transmission for transparent mediums. The conception of the Bouguer-Beer law began in 1729, when Bouguer discovered the relationship between absorption and thickness through a series of experiments [Bou29]. This law was reiterated by Lambert thirty years later [Lam60] and was also established to be applicable to liquids of different concentrations by Beer a century after its first discovery [Bee52, Bee54]. Therefore the logarithmic transformation of transmission for transparent mediums can be known as *Bouguer-Beer's law* or *Lambert-Beer's law* after its discoverers.

The Bouguer-Beer's law defines the logarithmic transformation of transmission for transparent mediums, T_λ, such that [Ber00]:

$$\begin{aligned} -\log_{10}(T_{\lambda,i}) &= \log_{10}\left(\frac{1}{T_{\lambda,i}}\right) \\ &= A_\lambda \\ &= a_\lambda bc \end{aligned} \tag{5.8}$$

where λ is a wavelength and A_λ represents the absorbance of the medium. a_λ and b define the absorption co-efficient and the thickness of the medium, respectively, while c represents the concentration value of the colourant. The Bouguer-Beer's law can also be used as a colour mixing law whereby the absorbance mixture, $A_{\lambda,i,mix}$, is defined as [Ber00]:

$$\begin{aligned} A_{\lambda,i,mix} &= A_{\lambda,1} + A_{\lambda,2} + \ldots + A_{\lambda,n} \\ &= a_{\lambda,1}b_1c_1 + a_{\lambda,2}b_2c_2 + \ldots + a_{\lambda,n}b_nc_n \\ &= \sum_{i=1}^{n} a_{\lambda,i}b_ic_i \end{aligned} \tag{5.9}$$

where n represents the number of pigments. $A_{\lambda,i,mix}$ can be seen as the summation of the individual colourant. The combination of equations 5.8 and 5.9 provides us with the value of the internal transmittance, $T_{\lambda,i,mix}$, where: [Ber00]

$$\begin{aligned} \log_{10}\left(\frac{1}{T_{\lambda,i,mix}}\right) &= A_{\lambda,i,mix} \\ T_{\lambda,i,mix} &= 10^{-A_{\lambda,i,mix}}. \end{aligned}$$

Figure 5.9 shows the results of the implementation of the Bouguer-Beer's law using mixtures of the same and different colours.

One of the disadvantages of Bouguer-Beer's law is that the law is only valid for mediums with low and moderate concentrations. Bouguer-Beer's law can be applied to mediums with high concentrations values, but the results produced are only an approximation [Hun98]. The applications of the Bouguer-Beer's law have been investigated in computer graphics, such as in the rendering of diamonds [SFCD99] and gemstones [GS04].

5.6.3 Comparisons on the Colour Mixing Laws

This section presents a summary comparison of the colour mixing laws that has been implemented in the previous sections. The colour mixing laws are examined based on the mixing of two colours that are the same and the mixing of two different colours. The mixing of the same colour involves the mixing of red with red, green with green and blue with blue. The mixing of two different colours consist of the mixing of blue with red, green with blue and green with red. It is generally expected that when the same colour is mixed with itself the resulting mixture is of the same colour, while the mixture of two different colours will result in a new combination of colour.

Figure 5.7 shows the implementation of Grassmann's third law. In Figure 5.7(a), it can be observed that Grassmann's third law seems to have difficulty in handling the mixing of the same colour. It is generally expected that when two colours of the same colour are mixed together, the colour mixture would result in the same colour of the original inputs. However, this is not the case for Grassmann's third law of mixing. One possible reason is that in Grassmann's third law, when two colour coordinates are added together, their coordinates are joined together with the *origin* and the resulting coordinates would form a parallelogram which would result in a new colour coordinate. Another problem that could arise is that the colour mixture might be out of the visible range for a specific display devices, as it operates in an unbounded region.

Figure 5.8 shows the implementation of the blend operation in the 4-colour channel model. The blending operation demonstrates that it is able to cope with the mixing of different colours (see Figure 5.8(a)) and also mixing using the same input colours (see Figure 5.8(b)). One possible reason is that colours are weighted by their opacities. An advantage of the blending operation is that the resulting colour mixture is always displayable as it operates in a bounded space.

Figures 5.9 and 5.5 show the implementation of the Bouguer-Beer's law and Kubelka-Munk theory of colour mixing. Both of the implementations demonstrate that Bouguer-Beer's law and Kubelka-Munk theory are both able to handle the mixing of the same input colours and also different input colours. In using Bouguer-Beer's law and Kubelka-Munk theory, we are also able to utilise the physical properties of the objects, such as scattering and absorption. Although the results are illustrated in 2D, the colour mixing laws can be implemented and extended to 3D.

5.7 Summary

This chapter began with a review on the algebraic framework of Constructive Volume Geometry, for modelling complex spatial objects using combinational operations. In the chapter, definitions of the CVG algebra for the opacity channel model and 4-colour channel model were specified. The definitions are similar to that defined in the original paper. In addition, a set of laws was derived for each of the models. In this chapter, the CVG framework has been extended to include a spatial object based on the Kubelka-Munk model of absorption and scattering, which utilises the notion of vector fields.

In this chapter, a discussion on the different types of colour mixing laws was also carried out. The different types of colour mixing laws were separated into two categories of additive colour mixing and subtractive colour mixing. The discussion was supported with the implementation of four types of colour mixing laws. The discussion on the colour mixing laws allows us to experiment with colour mixing laws that utilise the physical properties of the objects.

Chapter 6

Algebra of Spatial Displacement Functions

Contents

6.1 Introduction 111
6.2 Related Work 112
6.3 Definitions and Concepts 113
6.4 Combinational SDFs and STFs 117
6.5 Hierarchical SDFs and STFs 124
6.6 Identity Problem 127
6.7 Inverse Problem 130
6.8 Direct Rendering Volume Scene Graphs 133
6.9 Summary 140

6.1 Introduction

The concept of a *spatial transfer function* (STF) was first introduced by Chen *et al.* in 2003 [CSW$^+$03] as a technique for volume modelling and animation. In their paper, Chen *et al.* also briefly touched upon the concept of a *spatial displacement function* (SDF) as a generic form of constructive operator. This chapter presents the results of further investigations into the concept of SDFs. In this chapter, SDFs are introduced as a fundamental building block of an algebraic framework for modelling complex deformation in a constructive manner and are shown to be algebraically and functionally superior to some existing schemes, including spatial transfer functions.

This chapter presents a comprehensive study of the concepts of spatial displacement functions and spatial transfer functions. The two spatial functions are examined in the context of their functionality and algebraic properties. Part of the work that is presented in the chapter has been produced in collaboration with a fellow research student, Shoukat Islam [1].

The organisation of this chapter is as follows:

- In Section 6.2, a brief overview of the techniques used in volume deformation and displacement mapping is presented.
- In Section 6.3, the definitions and concepts that are utilised throughout the chapter are given together with the relative merits that can be found between STFs and SDFs.
- In Sections 6.4 and 6.5, the various forms of representations for SDFs and STFs are discussed as well as their algebraic properties.
- In Sections 6.6 and 6.7, the two algebraic issues that can be encountered in SDFs and STFs are presented and analysed.
- In Section 6.8, the algorithms for rendering SDFs and STFs are discussed.

Finally in Section 6.9, our observations and concluding remarks are presented.

6.2 Related Work

This section presents a brief overview of previous studies that are related to the algebraic framework of spatial displacement functions. This section is divided into the two parts of *volume deformation* and *displacement mapping*.

6.2.1 Volume Deformation

A significant amount of research effort has been invested in the development of deformation techniques. These research efforts have been highlighted in a number of major surveys in the context of computer graphics and animation [GM97], animation of articulated figures [LW98], facial modeling and animation [NN98], medical image analysis [MT96], surgical simulation [LTCK03], computer vision [CF01] and more recently volume deformation and animation [CIJ$^+$05]. Some of these techniques have been extended to include volume representations as demonstrated by Barr [Bar84] and Sederberg and Parry [SP86]. Further extensions of some of the deformation techniques include the usage of a mesh structure over volumetric data [Bro98].

The subject of deformation techniques with respect to volumetric representations is an active research topic, with a new collection of approaches developed nearly every year, with the focus on applications in volume visualisation and animation. A brief summary of the general work that has been implemented in volume deformation is as follows:

- In 1995, Kurzion and Yagel [KY95] presented the usage of local transformations in the form of 'ray deflectors' for modelling and rendering volume deformation;

[1]cshoukat@swansea.ac.uk

- Lerios *et al.* [LGL95] and Chen *et al.* [CJT96], later demonstrated the application of feature-based techniques for controlling volume deformation;
- The development of physically-based and physically-plausible volume deformation models was introduced by Gibson [Gib97], Chen *et al.* [CZK98] and Müller *et al.* [MDM+02];
- The important part that volume deformation plays in volume animation was then illustrated by Gagvani and Silver [GS01] and Wu and Prakash [WP00]; and
- Recently, Chen *et al.* [CSW+03] introduced the concept of spatial transfer functions together with its application in volume visualisation, deformation, sweeping and animation.

A number of specific works have also been implemented in the area of volume deformation. They are deformation-based volume browsing [MTB03], volume splitting [IDSC04], deformation of distance fields [SOM04, SPC03] and hardware-assisted volume deformation [WS01, RSSG01].

6.2.2 Displacement Mapping

In 1984, Cook introduced a technique called *displacement mapping* as an approach to modifying the geometry of a surface [Coo84]. The difference between Cook's approach and *bump mapping* [Bli78] is that the former technique has the capability to compute correct shadows and silhouettes, whereas the latter technique only alters the normals. Implementations of displacement mapping generally involve the subdivision of geometric primitives into micropolygons [CCC87] and ray tracing [Tai92, PHL95, PH96, SSS00]. A more recent development on displacement mapping can be seen in image-based rendering [SP99, Die00, KS01], hardware-assisted rendering [GH99, DH00, KS01] and subdivision surfaces [LMH00].

Some of the other work found relevant to displacement mapping include horizon mapping [Max88], enhanced bump mapping [HDKS00], relief texture mapping [OBM00], bidirectional texture mapping [DNvK99], polynomial texture mapping [MGW01] and view-dependent displacement mapping [WWT+03].

6.3 Definitions and Concepts

In this section, the concepts of *spatial displacement functions* as an algebraic framework for modelling a complex deformation in a constructive manner are introduced. This section begins by outlining the explicit and precise definitions of the notations that are utilised throughout this chapter. Later in the section, a demonstration of how a spatial displacement function can be defined and utilised as a *composite object* is given. Finally, the relative merits of spatial displacement functions and *spatial transfer functions* [CSW+03] are discussed.

6.3.1 Definitions

Let $\mathbb{R}$ denote the set of all real numbers, and $\mathbb{E}^3$ denote 3D Euclidean space.

Definition 1 (Scalar field). *A* scalar field *is a function* $F : \mathbb{E}^3 \rightarrow \mathbb{R}$.

A volumetric representation of an object can be specified as a set of *scalar fields*, A_0, A_1, $\ldots, A_k$, that defines the geometrical and physical properties of the object at every point p in $\mathbb{E}^3$.

Definition 2 (Spatial object). *A* spatial object *is a tuple,* $\mathbf{o} = \lambda p.(A_0(p), A_1(p), \ldots, A_k(p))$ *of scalar fields defined at every point p in the* $\mathbb{E}^3$*, where* $k \geq 0$.

A spatial object can be split into two groups, namely a *conventional spatial object* or a *non-conventional spatial object*. In a conventional spatial object, $A_i (i = 0, 1, \ldots, k)$ are used to specify its *attribute fields*, such as opacity, color components, refraction index, and so on, in a heterogeneous (*i.e.*, non-uniform) manner. In a non-conventional object, however, its scalar fields are not being used to define its physical properties. In this chapter, two kinds of non-conventional objects are to be discussed, namely *spatial transfer object* [CSW+03] and *spatial displacement object*.

Definition 3 (Spatial transfer object). *A* spatial transfer object *(or just* transfer object *for short) is a tuple,* $\mathbf{s} = (\Psi_x, \Psi_y, \Psi_z)$*, of scalar fields that defines the geometrical transformation of every point* $p \in \mathbb{E}^3$*, that is,*

$$\mathbf{s}(p) = \Psi(p) = (\Psi_x(p), \Psi_y(p), \Psi_z(p))$$

where $\Psi : \mathbb{E}^3 \rightarrow \mathbb{R}^3$ *and* $\Psi_x, \Psi_y, \Psi_z : \mathbb{E}^3 \rightarrow \mathbb{R}$.

Definition 4 (Spatial transfer function). *Given a spatial transfer object,* $\mathbf{s} = (\Psi_x, \Psi_y, \Psi_z)$*, and a conventional spatial object,* $\mathbf{o} = (A_0, A_1, \ldots, A_k)$*, a* spatial transfer function (STF) *defines a composite object,* $\mathbf{o}'$*, such that* $\forall p \in \mathbb{E}^3$*, we have:*

$$\mathbf{o}'(p) = \mathbf{T_s}(\mathbf{o})(p) = (A_0(\mathbf{s}(p)), A_1(\mathbf{s}(p)), \ldots, A_k(\mathbf{s}(p))) \tag{6.1}$$

where $\mathbf{T_s} : \mathbf{O} \rightarrow \mathbf{O}$ *and* $\mathbf{s}(p) = (\Psi_x(p), \Psi_y(p), \Psi_z(p))$. $\mathbf{O}$ *is defined as a class of spatial objects.*

Under the theoretical framework of Constructive Volume Geometry (CVG) [CT00], such an operator can be specified as an algebraic term $\mathbf{T_s}(\mathbf{o}) = \boxed{\mapsto}(\mathbf{s}, \mathbf{o})$.

Definition 5 (Spatial displacement object). *A* spatial displacement object *(or just* displacement object *for short) is a tuple,* $\mathbf{d} = (\psi_x, \psi_y, \psi_z)$*, of scalar fields that specify the displacement of every point* $p \in \mathbb{E}^3$ *of an* operand object $\mathbf{o}$*, that is,*

$$\mathbf{d}(p) = \psi(p) = (\psi_x(p), \psi_y(p), \psi_z(p))$$

where $\psi : \mathbb{E}^3 \rightarrow \mathbb{R}^3$ *and* $\psi_x, \psi_y, \psi_z : \mathbb{E}^3 \rightarrow \mathbb{R}$.

Definition 6 (Spatial displacement function). *Given a displacement object,* $\mathbf{d} = (A_x, A_y, A_z)$*, and a conventional spatial object,* $\mathbf{o} = (A_0, A_2, \ldots, A_k)$*, a* spatial displacement function (SDF) *defines a composite object,* $\mathbf{o}'$*, such that* $\forall p \in \mathbb{E}^3$*, we have:*

$$\mathbf{o}'(p) = \mathbf{D_d}(\mathbf{o})(p) = \mathbf{o}(p + \mathbf{d}(p)) = (A_0(p + \mathbf{d}(p)), \ldots, A_k(p + \mathbf{d}(p))) \tag{6.2}$$

where $\mathbf{D_d} : \mathbf{O} \rightarrow \mathbf{O}$.

Once more, the theoretical framework of CVG can be utilised again, whereby such an operation can be denoted as an algebraic term $\mathbf{D_d}(\mathbf{o}) = \boxed{\|\triangleright}(\mathbf{d}, \mathbf{o})$, and in practice can be implemented as a sub-tree in a volume scene graph.

Since each displacement object is essentially a 3D vector field, many algebraic operations on vector fields can be applied to such objects. For example, one operation used extensively in this work is $\boxed{+}$, which is a volumetric vector field operation based on point-wise vector addition. For instance, given two displacement objects, $\mathbf{d}_0 = (\psi_{x,0}, \psi_{y,0}, \psi_{z,0})$ and $\mathbf{d}_1 = (\psi_{x,1}, \psi_{y,1}, \psi_{z,1})$, we have $\mathbf{d} = \boxed{+}(\mathbf{d}_0, \mathbf{d}_1)$, such that, for $p \in \mathbb{E}^3$,

$$\begin{aligned}
\mathbf{d}(p) &= (\psi_x(p), \psi_y(p), \psi_z(p)) = \boxed{+}(\mathbf{d}_0, \mathbf{d}_1)(p) \\
&= \lambda p.(A_0(p + (\psi_{x,0}(p) + \psi_{x,1}(p),\ \psi_{y,0}(p) + \psi_{y,1}(p),\ \psi_{z,0}(p) + \psi_{z,1}(p))), \\
&\qquad \ldots, A_k(p + (\psi_{x,0}(p) + \psi_{x,1}(p),\ \psi_{y,0}(p) + \psi_{y,1}(p),\ \psi_{z,0}(p) + \psi_{z,1}(p)))).
\end{aligned}$$

Several other constructive operations are discussed later on in the chapter.

The following definitions of transformation, translation and scaling identities are also stated and their applications are demonstrated in the following sections.

Definition 7 (Transformation identity). *A spatial transfer object is called the* transformation identity, $\mathbf{s}^{\tau} : \mathbb{E}^3 \rightarrow \mathbb{R}^3$, *such that* $\forall p \in \mathbb{E}^3$, *we have:*

$$p' = \mathbf{s}^{\tau}(p) = \lambda p.p = p.$$

Definition 8 (Translation identity). *A spatial displacement object is called the* translation identity, $\mathbf{d}^{\iota} : \mathbb{E}^3 \rightarrow \mathbb{R}^3$, *such that* $\forall p \in \mathbb{E}^3$, *there is,*

$$\mathbf{d}^{\iota}(p) = \lambda p.(0, 0, 0).$$

Definition 9 (Scaling identity). *A spatial displacement object is called a* scaling identity, $\mathbf{d}^{\zeta} : \mathbb{E}^3 \rightarrow \mathbb{R}^3$, *such that* $\forall p \in \mathbb{E}^3$, *we have:*

$$\mathbf{d}^{\zeta}(p) = \lambda p.(1, 1, 1).$$

6.3.2 Regularisation

It is often possible in a surface deformation scheme to ensure that every surface point on a deformed object is backward-mapped onto a point on the original object. It is, however, neither desirable nor practical to impose this condition on spatial objects, whether they are bounded or unbounded. Consider a hypothetical but algebraically meaningful requirement for a displacement object $\mathbf{d}$, which makes any arbitrary spatial object $\mathbf{o}$ completely vanish from $\mathbb{E}^3$. The resulting object, $\mathbf{o}' = \boxed{\|\triangleright}(\mathbf{d}, \mathbf{o})$, would still be defined in $\mathbb{E}^3$, but has no single point that is 'visible' to a renderer.

This requirement can be facilitated by the introduction of a special *empty value*, denoted as $\varnothing$, in the specification of $\psi_x(p)$, $\psi_y(p)$ and $\psi_z(p)$. A displacement object $\mathbf{d} = (\psi_x, \psi_y, \psi_z)$

is said to make a point p vanish if $(\psi_x(p) = \varnothing) \vee (\psi_y(p) = \varnothing) \vee (\psi_z(p) = \varnothing)$. In terms of the backward mapping in Equation 6.2, this implies $p + \mathbf{d}(p) \notin \mathbb{E}^3$, and $\mathbf{o}'(p)$ becomes invalid or undefined. In practice, $\mathbf{o}'(p)$ can be considered to be fully transparent by a renderer, or if $\mathbf{o}'$ is bounded, p can be treated in a similar way as any points that do not belong to $\mathbf{o}'$.

The special value $\varnothing$ can be considered to be particularly useful in the specifications of many displacement objects. Its usage is further discussed in later sections in conjunction with some examples. The inclusion of $\varnothing$, however, complicates the design of constructive operations on these objects. This is due to the fact that traditional algebraic operations on vector fields do not feature $\varnothing$, which thereby involves a special treatment in the application of constructive operations on displacement objects. This special treatment is called a *regularisation* process. The regularisation process bears a number of similarities to the regularisation step in Constructive Solid Geometry (CSG) [Req77]. Their similarities can be seen in the aspects of pragmatics instead of functionality. Given a scalar operation **op** on n scalar values, $a_0, \ldots, a_n \in \mathbb{R}$, a regularised scalar operation **op**$*$, corresponding to **op** but in $\mathbb{R} \cup \varnothing$, is essentially:

$$\mathbf{op}*(a_0, \ldots, a_n) = \begin{cases} \mathtt{REG}(\mathbf{op}, a_0, \ldots, a_n) & \text{if } \exists i \in [0, n],\ a_i = \varnothing \\ \mathbf{op}(a_0, \ldots, a_n) & \text{if } \forall i \in [0, n],\ a_i \in \mathbb{R} \end{cases}$$

where **REG** is an operation-dependent regularisation process. Normally, **REG** may apply **op** to a subset of $a_0, \ldots, a_n$ valid in $\mathbb{R}$, or may result in a $\varnothing$ value or the first a_i value valid in $\mathbb{R}$.

It is worth noting that the empty value $\varnothing$ cannot be assumed to have the same feature or functionality as infinity ∞. This treatment is needed in order to prevent any potential confusion between our regularisation process and the well-established algebras for indeterminate forms in calculus.

6.3.3 Relative Merits

This section examines the relative merits between the concepts of SDFs and *spatial transfer functions* (STFs) [CSW+03]. STF was proposed by Chen *et al.* in 2003 as an algebraic framework for volume modelling and animation. A STF can be specified as a SDF by defining the deformation of a point p as:

$$\mathbf{o}'(p) = \mathbf{o}(\mathbf{s}(p)) = (A_0(\mathbf{s}(p)), \ldots, A_k(\mathbf{s}(p))),$$

where $\mathbf{s}$ is a mapping specification from p to p'. In comparison with Equation 6.2, a similar definition can also be specified for STF, such that,

$$p' = \mathbf{s}(p) = p + \mathbf{d}(p).$$

Although STFs are easy to specify in some applications, such as the assignment of captured motion to a volume object $\mathbf{o}$ via $\mathbf{s}$, they have very limited algebraic properties.

For example, consider a constant displacement object $\mathbf{d} = (1, 0, 0)$, which moves any $p \in \mathbb{E}^3$ by 1 in the x-direction [2]. In a SDF application, the combination of two identical copies of $\mathbf{d}$ using the above-mentioned operation $\boxplus$, prior to their application to another object $\mathbf{o}$, implies a rather intuitive outcome,

$$p' = p + 2 \cdot \mathbf{d}(p) = p + (2, 0, 0)$$

that is, for any $p \in \mathbb{E}^3$, p is transformed with twice the amount of the displacement defined by $\mathbf{d}$. In a STF application, however, this would have required

$$p' = 2 \cdot \mathbf{s}(p) = 2 \cdot p + 2 \cdot \mathbf{d}(p) = 2 \cdot p + (2, 0, 0).$$

In general, STFs are only suitable to be used in a sequential manner, and the specification of deformation $\mathbf{s}$ normally assumes that $p \in [0, 1]^3$. It is therefore generally difficult to use STFs for local deformation, or apply constructive operations to their specification.

In contrast, $\mathbf{o}$, which defines the relative displacement, exhibits many useful features that $\mathbf{s}$ does not. Like other ordinary spatial objects, conventional transfer functions can be used to define or modify attribute fields of $\mathbf{o}$, and constructive and combinational operations, such as addition, scaling, min-max functions and blending, can also be applied to such objects.

While SDF conceptually bears some resemblance to displacement mapping for surface modeling, its relative merits are reflected in its generality and flexibility. There is no constraint on which points can be deformed, and no constraint on where a point can be deformed to. For example, a folding effect would introduce a great deal of complexity in surface-based displacement mapping, in terms of both polygon subdivision and ray-displacement intersection. It is the main reason why displacement mapping is usually deployed only for modeling surface details with relatively 'small' displacement. Having the concept of a SDF defined in the volume domain, displacement can now be modelled and rendered based entirely on a point-to-point basis, and therefore can achieve more complex deformation, as illustrated in the following sections.

6.4 Combinational SDFs and STFs

There are a number of approaches in implementing SDFs and STFs. One such approach is *combinational* SDFs and STFs, which is explored in this section. This section begins with the evaluation of the concepts of combinational SDFs and STFs together with their applications. Later, an overview of the algebraic properties of both combinational SDFs and STFs is presented and an analysis of their algebraic properties is given.

Definition 10 (Combinational SDF). *A* combinational *SDF is a volume scene graph where a set of displacement objects, $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_n$, are first combined together using a constructive operation, and then applied to $\mathbf{o}$. An algebraic specification for such a SDF is:*

$$\boxed{\Vdash\!\!\!\rightarrow}(\Phi(\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_n), \mathbf{o}),$$

where Φ is an arbitrary n-ary operation, which can also be decomposed into a set of constructive operations in the form of a sub-tree.

[2] This actually implies moving an operand object by -1 in the x-direction due to backward mapping.

Name	Description	Definition
Inv	inverse vector, an unary operation.	$(-\psi_x(p), -\psi_y(p), -\psi_z(p))$.
Add	vector addition.	$(\psi_x(p) + \psi_x(p), \psi_y(p) + \psi_y(p), \psi_z(p) + \psi_z(p))$.
Diff	vector subtraction.	$(\psi_x(p) - \psi_x(p), \psi_y(p) - \psi_y(p), \psi_z(p) - \psi_z(p))$.
Times	component-based scalar multiplication.	$(\psi_x(p) \times \psi_x(p), \psi_y(p) \times \psi_y(p), \psi_z(p) \times \psi_z(p))$.
Div	component-based scalar division.	$\frac{\psi_i(p)}{\psi_i(p)}$ if $\psi_i(p) \neq 0$, $i = x, y, z$, otherwise $\varnothing$.
Avg	component-based scalar average.	$(\frac{\psi_x(p)+\psi_x(p)}{2}, \frac{\psi_y(p)+\psi_y(p)}{2}, \frac{\psi_z(p)+\psi_z(p)}{2})$.
Min	component-based minimum scalar value.	$(\min(\psi_i(p), \psi_i(p))$, $i = x, y, z$.
Max	component-based maximum scalar value.	$(\max(\psi_i(p), \psi_i(p))$, $i = x, y, z$.
MinAbs	choosing a value according to the minimum absolute value of each component.	$\psi_i(p)$ if $\lvert\psi_i(p)\rvert \leq \lvert\psi_i(p)\rvert$, otherwise $\psi_i(p)$, $i = x, y, z$.
MaxAbs	choosing a value according to the maximum absolute value of each component.	$\psi_i(p)$ if $\lvert\psi_i(p)\rvert \geq \lvert\psi_i(p)\rvert$, otherwise $\psi_i(p)$, $i = x, y, z$.
MinMag	choosing a value according to the minimum vector magnitude.	$\psi_i(p)$ if $\lVert\psi_x(p), \psi_y(p), \psi_z(p)\rVert \leq \lVert\psi_x(p), \psi_y(p), \psi_z(p)\rVert$, otherwise $\psi_i(p)$, $i = x, y, z$.
MaxMag	choosing a value according to the maximum vector magnitude.	$\psi_i(p)$ if $\lVert\psi_x(p), \psi_y(p), \psi_z(p)\rVert \geq \lVert\psi_x(p), \psi_y(p), \psi_z(p)\rVert$, otherwise $\psi_i(p)$, $i = x, y, z$.

Table 6.1: Examples of constructive operations on displacement objects. Except for *Inv*, they are all point-wise binary operations $\Phi(\mathbf{a}, \mathbf{b})$, where $\mathbf{a} = (\psi_x, \psi_y, \psi_z)$ and $\mathbf{b} = (\psi_x, \psi_y, \psi_z)$.

Given the fact that $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_n$ are essentially vector fields (refer to Section 6.3), a rich collection of constructive operations on displacement objects can be generated. One of the most important factors that encourage the development of combinational SDFs is that many such operations are able to satisfy the laws of vector algebra.

Definition 11 (Combinational STF). *A combinational STF is a volume scene graph where a set of transfer objects, $\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_n$, are first combined together using a constructive operation, and then applied to an operand object. An algebraic specification for such a STF is:*

$$\boxed{\mapsto}(\Phi(\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_n), \mathbf{o}),$$

where Φ is an arbitrary n-ary operation.

As an example, Figures 6.1(b),(d),(f) and (h) show a point-wise summation of three displacement objects, **wavel**, **wave2** and **wave3**, using a binary addition operation $\boxplus$ constructively. **wavel**, **wave2** and **wave3**, where $p \in [0,1]^3$ are defined as follows:

$$\textbf{wavel} = (\psi_x(p_x), \psi_y(p_y), \psi_z(p_z)),$$

where

$$\psi_x(p_x) = 0; \quad \psi_y(p_y) = \frac{(1.5)\sin\theta + 1}{2}; \quad \text{and} \quad \psi_z(p_z) = 0.$$

$$\textbf{wave2} = (\psi_x(p_x), \psi_y(p_y), \psi_z(p_z)),$$

where

$$\psi_x(p_x) = 0; \quad \psi_y(p_y) = \frac{\sin\theta + 1}{4}; \quad \text{and} \quad \psi_z(p_z) = 0.$$

$$\textbf{wave3} = (\psi_x(p_x), \psi_y(p_y), \psi_z(p_z)),$$

where

$$\psi_x(p_x) = 0; \quad \psi_y(p_y) = \frac{\sin\theta + 1}{22}; \quad \text{and} \quad \psi_z(p_z) = 0.$$

As $\boxplus$ satisfies the commutative and associative laws, users are able to intuitively achieve the effectual results. Several of the combinational SDF operations defined and implemented in this chapter are listed in Table 6.1. Figures 6.1(c),(e),(g) and (i) also show examples of a point-wise summation of three transfer objects, **wavel′**, **wave2′** and **wave3′**, using a binary addition operation $\boxplus$ constructively. **wavel′**, **wave2′** and **wave3′**, where $p \in [0,1]^3$ are defined as follows:

$$\textbf{wavel}' = (\Psi_x(p_x), \Psi_y(p_y), \Psi_z(p_z)),$$

where

$$\Psi_x(p_x) = p_x; \quad \Psi_y(p_y) = \frac{(1.5)\sin\theta + 1}{2} + p_y; \quad \text{and} \quad \Psi_z(p_z) = p_z.$$

$$\textbf{wave2}' = (\Psi_x(p_x), \Psi_y(p_y), \Psi_z(p_z)),$$

where

$$\Psi_x(p_x) = p_x; \quad \Psi_y(p_y) = \frac{\sin\theta + 1}{4} + p_y; \quad \text{and} \quad \Psi_z(p_z) = p_z.$$

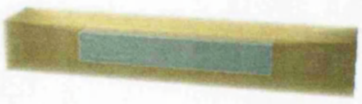
$$\textbf{wave3}' = (\Psi_x(p_x), \Psi_y(p_y), \Psi_z(p_z)),$$

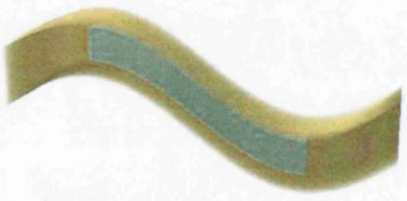
where

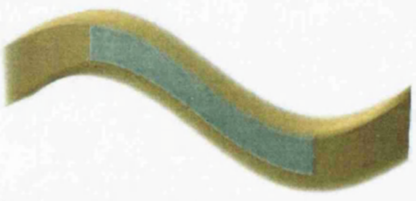
$$\Psi_x(p_x) = p_x; \quad \Psi_y(p_y) = \frac{\sin\theta + 1}{22} + p_y; \quad \text{and} \quad \Psi_z(p_z) = p_z.$$

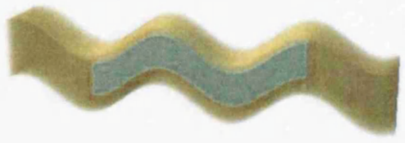
The implementation of the point-wise summation of the three transfer objects resulted in a different result than that of the three displacement objects (see Figures 6.1(h) and (i)). This is because in the transfer function, the points of the object are added up instead of their relative distance forcing the resulting object to be small. One possible solution to this problem would be remapping. However, such a solution would be difficult to implement as the values to be remapped are unknown. In parallel to Table 6.1, Table 6.2 also lists several of the combinational STF operations that can be implemented and are defined in this chapter.

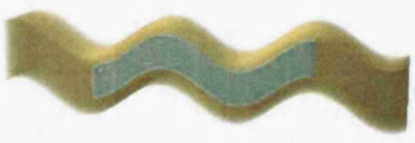
Another useful constructive operation that can be found among these operations is scalar multiplication $\boxed{*}$. The scalar multiplication $\boxed{*}$ scales the magnitude of the displacement vector at each point, and this in effect applies a transfer function to a displacement object based on a spatial location. In conjunction with an ordinary transfer function, it can be used to perform deformation on part of an operand object selected according to data values. Such an example can be seen in Figure 6.2(c) that exhibits a similar action to that of Figure 6.3(e), although the application of displacement is only limited to the external shell of $\mathbf{c}$. In order to achieve this effect, a masking object $\mathbf{c}_m$ can be defined by applying a simple lookup transfer to $\mathbf{c}$. The attribute fields of $\mathbf{c}_m$, $\psi_x(p), \psi_y(p), \psi_z(p)$, are set to 1 if the opacity (or data) field of $\mathbf{c}$ indicates that p is on the external shell of $\mathbf{c}$, and 0 otherwise. A volume scene graph $\boxed{\mapsto}(\boxed{*}(\mathbf{c}_m, \textbf{open}), \mathbf{c})$ achieves a masked opening displacement. Without much extra effort, a similar volume scene graph for a different volume object can be constructed, for instance, a spheric object in Figure 6.2(d). Similar operations can also be simulated using STF operations, as demonstrated in Figures 6.2(e) and (f).

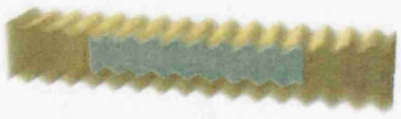

(a) $\mathbf{c}$

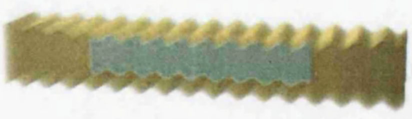

(b) $\boxed{\Mapsto}(\mathbf{wave1}, \mathbf{c})$


(c) $\boxed{\mapsto}(\mathbf{wave1'}, \mathbf{c})$


(d) $\boxed{\Mapsto}(\mathbf{wave2}, \mathbf{c})$


(e) $\boxed{\mapsto}(\mathbf{wave2'}, \mathbf{c})$


(f) $\boxed{\Mapsto}(\mathbf{wave3}, \mathbf{c})$


(g) $\boxed{\mapsto}(\mathbf{wave3'}, \mathbf{c})$

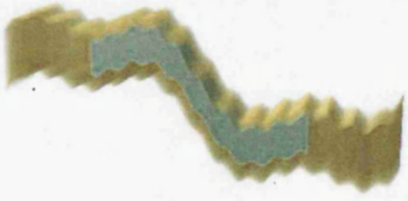
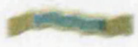
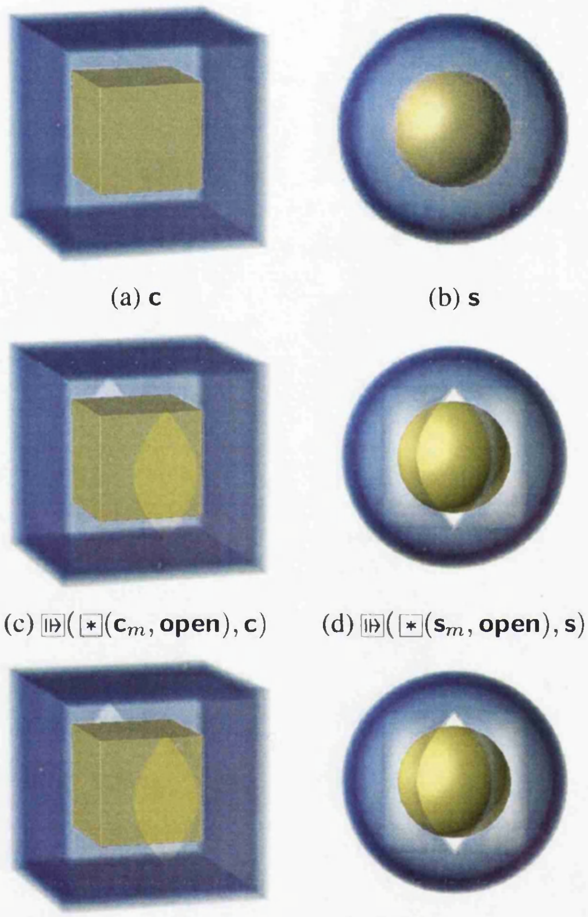


(h) $\boxed{\Mapsto}(\boxed{+}(\boxed{+}(\mathbf{wave1}, \mathbf{wave2}), \mathbf{wave3}), \mathbf{c})$ (i) $\boxed{\mapsto}(\boxed{+}(\boxed{+}(\mathbf{wave1'}, \mathbf{wave2'}), \mathbf{wave3'}), \mathbf{c})$

Figure 6.1: Examples of combining three SDFs and STFs, **wave1**, **wave1′**, **wave2**, **wave2′**, **wave3**, and **wave3′**, using a CVG operation based on vector addition, and applying the composite SDF and STF to a scaled cubic volume object, **c**.



(a) $\mathbf{c}$ (b) $\mathbf{s}$

(c) $\boxed{\Vert\!\mapsto}(\boxed{*}(\mathbf{c}_m, \mathbf{open}), \mathbf{c})$ (d) $\boxed{\Vert\!\mapsto}(\boxed{*}(\mathbf{s}_m, \mathbf{open}), \mathbf{s})$

(e) $\boxed{\mapsto}(\boxed{*}(\mathbf{c}_m, \mathbf{open}), \mathbf{c})$ (f) $\boxed{\mapsto}(\boxed{*}(\mathbf{s}_m, \mathbf{open}), \mathbf{s})$

Figure 6.2: (c)-(d) Examples of using the *Times* operation to 'mask' a displacement object **open** prior to its application to a cubic volume object **c** and a spheric volume object **s**. The same lookup table was used to create $\mathbf{c}_m$ and $\mathbf{s}_m$. Because the displacement involves motion only in the x and y directions, the effects on the continuity of normals and thickness of the shell become noticeable on **s**. Despite such effects, the examples demonstrate that one can apply a mask to a displacement object according to some segmentation information. (e)-(f) Similar operations that are performed for SDFs can also be simulated with STFs operations.

Name	Description	Definition
⊞ *Add*	vector addition.	$(\Psi_x(p) + \Psi_x(p), \Psi_y(p) + \Psi_y(p), \Psi_z(p) + \Psi_z(p))$.
⊟ *Diff*	vector subtraction.	$(\Psi_x(p) - \Psi_x(p), \Psi_y(p) - \Psi_y(p), \Psi_z(p) - \Psi_z(p))$.
⊛ *Times*	component-based scalar multiplication.	$(\Psi_x(p) \times \Psi_x(p), \Psi_y(p) \times \Psi_y(p), \Psi_z(p) \times \Psi_z(p))$.
⌹ *Div*	component-based scalar division.	$\frac{\Psi_i(p)}{\Psi_i(p)}$ if $\Psi_i(p) \neq 0$, $i = x, y, z$, otherwise $\varnothing$.
⊠ *Avg*	component-based scalar average.	$(\frac{\Psi_x(p)+\Psi_x(p)}{2}, \frac{\Psi_y(p)+\Psi_y(p)}{2}, \frac{\Psi_z(p)+\Psi_z(p)}{2})$.
▤ *Min*	component-based minimum scalar value.	$(\min(\Psi_i(p), \Psi_i(p)), i = x, y, z$.
▤ *Max*	component-based maximum scalar value.	$(\max(\Psi_i(p), \Psi_i(p)), i = x, y, z$.
◧ *MinAbs*	choosing a value according to the minimum absolute value of each component.	$\Psi_i(p)$ if $\lvert\Psi_i(p)\rvert \leq \lvert\Psi_i(p)\rvert$, otherwise $\Psi_i(p)$, $i = x, y, z$.
◨ *MaxAbs*	choosing a value according to the maximum absolute value of each component.	$\Psi_i(p)$ if $\lvert\Psi_i(p)\rvert \geq \lvert\Psi_i(p)\rvert$, otherwise $\Psi_i(p)$, $i = x, y, z$.
◧ *MinMag*	choosing a value according to the minimum vector magnitude.	$\Psi_i(p)$ if $\lVert\Psi_x(p), \Psi_y(p), \Psi_z(p)\rVert \leq \lVert\Psi_x(p), \Psi_y(p), \Psi_z(p)\rVert$, otherwise $\Psi_i(p)$, $i = x, y, z$.
◨ *MaxMag*	choosing a value according to the maximum vector magnitude.	$\Psi_i(p)$ if $\lVert\Psi_x(p), \Psi_y(p), \Psi_z(p)\rVert \geq \lVert\Psi_x(p), \Psi_y(p), \Psi_z(p)\rVert$, otherwise $\Psi_i(p)$, $i = x, y, z$.

Table 6.2: Examples of constructive operations on transfer objects. The operations are all point-wise binary operations $\Phi(\mathbf{a}, \mathbf{b})$, where $\mathbf{a} = (\Psi_x, \Psi_y, \Psi_z)$ and $\mathbf{b} = (\Psi_x, \Psi_y, \Psi_z)$.

6.4.1 Algebraic Properties of Combinational SDFs and STFs

This section presents an overview of the algebraic properties of the operations defined in Tables 6.1 and 6.2 on displacement and transfer objects. As displacement objects, $\mathbf{d}_0$, $\mathbf{d}_1$, ..., $\mathbf{d}_n$, are basically vector fields, many of the operations defined on the displacement object satisfy the laws of vector algebra, which can be seen in Table 6.3. Let $\mathbf{d}_1 = (\psi_{x,1}, \psi_{y,1}, \psi_{z,1})$, $\mathbf{d}_2 = (\psi_{x,2}, \psi_{y,2}, \psi_{z,2})$ and $\mathbf{d}_3 = (\psi_{x,3}, \psi_{y,3}, \psi_{z,3})$ be defined as displacement objects, and $\mathbf{s}_1 = (\Psi_{x,1}, \Psi_{y,1}, \Psi_{z,1})$, $\mathbf{s}_2 = (\Psi_{x,2}, \Psi_{y,2}, \Psi_{z,2})$ and $\mathbf{s}_3 = (\Psi_{x,3}, \Psi_{y,3}, \Psi_{z,3})$ be specified as transfer objects, a general set of algebraic properties can be obtained for both combinational SDFs and STFs. Table 6.3 shows the algebraic properties for both combinational SDFs and STFs.

Laws	SDFs	STFs
Commutative	$\boxed{+}(\mathbf{d}_1,\mathbf{d}_2) = \boxed{+}(\mathbf{d}_2,\mathbf{d}_1)$ $\boxed{*}(\mathbf{d}_1,\mathbf{d}_2) = \boxed{*}(\mathbf{d}_2,\mathbf{d}_1)$ $\boxed{\times}(\mathbf{d}_1,\mathbf{d}_2) = \boxed{\times}(\mathbf{d}_2,\mathbf{d}_1)$ $\boxed{\underline{=}}(\mathbf{d}_1,\mathbf{d}_2) = \boxed{\underline{=}}(\mathbf{d}_2,\mathbf{d}_1)$ $\boxed{\overline{=}}(\mathbf{d}_1,\mathbf{d}_2) = \boxed{\overline{=}}(\mathbf{d}_2,\mathbf{d}_1)$	$\boxed{+}(\mathbf{s}_1,\mathbf{s}_2) = \boxed{+}(\mathbf{s}_2,\mathbf{s}_1)$ $\boxed{*}(\mathbf{s}_1,\mathbf{s}_2) = \boxed{*}(\mathbf{s}_2,\mathbf{s}_1)$ $\boxed{\times}(\mathbf{s}_1,\mathbf{s}_2) = \boxed{\times}(\mathbf{s}_2,\mathbf{s}_1)$ $\boxed{\underline{=}}(\mathbf{s}_1,\mathbf{s}_2) = \boxed{\underline{=}}(\mathbf{s}_2,\mathbf{s}_1)$ $\boxed{\overline{=}}(\mathbf{s}_1,\mathbf{s}_2) = \boxed{\overline{=}}(\mathbf{s}_2,\mathbf{s}_1)$
Associative	$\boxed{+}(\mathbf{d}_1,\boxed{+}(\mathbf{d}_2,\mathbf{d}_3)) = \boxed{+}(\boxed{+}(\mathbf{d}_1,\mathbf{d}_2),\mathbf{d}_3)$ $\boxed{*}(\mathbf{d}_1,\boxed{*}(\mathbf{d}_2,\mathbf{d}_3)) = \boxed{*}(\boxed{*}(\mathbf{d}_1,\mathbf{d}_2),\mathbf{d}_3)$ $\boxed{\underline{=}}(\mathbf{d}_1,\boxed{\underline{=}}(\mathbf{d}_2,\mathbf{d}_3)) = \boxed{\underline{=}}(\boxed{\underline{=}}(\mathbf{d}_1,\mathbf{d}_2),\mathbf{d}_3)$ $\boxed{\overline{=}}(\mathbf{d}_1,\boxed{\overline{=}}(\mathbf{d}_2,\mathbf{d}_3)) = \boxed{\overline{=}}(\boxed{\overline{=}}(\mathbf{d}_1,\mathbf{d}_2),\mathbf{d}_3)$	$\boxed{+}(\mathbf{s}_1,\boxed{+}(\mathbf{s}_2,\mathbf{s}_3)) = \boxed{+}(\boxed{+}(\mathbf{s}_1,\mathbf{s}_2),\mathbf{s}_3)$ $\boxed{*}(\mathbf{s}_1,\boxed{*}(\mathbf{s}_2,\mathbf{s}_3)) = \boxed{*}(\boxed{*}(\mathbf{s}_1,\mathbf{s}_2),\mathbf{s}_3)$ $\boxed{\underline{=}}(\mathbf{s}_1,\boxed{\underline{=}}(\mathbf{s}_2,\mathbf{s}_3)) = \boxed{\underline{=}}(\boxed{\underline{=}}(\mathbf{s}_1,\mathbf{s}_2),\mathbf{s}_3)$ $\boxed{\overline{=}}(\mathbf{s}_1,\boxed{\overline{=}}(\mathbf{s}_2,\mathbf{s}_3)) = \boxed{\overline{=}}(\boxed{\overline{=}}(\mathbf{s}_1,\mathbf{s}_2),\mathbf{s}_3)$
Indempotent	$\boxed{\times}(\mathbf{d}_1,\mathbf{d}_1) = \mathbf{d}_1$ $\boxed{\underline{=}}(\mathbf{d}_1,\mathbf{d}_1) = \mathbf{d}_1$ $\boxed{\overline{=}}(\mathbf{d}_1,\mathbf{d}_1) = \mathbf{d}_1$	$\boxed{\times}(\mathbf{s}_1,\mathbf{s}_1) = \mathbf{s}_1$ $\boxed{\underline{=}}(\mathbf{s}_1,\mathbf{s}_1) = \mathbf{s}_1$ $\boxed{\overline{=}}(\mathbf{s}_1,\mathbf{s}_1) = \mathbf{s}_1$

Table 6.3: Algebraic properties of operations on displacement and transfer objects.

In Table 6.3, it can be seen that not all of the algebraic laws for SDFs and STFs have been stated, such as identities and inverse laws. One the reasons for these omissions is that identities and inverse laws for both SDFs and STFs require further discussion and therefore can be found later in Sections 6.6 and 6.7.

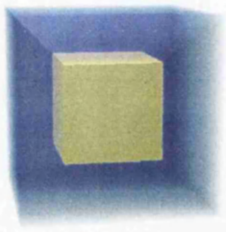
6.5 Hierarchical SDFs and STFs

A different approach to combinational SDFs and STFs is *hierarchical* SDFs and STFs.

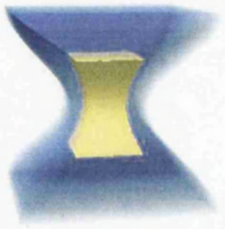
Definition 12 (Hierarchical SDF). *A* hierarchical *SDF is a volume scene graph where a series of displacement objects,* $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_n$, *are applied to an operand object* $\mathbf{o}$ *in a sequential order. An algebraic specification for such a SDF is:*

$$\boxed{\Vert\triangleright}(\mathbf{d}_n, \ldots, \boxed{\Vert\triangleright}(\mathbf{d}_1, \boxed{\Vert\triangleright}(\mathbf{d}_0, \mathbf{o}))\ldots).$$

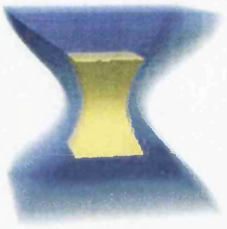
An equivalent software representation for hierarchical SDF is a right-heavy binary tree with $\boxed{\Vert\triangleright}$ as all the non-terminal nodes, $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_n$ as all the left terminal nodes, and $\mathbf{o}$ as the only right terminal node.



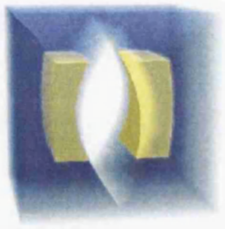
(a) **c**



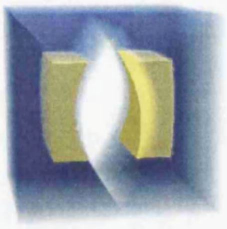
(b) $\boxed{\Vdash}$(**squeeze**, **c**)



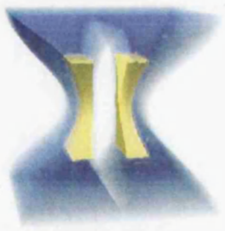
(c) $\boxed{\mapsto}$(**squeeze**, **c**)



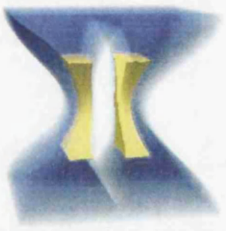
(d) $\boxed{\Vdash}$(**open**, **c**)



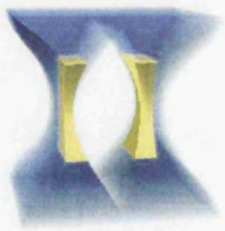
(e) $\boxed{\mapsto}$(**open**, **c**)



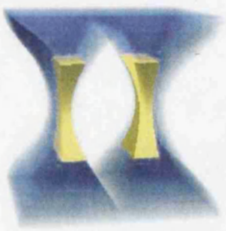
(f) $\boxed{\Vdash}$(**squeeze**, $\boxed{\Vdash}$(**open**, **c**))



(g) $\boxed{\mapsto}$(**squeeze**, $\boxed{\mapsto}$(**open**, **c**))



(h) $\boxed{\Vdash}$(**open**, $\boxed{\Vdash}$(**squeeze**, **c**))

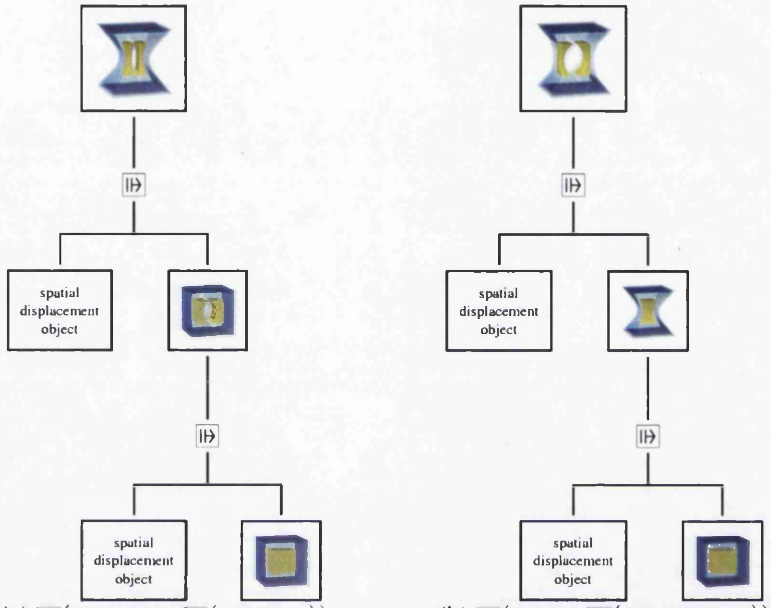


(i) $\boxed{\mapsto}$(**open**, $\boxed{\mapsto}$(**squeeze**, **c**))

Figure 6.3: Examples of applying two SDFs and STFs, **squeeze** and **open**, to a cubic operand object, **c**, in different hierarchical order. **c** is a procedurally-defined volume object, and has a translucent external shell and an opaque internal shell.

Definition 13 (Hierarchical STF). *A hierarchical STF [CSW+03] is a volume scene graph where a series of transfer objects,* $\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_n$*, are applied to an operand object* $\mathbf{o}$ *in a sequential order. An algebraic specification for such a STF is:*

$$\boxed{\mapsto}(\mathbf{s}_n, \ldots, \boxed{\mapsto}(\mathbf{s}_1, \boxed{\mapsto}(\mathbf{s}_0, \mathbf{o}))\ldots).$$



spatial displacement object
spatial displacement object
spatial displacement object
spatial displacement object

(a) $\boxed{\Vdash}(\mathbf{squeeze}, \boxed{\Vdash}(\mathbf{open}, \mathbf{c}))$ (b) $\boxed{\Vdash}(\mathbf{open}, \boxed{\Vdash}(\mathbf{squeeze}, \mathbf{c}))$

Figure 6.4: The volume scene graphs for Figures 6.3(d) and (e).

Figure 6.3 shows two example of hierarchical SDFs, $\boxed{\Vdash}(\mathbf{squeeze}, \boxed{\Vdash}(\mathbf{open}, \mathbf{c}))$ and $\boxed{\Vdash}(\mathbf{open}, \boxed{\Vdash}(\mathbf{squeeze}, \mathbf{c}))$. From Figure 6.3, it can be observed that **squeeze** and **open** are not commutative. In fact, in general that when $\mathbf{a} \neq \mathbf{b}$, $\boxed{\Vdash}(\mathbf{a}, \boxed{\Vdash}(\mathbf{b}, \mathbf{o})) \neq \boxed{\Vdash}(\mathbf{b}, \boxed{\Vdash}(\mathbf{a}, \mathbf{o}))$. Similar operations can also be simulated using hierarchical STFs, as demonstrated in Figure 6.3. Figure 6.4 shows volume scenes graph for specifying the two example of hierarchical SDFs, $\boxed{\Vdash}(\mathbf{squeeze}, \boxed{\Vdash}(\mathbf{open}, \mathbf{c}))$ and $\boxed{\Vdash}(\mathbf{open}, \boxed{\Vdash}(\mathbf{squeeze}, \mathbf{c}))$ in Figures 6.3(d) and (e).

As displacement objects are also spatial objects, one can also apply $\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_n$ in sequence among themselves, prior to the application to $\mathbf{o}$, for instance,

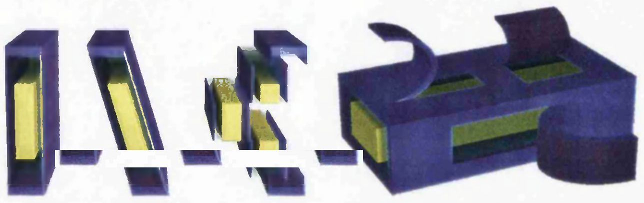
$$\boxed{\Vdash}(\boxed{\Vdash}(\boxed{\Vdash}(\ldots \boxed{\Vdash}(\mathbf{d}_n, \mathbf{d}_{n-1}), \ldots, \mathbf{d}_1), \mathbf{d}_0), \mathbf{o}).$$

It can also be shown that in general $\boxed{\Vdash}(\mathbf{a}, \boxed{\Vdash}(\mathbf{b}, \mathbf{c})) \neq \boxed{\Vdash}(\boxed{\Vdash}(\mathbf{a}, \mathbf{b}), \mathbf{c})$. In real life, deformation actions often take place in sequence, and very often, they are not 'commutative'. Hence, while satisfying the commutative law would be desirable, it is not essential.

Similar to displacement objects, transfer objects are also spatial objects and like other spatial objects, one can also apply $\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_n$ in sequence among themselves, prior to the application to $\mathbf{o}$, such that [CSW+03],

$$\boxed{\mapsto}(\boxed{\mapsto}(\boxed{\mapsto}(\ldots \boxed{\mapsto}(\mathbf{s}_n, \mathbf{s}_{n-1}), \ldots, \mathbf{s}_1), \mathbf{s}_0), \mathbf{o}).$$

Further examples of simple hierarchical SDFs can be seen in Figure 6.5. Figure 6.5(a) demonstrates how displacement functions can be used to imitate transfer functions in the manipulation of their individual attribute fields, ψ_x, ψ_y, ψ_z. Figure 6.5(b) also shows how geometrical transformations, such as rotation and scaling, can be applied to displacement objects as operand objects.



(a) Transfer function (b) Geometric transformation

Figure 6.5: Displacement objects can be manipulated in the same way as conventional objects. In (a), a rectangular volume object (left) is deformed by a simple displacement object, with $\psi_x(p) = (0.3)y$ and $\psi_y(p) = \psi_z(p) = 0$ (middle). Application of a transfer function to $\psi_x(p)$ in the form of a lookup table, resulting in a layered deformation (right). In (b), a displacement object is applied to different parts of an operand object with three different geometric transformations.

6.6 Identity Problem

The combinational approach of SDFs and STFs presents us with several benefits and drawbacks. One of the benefits of the combinational approach is its capability to combine both simple and complex non-conventional objects through the application of constructive operators. Some difficulties, however, can occur in the handling of these objects. An example of such difficulties is the case of *no deformation*. No deformation occurs when non-conventional objects are *temporal non-conventional objects*. Such a state occurs when the objects are in the state of not specifying any form of deformations. It therefore is necessary for the combinational approach to be able to handle such a state correctly.

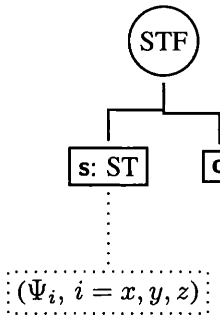
The commutativity, associativity and distributivity of ST and SD objects have been outlined in Section 6.4.1. This section investigates the satisfiability of the general identity definitions for both combinational ST and SD objects. In the course of proving this satisfiability, the identities of both ST and SD objects must be identified. A ST object can only have one form of an identity object, called a *transformation identity*. This is because a ST object can only define a mapping from one point to another. A SD object, however, is essentially a vector field which represents the relative distance of the new points from the original ones and therefore makes its *no transformation* (or identity transformation) to be operation dependent. Operation dependent identities of SD objects are divided into *translation identity* for $\boxed{+}$ and $\boxed{-}$, and *scaling identity* for $\boxed{*}$ and $\boxed{\div}$. These operations are as defined in Section 6.4.

A general definition for a combinational ST objects can be stated, such that:

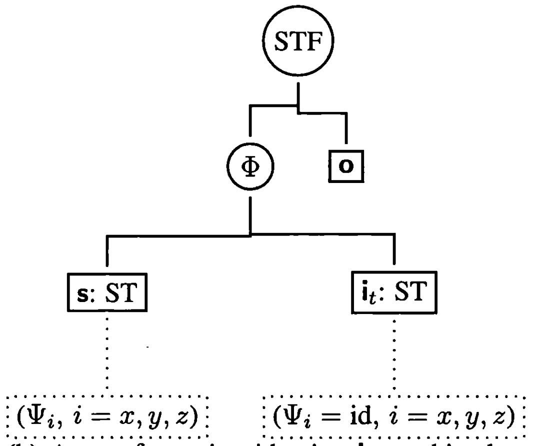
Definition 14 (General identity definition for ST object). *Let* $\mathbf{S}_T$ *be a class of* ST *objects. Then* $\mathbf{i}_t$ *is an identity of ST object, i.e.,* transformation identity, *if*

$$\forall \mathbf{s} \in \mathbf{S}_T : op(\mathbf{s}, \mathbf{i}_t) = \mathbf{s}$$

Figure 6.6 shows an illustration of the general identity definition for the combinational ST objects.



(a) A ST object **s**.



(b) A transformation identity, **i**, combined together with **s** using a constructive operator *op*.

Figure 6.6: General identity definition for the combinational ST objects.

Based on the definition of transformation identity (see Definition 7) and $\boxed{+}$, an additive identity of a ST object can be stated, such that:

Definition 15 (Additive identity of ST object). *Let* $\mathbf{S}_T$ *be a class of* ST *objects. Then* $\mathbf{i}_t$ *is an identity of ST object, i.e.,* transformation identity, *if*

$$\forall \mathbf{s} \in \mathbf{S}_T : \boxed{+}(\mathbf{s}, \mathbf{i}_t) = \mathbf{s}$$

The identity for $\boxed{+}$ can be specified, such that,

$$\mathbf{i}_t(p) = (0, 0, 0).$$

Other identities can also be shown for $\boxed{-}$, $\boxed{*}$ and $\boxed{\div}$ where:

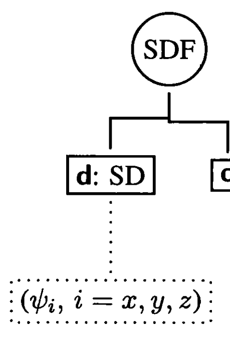
$$\begin{aligned} \boxed{-} &: \mathbf{i}_t(p) = (0, 0, 0) \\ \boxed{*} &: \mathbf{i}_t(p) = (1, 1, 1) \\ \boxed{\div} &: \mathbf{i}_t(p) = (1, 1, 1). \end{aligned}$$

A general identity definition for a combinational SD objects can also be stated, such that:

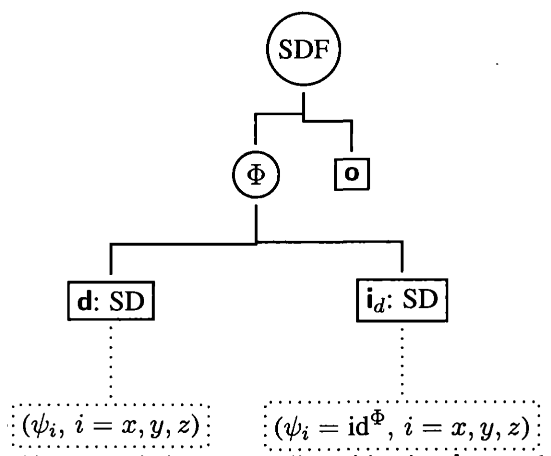
Definition 16 (General identity definition for SD object). *Let* $\mathbf{S}_D$ *be a class of* SD *objects. Then* $\mathbf{i}_d$ *is an identity of SD object, i.e., a* translation *or* scaling identity, *if*

$$\forall \mathbf{d} \in \mathbf{S}_D : op(\mathbf{d}, \mathbf{i}_d) = \mathbf{d}.$$

Figure 6.7 shows a volume scene graph of the general identity definition for the combinational SD objects.



(a) A SD object **d**.



(b) A translation (or scaling) identity, $\mathbf{i}_d$, combined together with **d** using a constructive operator *op*.

Figure 6.7: General identity definition for the combinational SD objects.

From the definition of the translation identity (see Definition 8) and $\boxed{+}$, the following definition can be stated:

Definition 17 (Additive identity of SD object). *Let* $\mathbf{S}_D$ *be a class of* SD *objects. Then* $\mathbf{i}_d$ *is an identity of SD object, i.e.,* translation identity, *if*

$$\forall \mathbf{d} \in \mathbf{S}_D : \boxed{+}(\mathbf{d}, \mathbf{i}_d) = \mathbf{d}.$$

This can be shown from the definition of $\boxed{+}$ where $\forall p, p' \in \mathbb{E}^3$, there is:

$$\boxed{+}(\mathbf{d}, \mathbf{i})(p) = \psi^d(p) + \psi^i(p) = (\psi^d_x, \psi^d_y, \psi^d_z) + (0, 0, 0) = (\psi^d_x, \psi^d_y, \psi^d_z).$$

The same definition can also be defined for $\boxed{-}$, such that:

Definition 18 (Subtractive identity of SD object). *Let* $\mathbf{S}_D$ *be a class of* SD *objects. Then* $\mathbf{I}_d$ *is an identity of SD object, i.e.,* transformation identity, *if*

$$\forall \mathbf{d} \in \mathbf{S}_D : \boxed{-}(\mathbf{d}, \mathbf{i}_d) = \mathbf{d}.$$

Using the definition of scaling identity (see Definition 9) and $\boxed{*}$, the following definition can also be stated:

Definition 19 (Multiplicative identity of SD object). *Let* $\mathbf{S}_D$ *be a class of* SD *objects. Then* $\mathbf{i}_d$ *is an identity of SD object, i.e., the* scaling identity, *if*

$$\forall \mathbf{d} \in \mathbf{S}_D : \boxed{*}(\mathbf{d}, \mathbf{i}_d) = \mathbf{d}.$$

This can also be shown using the definition of $\boxed{*}$ where $\forall p, p' \in \mathbb{E}^3$, there is:

$$\boxed{*}(\mathbf{d}, \mathbf{i})(p) = \psi^d(p) * \psi^i(p) = (\psi^d_x, \psi^d_y, \psi^d_z) * (1, 1, 1) = (\psi^d_x, \psi^d_y, \psi^d_z).$$

Similarly, for $\boxed{\div}$ it also can be stated, such that:

Definition 20 (Division identity of SD object). *Let* $\mathbf{S}_D$ *be a class of* SD *objects. Then* $\mathbf{i}_d$ *is an identity of SD object, i.e., the* scaling identity, *if*

$$\forall \mathbf{d} \in \mathbf{S}_D : \boxed{\div}(\mathbf{d}, \mathbf{I}_d) = \mathbf{d}.$$

For the rest of the constructive operations defined in Tables 6.1 and 6.2, SD and ST objects do not satisfy the identity definition.

6.7 Inverse Problem

One of the complexities that can occur in the combinational approach of SDFs and STFs is the desire to maintain the reversibility of a deformation defined in a collection of non-conventional objects. Such a situation can occur during interactive (or incremental) deformations whereby a series of non-conventional objects have been combined together using constructive operators and one of the deformation stages has been transformed through the reversion of one or more non-conventional objects. More directly, inverses are required in the combinational approach to handle the reversibility of a deformation and are examined in further detail in this section.

A general inverse definition for a combinational ST objects can be stated, such that:

Definition 21 (General inverse definition for ST object). *Let* $\mathbf{S}_T$ *be a class of ST objects,* $\mathbf{s}_1 \in \mathbf{S}_T$, *and* $\mathbf{s}_2 \in \mathbf{S}_T$ *be two ST objects.* $\mathbf{s}_1$ *and* $\mathbf{s}_2$ *are called inverse of each other with respect to* Φ

$$\Phi(\mathbf{s}_1, \mathbf{s}_2) = \mathbf{s}^\tau$$

i.e.,

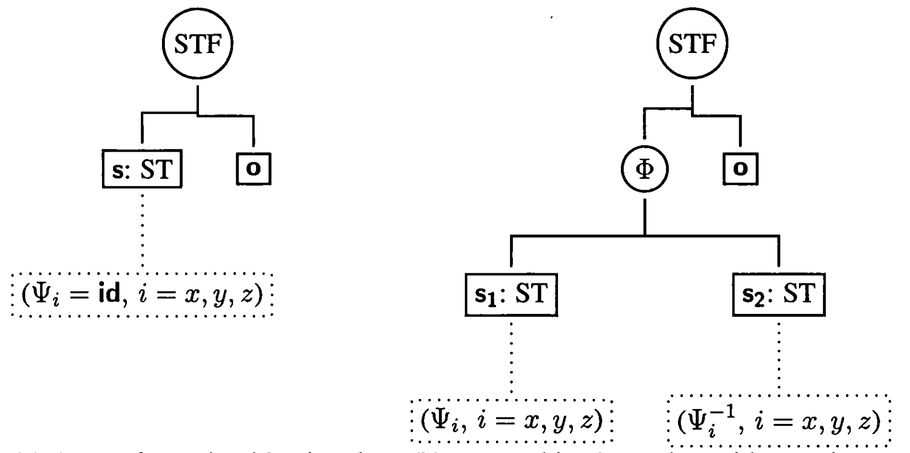
$$\boxed{\mapsto}(\Phi(\mathbf{s}_1, \mathbf{s}_2), \mathbf{o}) = \mathbf{o} = \boxed{\mapsto}(\Phi(\mathbf{s}_2, \mathbf{s}_1), \mathbf{o}).$$

Then $\boxed{\mapsto}(\mathbf{s}_1, \boxed{\mapsto}(\mathbf{s}, \mathbf{o})) = \mathbf{o}$ if $\Phi(\mathbf{s}_1, \mathbf{s}_2) = \mathbf{s}_1 \circ \mathbf{s}_2$. For Φ being component-wise composition of functions, *i.e.,*

$$(\Psi_{1,x}, \Psi_{1,y}, \Psi_{1,z}) \circ (\Psi_{2,x}, \Psi_{2,y}, \Psi_{2,z}) = (\Psi_{1,x} \circ \Psi_{2,x}, \Psi_{1,y} \circ \Psi_{2,y}, \Psi_{1,z} \circ \Psi_{2,z})$$

$\mathbf{s}_1$ can be defined to be a ST object such that $\forall p \in \mathbb{E}^3$, $p' = \mathbf{s}_1(p) = \Psi(p) \neq p$ for some function, $\Psi : \mathbb{E}^3 \to \mathbb{R}^3$. $\mathbf{s}_1$ has an inverse ST object $\mathbf{s}_2$ iff Ψ has an inverse function Ψ^{-1}, *i.e.,* $\Psi^{-1}(\Psi(p)) = p$. Such Ψ^{-1} would only exist iff Ψ is a bijective function, that is:

- Ψ is an injective function: $\Psi(p) = \Psi(p')$ iff $p = p'$ and
- Ψ is a surjective function, where every point from the range has its mapped point in the domain.



(a) A transformation identity object $\mathbf{d}^{\tau}$.

(b) $\mathbf{s_1}$, combined together with $\mathbf{s_2}$ using a constructive operator Φ.

Figure 6.8: General inverse definition for the combinational ST objects.

Figure 6.8 presents a representation of the general inverse definition for the ST combinational objects.

The determination of the inverse of a SD object is, however, a less complicated process as a SD object, **d**, is essentially a vector field. Therefore, the inverse of **d** can be determined through the inverse of its vector field. A general inverse definition for the combinational SD objects can be stated, such that:

Definition 22 (General inverse definition for SD object). *Let* $\mathbf{S}_D$ *be a class of SD objects,* $\mathbf{d_1} \in \mathbf{S}_D$*, and* $\mathbf{d_2} \in \mathbf{S}_D$ *be two SD objects. Then* $\mathbf{d_1}$ *and* $\mathbf{d_2}$ *are called inverse of each other if*

$$\Phi(\mathbf{d_1}, \mathbf{d_2}) = \mathbf{id}^{\Phi}.$$

An illustration of the general inverse definition of the combinational SD objects can be seen in Figure 6.9.

From the definition of $\boxed{+}$, the additive inverse of the SD objects can be stated, such that:

Definition 23 (Additive inverse of SD objects). *Let* $\mathbf{S}_D$ *be a class of SD objects,* $\mathbf{d_1} \in \mathbf{S}_D$*, and* $\mathbf{d_2} \in \mathbf{S}_D$ *be two SD objects.* $\mathbf{d_1}$ *and* $\mathbf{d_2}$ *are called inverse of each other if,*

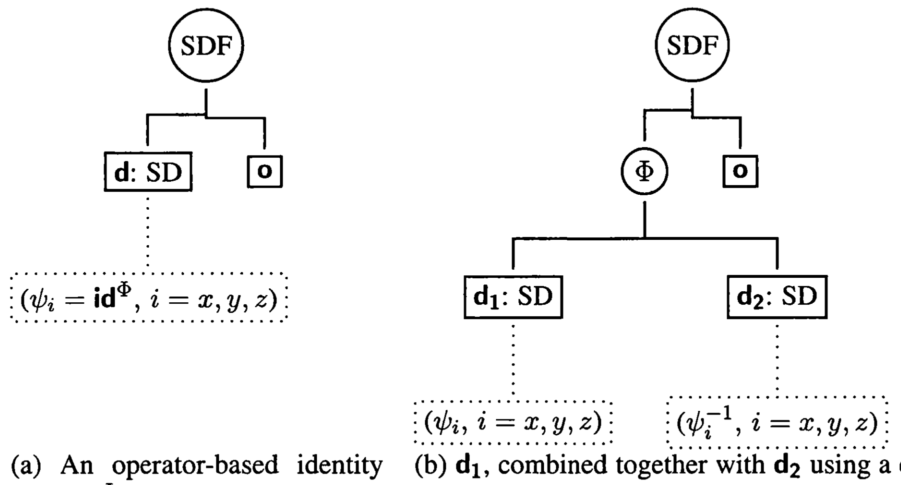
$$\forall \mathbf{d_1} \in \mathbf{S}_D, \exists \mathbf{d_2} \in \mathbf{S}_D : \boxed{+}(\mathbf{d_1}, \mathbf{d_2}) = \mathbf{d}^{\iota} \Rightarrow \mathbf{d_2} = -\mathbf{d_1}$$

or

$$\boxed{\Vdash}(\boxed{+}(\mathbf{d_1}, \mathbf{d_2}), \mathbf{o}) = \mathbf{o} = \boxed{\Vdash}(\boxed{+}(\mathbf{d_2}, \mathbf{d_1}), \mathbf{o}) \Rightarrow \mathbf{d_2} = -\mathbf{d_1}.$$

This definition can be shown with the following arguments:

Let $\mathbf{d_1}$ and $\mathbf{d_2}$ be two SD objects, such that $\mathbf{d_1} = (\psi_x, \psi_y, \psi_z)$ and $\mathbf{d_2} = (\psi_x, \psi_y, \psi_z)$.



(a) An operator-based identity object $\mathbf{d}^{\Phi}$.

(b) $\mathbf{d}_1$, combined together with $\mathbf{d}_2$ using a constructive operator Φ.

Figure 6.9: General inverse definition for the combinational SD objects.

From the definition $\boxplus$, $\forall p \in \mathbb{E}^3$ there is:

$$p' = p + \boxplus(\mathbf{d}_1, \mathbf{d}_2)(p) \overset{?}{=} p$$

Such a statement is true if:

$$\begin{array}{rrcl} & \boxplus(\mathbf{d}_1, \mathbf{d}_2)(p) & = & \mathbf{d}^{\iota}(p) \\ \Rightarrow & (\psi_x, \psi_y, \psi_z) + (\psi_x, \psi_y, \psi_z) & = & (0,0,0) \\ \Rightarrow & (\psi_x, \psi_y, \psi_z) & = & (0,0,0) - (\psi_x, \psi_y, \psi_z) \\ \Rightarrow & (\psi_x, \psi_y, \psi_z) & = & (-\psi_x, -\psi_y, -\psi_z) \end{array}$$

Clearly, for the operator $\boxminus$ its inverse exist and its definition can be stated, such that,

Definition 24 (Subtractive inverse of SD objects). *Let* $\mathbf{S}_D$ *be a class of SD objects,* $\mathbf{d}_1 \in \mathbf{S}_D$*, and* $\mathbf{d}_2 \in \mathbf{S}_D$ *be two SD objects. Then* $\mathbf{d}_1$ *and* $\mathbf{d}_2$ *are called inverse of each other if,*

$$\forall \mathbf{d}_1 \in \mathbf{S}_D, \exists \mathbf{d}_2 \in \mathbf{S}_D : \boxminus(\mathbf{d}_1, \mathbf{d}_2) = \mathbf{d}^{\iota} \Rightarrow \mathbf{d}_2 = \mathbf{d}_1$$

or

$$\forall \mathbf{d}_1 \in \mathbf{S}_D, \exists \mathbf{d}_2 \in \mathbf{S}_D : \boxed{\Vdash}(\boxminus(\mathbf{d}_1, \mathbf{d}_2), \mathbf{o}) = \mathbf{o} = \boxed{\Vdash}(\boxminus(\mathbf{d}_2, \mathbf{d}_1), \mathbf{o}) \Rightarrow \mathbf{d}_2 = \mathbf{d}_1.$$

From the definition of $\boxed{*}$, the following definition can also be stated where:

Definition 25 (Multiplicative inverse of SD objects). *Let* $\mathbf{S}_D$ *be a class of SD objects,* $\mathbf{d}_1 \in \mathbf{S}_D$*, and* $\mathbf{d}_2 \in \mathbf{S}_D$ *be two SD objects. Then* $\mathbf{d}_1$ *and* $\mathbf{d}_2$ *are called inverse of each other if*

$$\forall \mathbf{d}_1 \in \mathbf{S}_D, \exists \mathbf{d}_2 \in \mathbf{S}_D : \boxed{*}(\mathbf{d}_1, \mathbf{d}_2) = \mathbf{d}^{\zeta} \Rightarrow \mathbf{d}_2 = \boxed{\div}(\mathbf{d}^{\zeta}, \mathbf{d}_1)$$

or

$$\forall \mathbf{d}_1 \in \mathbf{S}_D, \exists \mathbf{d}_2 \in \mathbf{S}_D : \boxed{\Vdash}(\boxed{*}(\mathbf{d}_1, \mathbf{d}_2), \mathbf{o}) = \mathbf{o} = \boxed{\Vdash}(\boxed{*}(\mathbf{d}_2, \mathbf{d}_1), \mathbf{o}) \Rightarrow \mathbf{d}_2 = \boxed{\div}(\mathbf{d}^{\zeta}, \mathbf{d}_1).$$

This definition can be shown in the following arguments:

Let $\mathbf{d_1}$ and $\mathbf{d_2}$ be two SD objects, such that $\mathbf{d_1} = (\psi_x, \psi_y, \psi_z)$ and $\mathbf{d_2} = (\psi_x, \psi_y, \psi_z)$.

From the definition of $\boxed{*}$, it is found that $\forall p \in \mathbb{E}^3$:

$$p' = p + \boxed{*}(\mathbf{d_1}, \mathbf{d_2})(p) \overset{?}{=} p$$

This is true if:

$$\begin{array}{lrcl} & \boxed{*}(\mathbf{d_1}, \mathbf{d_2})(p) & = & \mathbf{d}^{\varsigma}(p) \\ \Rightarrow & (\psi_x, \psi_y, \psi_z) * (\psi_x, \psi_y, \psi_z) & = & (1,1,1) \\ \Rightarrow & (\psi_x, \psi_y, \psi_z) & = & \frac{(1,1,1)}{(\psi_x, \psi_y, \psi_z)} \\ \Rightarrow & (\psi_x, \psi_y, \psi_z) & = & \boxed{\div}(\mathbf{d}^{\varsigma}, \mathbf{d_1}) \end{array}$$

Using the same approach, it can also be shown for $\boxed{\div}$, that there is:

Definition 26 (Division inverse of SD objects). *Let* $\mathbf{S}_D$ *be a class of SD objects,* $\mathbf{d_1} \in \mathbf{S}_D$, *and* $\mathbf{d_2} \in \mathbf{S}_D$ *be two SD objects. Then* $\mathbf{d_1}$ *and* $\mathbf{d_2}$ *are called inverse of each other if*

$$\forall \mathbf{d_1} \in \mathbf{S}_D, \exists \mathbf{d_2} \in \mathbf{S}_D : \boxed{\div}(\mathbf{d_1}, \mathbf{d_2}) = \mathbf{d}^{\varsigma} \Rightarrow \mathbf{d_2} = \mathbf{d_1}$$

or

$$\forall \mathbf{d_1} \in \mathbf{S}_D, \exists \mathbf{d_2} \in \mathbf{S}_D : \boxed{\Vdash}(\boxed{\div}(\mathbf{d_1}, \mathbf{d_2}), \mathbf{o}) = \mathbf{o} = \boxed{\Vdash}(\boxed{\div}(\mathbf{d_2}, \mathbf{d_1}), \mathbf{o}) \Rightarrow \mathbf{d_2} = \mathbf{d_1}.$$

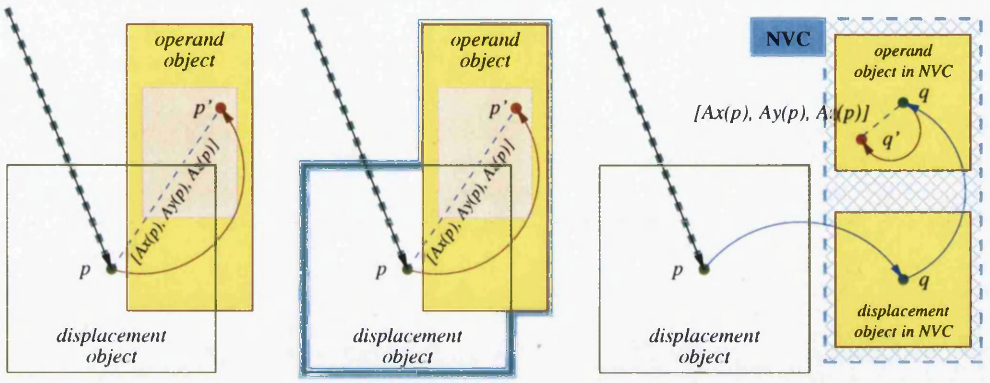
6.8 Direct Rendering Volume Scene Graphs

The generic framework defined in Section 6.3 enables a discrete ray tracer for a volume scene graph to handle both non-conventional and conventional objects in a very similar manner. Geometric transformations, such as scaling, rotation and translation, can be performed on displacement and transfer objects in the same approach as in conventional objects. The generic framework also enables both displacement and transfer objects to be constructed based on their physical properties or luminance attributes, such as opacity, colour components and refractive indices. The ability to support such a construction allows both non-conventional objects to accomplish a multiple of complex deformations without the need to modify their deformation functions. This section examines the algorithms that are formulated for rendering three types of generic SDFs and STFs.

6.8.1 With Spatial Displacement Functions

In this section, algorithms for rendering three generic types of SDFs are presented. The three generic types of SDFs that are to be presented are *isolated SDF*, *local SDF* and *normalised SDF*.

A discrete ray tracer ray traces a displacement object by sampling along a ray at a specified interval. At each sampling point, it traverses the volume scene graph recursively from its root. Figure 6.10(a) shows that when a sampling point p is within the bounding box of a SDF node, the first sample that is to be examined is its displacement object **d**, which is located on its left branch. The object that is to be examined can either be an elemental object (on a terminal node) or a composite object (in the form of a subtree). The result of a successful sampling is a displacement vector, $\mathbf{v}_d = [A_x(p), A_y(p), A_z(p)]$, whereby p is transferred to p', $p' = p + \mathbf{v}_d$. The operand object **o** on the right branch of the SDF node is then continued to be sampled by the ray tracer. The operand object can either be an elemental (*i.e.*, conventional volume object) or a composite object (*i.e.*, a displacement object). In the context of deformation, this enables a backward mapping from p to p', as the luminance attributes **o** at p' seem to have been repositioned to p (*i.e.*, forward mapping).



(a) Type-I SDF (b) Type-L SDF (c) Type-N SDF

Figure 6.10: Three generic types of SDFs.

As **d** and **o** are both considered to be bounded spatial objects *i.e.*, volume objects, it is typically the case that the spatial domain of **d** does not have a one-to-one mapping with that of **o**. The ray tracer usually returns a status of sampling failure to the parent node in the case of $p \notin \mathbf{d}$ or $p \in \mathbf{d}$ but $p' \notin \mathbf{o}$. When the ray tracer reaches the root level, this sample is skipped without invoking shading and composition routines unless the failure status has been overwritten by one of the ancestral nodes (*e.g.*, by a union operator).

It is worth noting that having a special value $\varnothing$ does not directly indicate a sampling failure. The special value $\varnothing$ just enforces the ray tracer to deal with it differently depending on whether it is sampling a subtree for displacement or for luminance attributes.

Figure 6.10(a) demonstrates that **d** may have a map to cover a specific part of **o**. It is therefore not necessary that every point in **o** has a corresponding point in **d**. Any of the points that have not been sampled will not be visited by the ray tracer, unless there are other displacement objects that have maps covering these points. This type of SDF can be defined as a rather 'self-important' SDF as it only takes into consideration part of the scene that it can reach. This type of SDF is known as *isolated SDF* (or *type-I SDF*) and can be denote as $\boxed{\Vdash}^{\mathrm{I}}$. Figures 6.1(b), (d), (f), (h), 6.3(b) and 6.5(a) demonstrate the implementations of $\boxed{\Vdash}^{\mathrm{I}}$.

One of the benefits of $\boxed{\Vdash}^{\mathrm{I}}$ is that various forms of deformation can be implemented through the application of geometric transformation on displacement objects without the need to alter its mapping function. A typical example of such implementations includes volume clipping, a magnifying glass effect and a region based deformation. The area of interest can be shifted within the operand object either through the modification of the mapping function or the combination of the displacement object with other displacement objects using one of the combination operators defined in Section 6.4.

An alternative version of $\boxed{\Vdash}^{\mathrm{I}}$ would be to define an operand object $\mathbf{o}$ of $\boxed{\Vdash}^{\mathrm{I}}$ as a composite displacement object. In this approach, $\mathbf{d}^{\iota}$ is considered to be a ghost object where $\forall p \in \mathbf{d}$ the displacement object is sampled first, which may result in a displacement vector $\mathbf{v}_d$. A new sampling point, p', can be calculated through the addition of $\mathbf{v}_d$ to p, $p' = p + \mathbf{v}_d$. The operand object of $\boxed{\Vdash}^{\mathrm{I}}$ is then sampled by the ray tracer. In this situation, the operand object is a composite object specified through $\boxed{\Vdash}^{\mathrm{N}}(\mathbf{d}^{\iota}, \mathbf{o})$. If $p' \in \mathbf{d}^{\iota}$ then the operand object of $\boxed{\Vdash}^{\mathrm{N}}$ is evaluated at p', otherwise the ray tracer skips the current sampling point and returns a status of sampling failure to the parent node.

If $p \notin \mathbf{d}$, $\mathbf{v}_d$ is specified to $\varnothing$. p would again be skipped and the ray tracer will return a sampling failure to the root node unless it has been overwritten by another ancestral node. If $\mathbf{d}$ is, however, a combinational object, $\mathbf{d} = \Phi(\mathbf{d}_0, \ldots, \mathbf{d}_n)$, then $\forall \mathbf{d}_i$ where $p \in \mathbf{d}_i$ have been sampled a series of displacement vectors $\mathbf{v}_{d_i}$ would have been produced. $\mathbf{v}_{d_i}$ can be joined together through a user defined combinational operator, Φ. A new sampling point, p', can be calculated through the addition of $\mathbf{v}_{d_i}$ to p, $p' = p + \Phi(\mathbf{v}_{d_0}, \ldots, \mathbf{v}_{d_n})$. The operand object is sampled for every valid sampling point, p'. The ray tracer returns a status of sampling failure to the root node $\forall p' \notin \mathbf{o}$.

This implementation is similar to the previous implementation in that geometric transformations such as rotation and scaling can be performed on various area of an operand object (*i.e.,* a ghost object). Such an implementation however limits the world position of a displacement object, as it has to remain within the bounding box of the ghost object during a geometric transformation. One way of overcoming this limitation is to extend the area of interest through a scaling transformation of the displacement object instead of enlarging it.

One of the drawbacks of using $\boxed{\Vdash}^{\mathrm{I}}$ is that it can result in the partial appearance of an operand object in a number of cases. The truncation of an operand object can be eliminated through the implementation of a different generic type of SDF, called *local SDF* (or *type-L SDF*). $\boxed{\Vdash}^{\mathrm{L}}$ can be used to denote a *type-L SDF.*

In a sub-tree, $\boxed{\Vdash}^{\mathrm{L}}(\mathbf{d}, \mathbf{o})$, its valid spatial domain is equivalent to that of $\boxed{\cup}(\mathbf{d}, \mathbf{o})$. Once more, a displacement vector, $\mathbf{v}_d$ is calculated by sampling the left child, $\mathbf{d}$, of $\boxed{\Vdash}^{\mathrm{L}}$. A new sampling point, p', can then be obtained through the addition of the displacement vector and the current sampling point, p. The right child is only sampled if p' is a valid sampling point. The ray tracer would return a status of sampling failure if $p' \notin \mathbf{o}$ and evaluate the right child of the tree at the new sampling point p'. In $\boxed{\Vdash}^{\mathrm{L}}$, however, the ray tracer does not return a status of sampling failure to the root if $p \notin \mathbf{d}$ and still sample the operand object. In this scenario, the new sampling point, p', and the old sampling point, p are defined to be the same value (*i.e.,* no deformation).

$\boxed{\Vdash}^{\mathrm{I}}$ is similar to $\boxed{\Vdash}^{\mathrm{L}}$ in that both types of SDF are able to have a combination object as its displacement object, $\mathbf{d} = \Phi(\mathbf{d}_0, \ldots, \mathbf{d}_n)$. In $\boxed{\Vdash}^{\mathrm{L}}$, however, each of the displacement objects, $\mathbf{d}_i$, have to be evaluated in the order in which they have been defined, resulting in a displacement vector, $\mathbf{v}_{d_i}$. The evaluation of the right child of $\boxed{\Vdash}^{\mathrm{L}}$ is then done at the newly defined sampling point, $p' = p + \Phi(\mathbf{v}_{d_1}, \ldots, \mathbf{v}_{d_n})$. The operand object is sampled for every valid $p' \in \mathbf{o}$.

The above description is illustrated in Figure 6.10(b), where based on $\mathbf{d}$, a local deformation in the mapped region (shown in pink) of $\mathbf{o}$ has been performed by $\boxed{\Vdash}^{\mathrm{L}}$, while leaving the unmapped region unaltered. Figures 6.2, 6.3(c) and 6.5(b) demonstrate the implementations of $\boxed{\Vdash}^{\mathrm{L}}$.

Similarly to $\boxed{\Vdash}^{\mathrm{I}}$, $\boxed{\Vdash}^{\mathrm{L}}$ can also be implemented where a ghost object, $\mathbf{d}^{\iota}$, can be used in the definition of the bounding box of an operand object. The displacement object of $\boxed{\Vdash}^{\mathrm{L}}$ can be calculated in the same manner shown in the previous implementation, which returns a new sampling point, p'. For every valid point p', the right child of $\boxed{\Vdash}^{\mathrm{L}}$ can also be calculated in the same approach as that shown in the alternative implementation of $\boxed{\Vdash}^{\mathrm{I}}$. If the displacement object of $\boxed{\Vdash}^{\mathrm{L}}$ is, however, a combinational object, then $\forall p \notin \mathbf{d}_{\mathrm{i}}$, and the ray tracer will sample the operand object of $\boxed{\Vdash}^{\mathrm{L}}$ rather than returning a status of sampling failure to the parent node. Further implementations of $\boxed{\Vdash}^{\mathrm{L}}$ can be observed in Figure 6.11 where a combination of **bend** and **twist** operations has been applied to a Bonsai tree.

In both of the $\boxed{\Vdash}^{\mathrm{I}}$ and $\boxed{\Vdash}^{\mathrm{L}}$ implementations, p and p' are both defined in the world coordinates. Although their usage in local deformation is generally very effective, their usage in certain applications, such as volume animation, can be rather awkward. An example of such situation is when a sequence of captured motion is attached to a volume object $\mathbf{o}$ through displacement objects $\mathbf{d}_0, \ldots, \mathbf{d}_n$. During that process, it is often desirable to have a continuous mapping from each $\mathbf{d}_i$ to $\mathbf{o}$, regardless of where and how $\mathbf{d}_i$ is positioned in a scene. This need prompted the introduction of the third generic type of SDF, called *normalised SDF* (or *type-N SDF*) and can be denoted as $\boxed{\Vdash}^{\mathrm{N}}$.

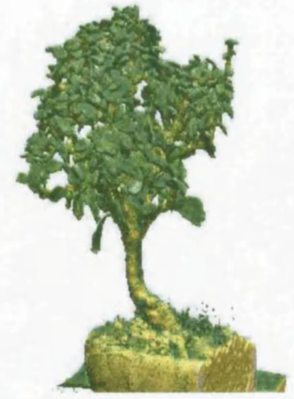
Figure 6.10(c) presents a pictorial view of *type-N SDF*. In $\boxed{\Vdash}^{\mathrm{N}}(\mathbf{d}, \mathbf{o})$ deformation is often performed in a normalised volume domain of $[0,1]^3$ with the assumption that $\mathbf{d}$ and $\mathbf{o}$ are tightly coupled together in the normalised volume domain. Given that $p \in \mathbf{d}$, p is first mapped from world coordinates to normalised volume coordinates (NVC), $q \in [0,1]^3$. q can then be transferred to $q' = q + \mathbf{v}_d$, where $\mathbf{v}_d = \mathbf{d}(p) = [A_x(p), A_y(p), A_z(p)]$. $\mathbf{o}$ at q' in the normalised volume domain is then sampled by the ray tracer. For a displacement specification, any $[A_x(p), A_y(p), A_z(p)] \notin [-1,1]$ would result in a sampling failure.

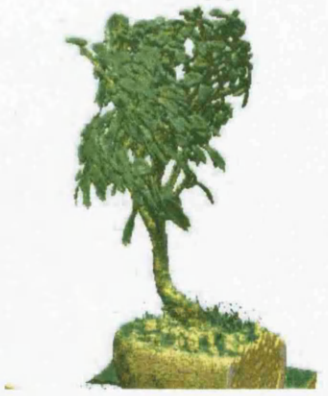
One important aspect of $\boxed{\Vdash}^{\mathrm{N}}$ is that it can be implemented to achieve exactly the same functionality as that of an existing deformation scheme proposed by Chen *et al.* [CSW+03], which has been successfully utilised for modelling and rendering volume animation.

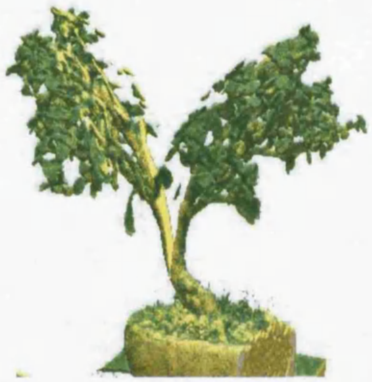
6.8.2 With Spatial Transfer Functions

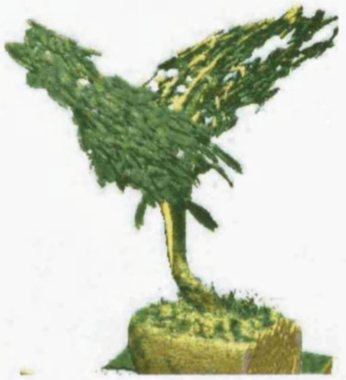
Similar to the previous section, this section also examines algorithms for rendering three generic types of STFs. The three generic types of STFs that are to be examined are *isolated STF*, *local STF* and *normalised STF*. Each of the generic types of STFs is denoted by $\boxed{\vdash}^{\mathrm{I}}$, $\boxed{\vdash}^{\mathrm{L}}$ and $\boxed{\vdash}^{\mathrm{N}}$, respectively.

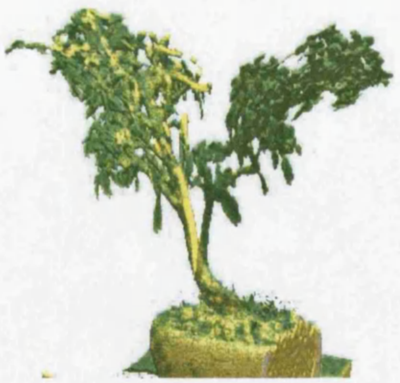
A transfer object is treated by a discrete ray tracer in the same approach as that of a displacement object. A transfer object is ray traced by taking sampling points along the ray at


(a) A Bonsai tree, **b**, with no deformation, $\mathbf{b} = \boxed{\Vdash}^{\mathrm{N}}(\mathbf{d}^{\iota}, \textbf{bonsai})$.


(b) A **twist** operation applied to the Bonsai tree, $\boxed{\Vdash}^{\mathrm{L}}(\textbf{twist}, \mathbf{b})$.


(c) **bend** operations applied to both halves of the Bonsai tree together with a rotation applied to **bend** operation, $\boxed{\Vdash}^{\mathrm{L}}(\boxed{\cup}(\textbf{bend}_l, \textbf{bend}_r), \mathbf{b})$.


(d) A **twist** operation applied to the Bonsai tree after the **bend** operations, $\boxed{\Vdash}^{\mathrm{L}}(\textbf{twist}, \boxed{\Vdash}^{\mathrm{L}}(\boxed{\cup}(\textbf{bend}_l, \textbf{bend}_r), \mathbf{b}))$.


(e) A **twist** operation applied to the Bonsai tree before the **bend** operations, $\boxed{\Vdash}^{\mathrm{L}}(\boxed{\cup}(\textbf{bend}_l, \textbf{bend}_r), \boxed{\Vdash}^{\mathrm{L}}(\textbf{twist}, \mathbf{b}))$.

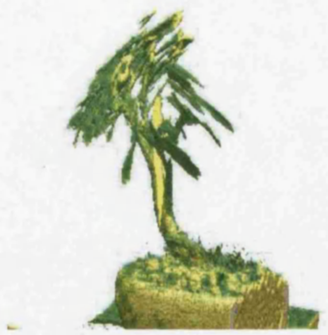
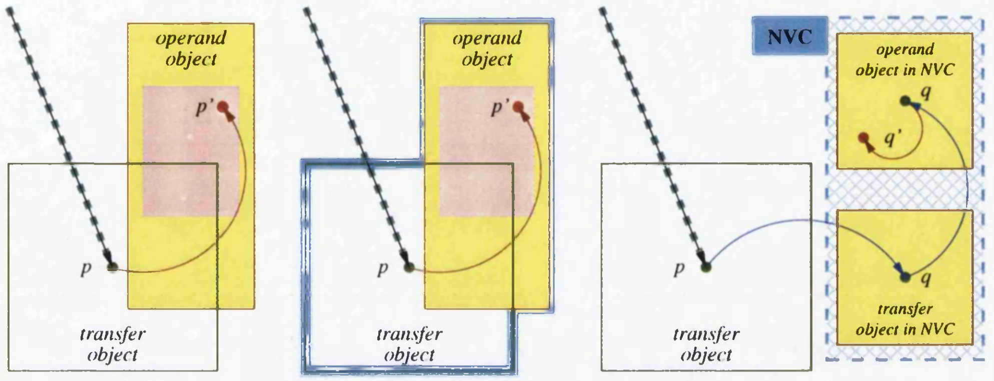

(f) **bend** and **twist** operations combined together before being applied to the Bonsai tree, $\boxed{\Vdash}^{\mathrm{L}}(\boxed{\cup}(\textbf{bend}_l, \textbf{bend}_r, \textbf{twist}), \mathbf{b}))$.

Figure 6.11: A combination of **bend** and **twist** operations applied to a Bonsai tree using *type-L SDF*.

a user-defined interval. At each sampling point, the ray tracer traverses the volume scene graph recursively from its root. In Figure 6.12(a), it can be seen that when a sampling point p is within the bounding box of a STF node, its transfer object, **s**, on the left branch is first sampled. The transfer object can be either an elemental object (on a terminal node) or a combinational object (in the form of a subtree). The result of a successful sampling will generate a new sampling point, $p' \in \mathbb{E}^3$. The operand object, **o**, on the right branch of the STF node is then continued to be sampled by the ray tracer. The operand object can be either an elemental (*i.e.,* a conventional volume object) or a composite transfer object.



(a) Type-I STF (b) Type-L STF (c) Type-N STF

Figure 6.12: Three generic types of STFs.

The nature of **s** and **o** is similar to that of **d** in that they are all bounded spatial objects, *i.e.,* volume objects. The spatial domain of **s** does not usually have a one-to-one mapping with **o**. In such a situation where $p \notin \mathbf{s}$ or $p \in \mathbf{s} : p' \notin \mathbf{o}$, the ray tracer generally returns a status of sampling failure to the parent node. When the ray tracer reaches the root level, it skips this sampling point without initiating shading and composition routines unless the failure status has been overwritten by one of the ancestral nodes (*e.g.,* by a union operator).

The *empty value*, $\varnothing$, is also introduced in STF implementation. It is to be reiterated that a special value $\varnothing$ does not directly indicate a sampling failure, but only enforces the ray tracer to manage it differently, by treating it either as a new sampling point of a subtree or a luminance attribute.

Figure 6.12(a) illustrates the concept of $\boxed{\mapsto}^{\mathrm{I}}$. One of the aims of $\boxed{\mapsto}^{\mathrm{I}}$ is to enable an approach where only a region of an operand object is required. An advantage of this approach is that the region of interest can be focused on while its context is ignored. In $\boxed{\mapsto}^{\mathrm{I}}$ implementation, the transfer object **s** may have a map that covers only specific part(s) of the operand object **o**. With this form of mapping, it is not necessary that every point $p \in \mathbf{o}$ has a corresponding point in **s**. Any of the points that have not been sampled will not be visited by the ray tracer, unless there are other transfer objects that have maps covering these points.

In $\boxed{\mapsto}^{\mathrm{I}}$ approach, the left child of the node $\boxed{\mapsto}^{\mathrm{I}}$ is first ray traced by the ray tracer. A new sampling point, p', is returned for every sampling point $p \in \mathbf{s}$ where the deformation or mapping function, Ψ, of **s** has been evaluated. $\forall p'$ the operand object, **o**, of $\boxed{\mapsto}^{\mathrm{I}}$ is sampled. The ray tracer returns a status of sampling failure if $p \notin \mathbf{s}$ or $p' \notin \mathbf{o}$. One of the advantages of $\boxed{\mapsto}^{\mathrm{I}}$ is that geometric transformations, such as rotation and scaling, can be performed while the region of interest is kept unaltered.

As seen early in the chapter, combinational transfer objects are not able to satisfy all of the algebraic definitions. The unsatisfiability of some of the definitions presents a number of difficulties for $\boxed{\mapsto}^{\mathrm{I}}$. One of the difficulties is that a region of interest that is covered by a transfer object $\mathbf{s}$ may not be easily changed by combining it with other transfer objects using a combinational operator, Φ. This difficulty is synonymous with limitation found in transfer objects. Hence in order to alter the region of interest, a modification to the mapping function Ψ of $\mathbf{s}$ has to be made.

An alternative version of $\boxed{\mapsto}^{\mathrm{I}}$ can also be implemented. In this implementation, a new sampling point p' is computed by the sampling of the transfer object of $\boxed{\mapsto}^{\mathrm{I}}$ at each sampling point. For every valid p', the operand object $\mathbf{o}'$ is sampled where $\mathbf{o}'$ is a composite transfer object. The sampling of $\mathbf{o}'$ requires the sampling of the identity transfer object $\mathbf{id}^{\Phi}$ on the left of the tree. $\mathbf{id}^{\Phi}$ can be treated as a ghost object. $\forall p' \in \mathbf{id}^{\Phi}$, $\mathbf{o}$ is then sampled. If $p \notin \mathbf{s}$ or $p' \notin \mathbf{id}^{\Phi}$, the ray tracer then returns a status of sampling failures to the root node.

One of the benefits of the alternative $\boxed{\mapsto}^{\mathrm{I}}$ is that in order to modify the region of interest covered by $\mathbf{s}$, the mapping function Ψ need not be changed. Modifications to the mapped region can be implemented through geometric transformations, such as rotation and translation. The mapping region can also be extended through the scaling of the transfer object. By doing so, the transfer object is now dependent on the position of the $\mathbf{id}^{\Phi}$ and any alteration to the orientation of $\mathbf{id}^{\Phi}$ may break the functionality of the transfer object.

One of the drawbacks of $\boxed{\mapsto}^{\mathrm{I}}$ is the loss of context that may occur where the region of interest is defined. The loss of context makes it difficult to judge the correctness and effectiveness of the $\boxed{\mapsto}^{\mathrm{I}}$. This drawback can be overcome by a different type of STF called *local STF* (or *type-L STF*), which is able to support context and focus deformation.

The implementation of $\boxed{\mapsto}^{\mathrm{L}}$ is similar to that of $\boxed{\Mapsto}^{\mathrm{L}}$. The context of deformation is defined by $\mathbf{o}$ whereas the focus of deformation is defined by $\mathbf{s}$, or $(\mathbf{s}_0, \ldots, \mathbf{s}_n)$ if $\mathbf{s}$ is a combinational transfer object. Given a subtree, $\boxed{\mapsto}^{\mathrm{L}}(\mathbf{s}, \mathbf{o})$, the valid spatial domain is the same as that of $\mathbf{s}\boxed{\cup}\mathbf{o}$. A new sampling point p' is computed by evaluating $\mathbf{s}$ at a sampling point $p \in \mathbf{s}$. Similar to $\boxed{\mapsto}^{\mathrm{I}}$, the ray tracer will not return a sampling failure if $p \notin \mathbf{s}$. In this situation p' would still be considered as a valid sampling point and the operand object of $\boxed{\mapsto}^{\mathrm{L}}$, $\mathbf{o}$ is evaluated. The ray tracer, however, returns a status of sampling failure to the root if $p' \notin \mathbf{o}$.

If $\mathbf{s}$ is, however, a combinational transfer object and an overlapping of their mapped regions occurs then their combinational operator may not be as direct as $\boxed{\Mapsto}^{\mathrm{L}}$. This is because the combined points are now used instead of their relative distance, which is the case for $\boxed{\Mapsto}^{\mathrm{L}}$. The results that are produced may not be what are expected.

An alternative version of $\boxed{\mapsto}^{\mathrm{L}}$ can be defined by specifying the context of deformation using an identity transfer object, $\mathbf{id}^{\Phi}$. In this implementation, $\mathbf{id}^{\Phi}$ is treated as a ghost object. Similar in the previous approach, the transfer object of $\boxed{\mapsto}^{\mathrm{L}}$ is evaluated in the calculation of a new sampling point, p'. For every valid p', the operand object of $\boxed{\mapsto}^{\mathrm{L}}$ is sampled. The operand object of $\boxed{\mapsto}^{\mathrm{L}}$ is now defined as a composite transfer object $(\boxed{\mapsto}^{\mathrm{N}}(\mathbf{id}^{\Phi}, \mathbf{o}))$. The operand object $\mathbf{o}$ is computed then $\forall p' \in \mathbf{id}^{\Phi}$. The ray tracer returns a status of sampling failure to the root node if $p' \notin \mathbf{id}^{\Phi}$ or $p' \notin \mathbf{o}$.

This effect can be seen in Figure 6.12(b), where based on $\mathbf{s}$, $\boxed{\mapsto}^{\mathrm{L}}$ performs a local deformation in the mapped region (shown in pink) of $\mathbf{o}$, while leaving the unmapped region unchanged. Figures 6.2(c)-(d) are rendered using the $\boxed{\mapsto}^{\mathrm{L}}$ implementation.

The third type of STF that is to be discussed is *type-N STF* and can be denoted as $\boxed{\mapsto}^{\mathrm{N}}$. The definition of $\boxed{\mapsto}^{\mathrm{N}}$ is similar to that discussed by Chen *et al.* [CSW$^+$03]. The concept of $\boxed{\mapsto}^{\mathrm{N}}$ can be seen in Figure 6.12(c). In $\boxed{\mapsto}^{\mathrm{N}}(\mathbf{d}, \mathbf{o})$, the deformation is generally defined in a normalised volume domain of $[0, 1]^3$ with the assumptions that $\mathbf{s}$ and $\mathbf{o}$ are tightly coupled together in the normalised volume domain. Given that $p \in \mathbf{s}$, it is first mapped from world coordinates to normalised volume coordinates, $p \in [0, 1]^3$, and later transferred to a new sampling point, p'. The ray tracer then samples $\mathbf{o}$ at p' in normalised volume domain. In a transfer function specification, any p that does not belong to the domain would have resulted in a sampling failure.

6.9 Summary

In this chapter, a new algebraic framework for modelling complex deformation in a constructive manner has been presented. We have also introduced the concept of *spatial displacement function* (SDF) as the fundamental building block for the algebraic framework. A comprehensive analysis of SDFs and STFs has been carried out and its results can be seen throughout the chapter. In this analysis, it has been demonstrated that SDF is functionally more powerful and algebraically more sophisticated than STF. In particular, it has been shown that SDF is able to handle the general identity and inverse definitions in a more precise manner than STF. From the perspective of volume deformation, SDFs enable the specifications of deformation to be modelled, combined and manipulated in the same manner as that of conventional volume objects.

Chapter 7

Colour-Spectrum Conversions

Contents

7.1 Introduction 141
7.2 Metamerism 142
7.3 Related Work 143
7.4 Reduction of the Negative Energy through Negative Minimisation . 148
7.5 Reduction of the Negative Energy through Shifting Fourier Basis . 149
7.6 Evaluation of Results 151
7.7 Summary 160

7.1 Introduction

The RGB space is perhaps the most commonly implemented colour representation in rendering software and graphics platforms. However, the RGB representation presents us with several limitations [Hal89]. One of its limitations is the complication of rendering physical optics phenomena, which requires spectral calculations [Pee93, JF99]. Despite its limitations, RGB representation has been embedded as a standard in most rendering software and graphics platforms. This is due to the fact that its use is more intuitive for an average user in comparison with spectral calculations which require some knowledge of colour science and colour-vision models. Therefore, in this chapter a collection of colour-spectrum conversion algorithms is presented which can be implemented at the back-end of a graphics system while still allowing the system to maintain its RGB representation at the front-end.

The process of colour-spectrum conversions is normally categorised into two different types of conversions, which can be classified as *colour-to-spectrum* conversions and *spectrum-to-colour* conversions. A spectrum-to-colour conversion is a direct method of mapping a spectrum to its corresponding RGB coordinates and it is considered to be a *many-to-one* relationship. A colour-to-spectrum conversion, however, is a more complex procedure as it is a *one-to-many* relationship whereby a specified RGB coordinates can corresponds to several different spectra. This type of relationship is commonly known as a *metamer*.

In this chapter, a comprehensive study on a collection of colour-spectrum conversion algorithms is presented. The study comprises a detailed investigation on a number of colour-spectrum conversion algorithms, in particular of the combination of linearly independent basis functions; *Fourier* [Gla89] and *Gaussian* [SFCD99]. The study of the colour-spectrum conversion algorithms also highlights on the advantages and disadvantages that can be found in each of the linearly independent basis functions. One common disadvantage that can be found in each of the basis functions is the likelihood of negative energy in the reconstructed spectrum that belongs to RGB coordinates of a highly saturated colours [SFCD99]. In this chapter, we present two forms of modification that can be implemented with Fourier basis functions that can minimise the negative energy that may be present in the reconstructed spectrum.

The organisation of this chapter is as follows:

- In Section 7.2, the concept of metamerism, which is referred to throughout the chapter, is examined in detail.
- In Section 7.3, a brief overview of the existing colour-to-spectrum conversion algorithms is given.
- In Sections 7.4 and 7.5, the modifications that can be implemented when using Fourier basis functions is presented.
- In Section 7.6, the results of the implementation are presented. In this section, an overview of the reflectance data sets used is given and an analysis of the results is performed. The results and analysis will demonstrate that although our newly presented methods do not generate the perfect results, they are an improvement in comparison with its predecessor methods.

Finally in Section 7.7, our observations and concluding remarks are presented.

7.2 Metamerism

The spectrum reconstructed by a colour-to-spectrum algorithm may only be a metamer of the original spectrum. *Metamerism* [Ber00, The06c] occurs when different spectra, which are of varied physical properties, are perceived to be of the same colour. Metamerism is widely encountered in the process of colour reproduction. Typical examples of colour reproduction can be seen in media such as television, photography and printing where an exact reproduction of a scene can be reproduced through these media with close representation of the original scene. However, metamerism presents us with a problem whereby the reproductions may match under a specific environment, but may not match under all environments. This causes a colour mismatch.

Metamerism can be divided into four different categories. These categories can be classified as [Rig97]:

Illuminant metamerism. Illuminant metamerism is the most common type of metamerism that can be seen. Illuminant metamerism occurs when sample colours are perceived to be the same under a specific illuminant but may not be perceived to be the same when viewed under a different illuminant.

Observer metamerism. Observer metamerism, however, occurs when sample colours are perceived to be the same under a specific viewing system, *e.g.*, the eye, but are different under a different viewing system, *e.g.*, a camera.

Field size metamerism. Field size metamerism occurs when sample colours are perceived to be the same when viewed under a particular field of view. Typical field of views that can be found in computer graphics are 2 ° and 10 °.

Geometric metamerism. Geometric metamerism, however, occurs when sample colours are seen to be the same under the same viewing geometry specifications.

The concept of metamerism can be observed alongside the subject of *colour constancy* [Lan77, MW86, DI93] whereby the perceived colour of an object remains the same regardless of its changing illumination conditions.

A group of different spectra that provides us with the same colour perception can be classified as a *metamer set* [FM99]. Each metamer set is defined by the *metamer crossover* and *colour probabilities*. In the study of metamer crossover [Tho73, FM00], it is discovered that the members of a metamer set intersect with each other at least three times across the visible spectrum. These crossover points can be known as the *prime wavelengths* [BFHT98] and correspond to $450nm$, $540nm$ and $610nm$, respectively. In the study of colour probabilities, however, investigation into the number of metamer that can be found for a specified sensor response is carried out.

7.3 Related Work

The process of reconstructing a spectrum for a given RGB coordinates is complex. One of the reasons, as mentioned briefly in the previous section, is that specified RGB coordinates can correspond to multiple spectra. Therefore, the spectrum reconstructed may only be a metamer of the original spectrum. There are many different types of algorithms that can be implemented for the reconstruction of a spectrum. These algorithms range from the straightforward *delta* [Gla89] and *box functions* [Hal89, HG83] to the more complex combination of linearly independent basis functions of *Fourier* [Gla89] and *Gaussian* [SFCD99].

The delta and box functions are considered to be two of the most simple approaches in the reconstruction of spectrum. The delta function can be mathematically defined as:

$$\delta(x) = \begin{cases} 1 & x = 0 \\ 0 & x \neq 0 \end{cases}$$

which can then be used in the reconstruction of a spectrum, such that, [Gla89]

$$F_i(\lambda) = \delta(\lambda - \lambda_i) = \begin{cases} 0 & \text{if } \lambda \neq \lambda_i \\ 1 & \text{otherwise} \end{cases}$$

where $i = 1, 2, 3$ and λ is the wavelength. λ_i represents the three different wavelengths of red, green and blue colours, respectively, and their values are specified to be $590nm$, $560nm$ and $440nm$. Such values are chosen as they hold the largest values in the sample of $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, $\bar{z}(\lambda)$ in the colour matching functions CIE XYZ [Gla89]. The reconstructed spectrum, S, can be computed as:

$$S(\lambda) = w_1 F_1(\lambda) + w_2 F_2(\lambda) + w_3 F_3(\lambda)$$

where w_1, w_2 and w_3 can be defined in matrix $\mathbf{W} = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$. $\mathbf{W}$ can be calculated as:

$$\mathbf{W} = \mathbf{X}\mathbf{C}^{-1}$$

where $\mathbf{X}$ is the XYZ coordinates and $\mathbf{C}$ is defined as [Gla89]:

$$\mathbf{C} = \begin{bmatrix} 1.026 & 0.757 & 0.001 \\ 0.594 & 0.995 & 0.004 \\ 0.348 & 0.023 & 1.747 \end{bmatrix}$$

$\mathbf{C}$ is constructed from the values of the colour matching values at the three different wavelengths of $590nm$, $560nm$ and $440nm$ [Gla89].

The box functions can be defined to be [Hal89, HG83]:

$$F_i(\lambda) = \begin{cases} 1 & \lambda_i < \lambda < \lambda_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

where $i = 1, 2, 3$ and λ_i and λ_{i+1} represents the red, green and blue regions, which can be specified to be $400nm$, $500nm$, $600nm$ and $700nm$, respectively. These functions can be extended to include *fitting functions* [Mey88], such as the Fourier [Gla89] and Gaussian [SFCD99] which are investigated in Sections 7.3.1 and 7.3.2, respectively.

An alternative approach to the reconstruction of a spectrum in comparison with the rest of the conversion algorithms has been presented by Smits [Smi99] in 1999. This approach is considered to be different in the sense that instead of producing just a single spectrum, the algorithm 'retrieves' a spectrum that best matches a set of conditions after an exploration of its metamer space. By doing so, the spectrum reconstructed will be a 'good' spectrum as it would match a set of specific conditions. The set of specific conditions is set to ensure that the spectrum follows the physical laws of nature, such as smoothness and validity of the spectrum range value (*i.e.,* between the range value of zero and one). An application of this method has been implemented by Ward and Eydelberg-Vileshin [WE02].

Wang *et al.* [WXS04] have also presented an alternative approach for the reconstruction of a spectrum whereby given a colour coordinate in a colour space, a tetrahedron containing the colour coordinates is generated and from this a spectrum can be reconstructed. The algorithm utilises a set of base spectra as its search space. The base spectra are constructed from the union of a set of measured real reflectances and a set of spectra generated through the Bouguer-Beer's law [WS82]. Extensions of this algorithm have been investigated by Vadakkumpadan and Sun [VS05] and Vadakkumpadan *et al.* [VWS05].

7.3.1 Fourier Basis

The implementation of Fourier functions as basis functions in the reconstruction of a spectrum was first presented by Horn [Hor84] in 1984. The implementation of the Fourier basis have also been investigated by Glassner [Gla89] and Wandell [Wan87]. Drew and Funt [DF92] have also presented an analysis of the Fourier basis through the concept of metamerism. The difference between the rest of the mentioned work and Glassner's is that the Fourier basis implemented by Glassner is connected to a specified monitor RGB, whereas in the others, the Fourier basis are independent of the RGB monitor specifications.

The reconstruction of the spectrum that is to be presented here is an XYZ-to-spectrum conversion instead of an RGB-to-spectrum as it is less complicated to reconstruct a corresponding spectrum for specific XYZ coordinates than that of RGB coordinates [Gla89]. Given a spectrum, S, its specific XYZ coordinates can be calculated using the 1931 CIE colour-matching functions, $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, $\bar{z}(\lambda)$, as:

$$\begin{aligned} X &= \int_\lambda S(\lambda)\bar{x}(\lambda)\,d\lambda \\ Y &= \int_\lambda S(\lambda)\bar{y}(\lambda)\,d\lambda \\ Z &= \int_\lambda S(\lambda)\bar{z}(\lambda)\,d\lambda \end{aligned} \tag{7.1}$$

and the XYZ coordinates can be translated to an RGB coordinates as [FvDFH95]:

$$\begin{bmatrix} R & G & B \end{bmatrix} = \begin{bmatrix} X & Y & Z \end{bmatrix} \quad \mathbf{M}$$

where $\mathbf{M}$ is a 3×3 transformation matrix constructed from the chromaticities of a specific monitor.

The Fourier basis [Gla89] consists of a flat spectrum and a single cycle of sine and cosine, respectively. The Fourier basis can be mathematically defined as:

$$\begin{aligned} F_1(\lambda) &= 1.0 \\ F_2(\lambda) &= \sin\left(2\pi \frac{\lambda - \lambda_{min}}{\lambda_{max} - \lambda_{min}}\right) \\ F_3(\lambda) &= \cos\left(2\pi \frac{\lambda - \lambda_{min}}{\lambda_{max} - \lambda_{min}}\right) \end{aligned} \tag{7.2}$$

where λ is the wavelength and λ_{min} and λ_{max} represent the minimum and maximum wavelengths of the visible range. A matrix, $\mathbf{D}$, of the Fourier basis can be built using the Fourier basis specified in Equation 7.2. $\mathbf{D}$ can be defined as:

$$\mathbf{D} = \begin{bmatrix} \int_\lambda F_1(\lambda)\bar{x}(\lambda)d\lambda & \int_\lambda F_1(\lambda)\bar{y}(\lambda)d\lambda & \int_\lambda F_1(\lambda)\bar{z}(\lambda)d\lambda \\ \int_\lambda F_2(\lambda)\bar{x}(\lambda)d\lambda & \int_\lambda F_2(\lambda)\bar{y}(\lambda)d\lambda & \int_\lambda F_2(\lambda)\bar{z}(\lambda)d\lambda \\ \int_\lambda F_3(\lambda)\bar{x}(\lambda)d\lambda & \int_\lambda F_3(\lambda)\bar{y}(\lambda)d\lambda & \int_\lambda F_3(\lambda)\bar{z}(\lambda)d\lambda \end{bmatrix}. \tag{7.3}$$

where it can be used in the calculation of the matrix, $\mathbf{W} = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$, for calculating the spectrum to be reconstructed.

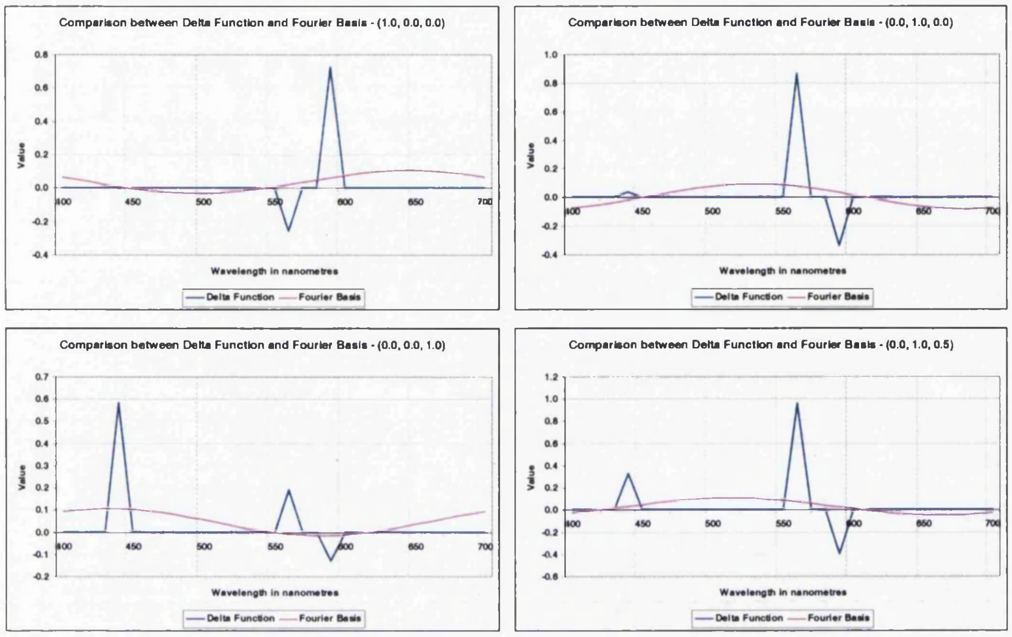


Figure 7.1: Comparison between the delta function and Fourier basis on four different RGB coordinates.

The values of the matrix, $\mathbf{W}$, can be computed as:

$$\mathbf{W} = \mathbf{X}\mathbf{D}^{-1} \tag{7.4}$$

where $\mathbf{X}$ is the XYZ coordinates, $\mathbf{X} = \begin{bmatrix} X & Y & Z \end{bmatrix}$, of the reconstructed spectrum and $\mathbf{D}^{-1}$ is the inverse matrix of $\mathbf{D}$ defined in Equation 7.3. Substituting Equation 7.3 in Equation 7.4 to obtain $\mathbf{W}$, the spectrum to be reconstructed can be computed as follows:

$$S(\lambda) = w_1 F_1(\lambda) + w_2 F_2(\lambda) + w_3 F_3(\lambda).$$

Figure 7.1 shows example spectra reconstructed using the delta function and Fourier basis for four different RGB colours of $(1.0, 0.0, 0.0)$, $(0.0, 1.0, 0.0)$, $(0.0, 0.0, 1.0)$ and $(0.0, 1.0, 0.5)$. These colours are chosen as they are highly saturated. The spectra reconstructed from the Fourier basis are smoother in comparison with the ones reconstructed from the delta function. It is observed that the spectra reconstructed from the delta functions are of zero values at almost all of the visible wavelengths with the exception of the three peaks for red, green and blue wavelengths. Although the spectra reconstructed from the Fourier basis are smoother, the spectra may contain negative energy, which contradicts the physical laws of nature as illustrated in Figure 7.1. This negative energy typically occurs when the specified RGB coordinates belong to highly saturated colours [SFCD99]. One of the solutions in solving this problem would be to reassign the negative values to zero. However, this would only introduce errors in the reconstructed spectra.

7.3.2 Gaussian Basis

Besides using Fourier functions in the reconstruction of spectra, Gaussian functions can also be implemented as basis functions in the colour-to-spectrum conversions, as demonstrated by Sun *et al.* in 1999 [SFCD99]. The Gaussian basis in the colour-to-spectrum conversions can be defined as:

$$F_i(\lambda) = e^{\ln 2(2(\lambda-\lambda_{c,i})/l_i)^2} \tag{7.5}$$

where $i = 1, 2, 3$ and $\lambda_{c,i}$ represents the centre of the Gaussian functions at $680nm$, $550nm$ and $420nm$, respectively. These values are chosen as they are located in the region of red, green and blue, respectively.

The width of the Gaussian functions, l_i, defined in Equation 7.5, can be calculated as:

$$\begin{aligned} l_1 &= \chi_{12}(\mathbf{R})l_{\text{min}} + (1 - \chi_{12}(\mathbf{R}))l_{\text{max}} \\ l_3 &= \chi_{32}(\mathbf{R})l_{\text{min}} + (1 - \chi_{32}(\mathbf{R}))l_{\text{max}} \\ l_2 &= \min(l_1, l_3). \end{aligned}$$

where l_{min} and l_{max} are defined to be $10nm$ and $150nm$, respectively. χ_{ij} can be defined as:

$$\chi_{ij}(\mathbf{R}) = \chi_{ji}(\mathbf{R}) = \frac{|r_i - r_j|}{r_i + r_j}.$$

where $i \neq j$ and $\mathbf{R}$ holds an RGB tuple of (r_1, r_2, r_3). A matrix, $\mathbf{D}$, of the Gaussian basis can then be built using the Gaussian basis specified in Equation 7.5. $\mathbf{D}$ can be specified as:

$$\mathbf{D} = \begin{bmatrix} \int_\lambda F_1(\lambda)\bar{x}(\lambda)d\lambda & \int_\lambda F_1(\lambda)\bar{y}(\lambda)d\lambda & \int_\lambda F_1(\lambda)\bar{z}(\lambda)d\lambda \\ \int_\lambda F_2(\lambda)\bar{x}(\lambda)d\lambda & \int_\lambda F_2(\lambda)\bar{y}(\lambda)d\lambda & \int_\lambda F_2(\lambda)\bar{z}(\lambda)d\lambda \\ \int_\lambda F_3(\lambda)\bar{x}(\lambda)d\lambda & \int_\lambda F_3(\lambda)\bar{y}(\lambda)d\lambda & \int_\lambda F_3(\lambda)\bar{z}(\lambda)d\lambda \end{bmatrix}. \tag{7.6}$$

where it is later used in the computation of the matrix, $\mathbf{W} = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$, as shown in Equation 7.4. The spectrum to be reconstructed can then be specified as follows:

$$S(\lambda) = w_1F_1(\lambda) + w_2F_2(\lambda) + w_3F_3(\lambda).$$

Figure 7.2 shows example spectra that have been reconstructed using both the Fourier and Gaussian basis for four different RGB colours of $(1.0, 0.0, 0.0)$, $(0.0, 1.0, 0.0)$, $(0.0, 0.0, 1.0)$ and $(0.0, 1.0, 0.5)$. In the all of the four colours, it can observed that although the spectra generated from the Gaussian basis contained more interesting curves in comparison with the Fourier basis, the Gaussian basis is considered to be more expensive computationally. This is because in the Gaussian basis, re-computation is required for each individual basis function as they are dependent on the RGB colour input. For RGB coordinates belonging to highly saturated colours, the spectra reconstructed may contain negative energy [SFCD99]. This drawback can be seen in Figure 7.2 and is similar to that discussed early on in Section 7.3.1.

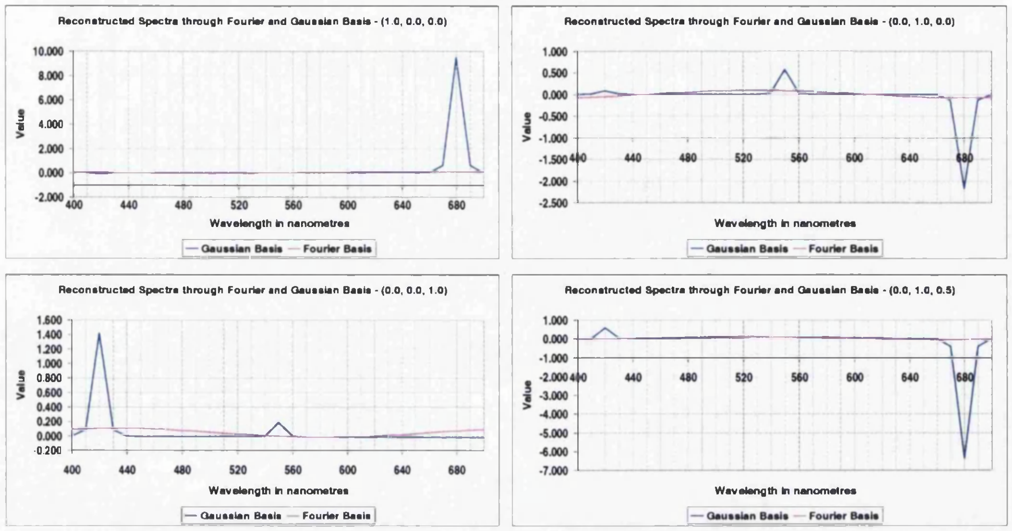


Figure 7.2: Comparison between the Fourier and Gaussian basis on four different RGB coordinates.

7.4 Reduction of the Negative Energy through Negative Minimisation

As previously discussed, the spectra reconstructed using Fourier basis may contained negative energy. In this section, one of the modifications that can be implemented on the Fourier basis is presented. The modification is introduced as a method for reducing the negative energy in the reconstructed spectrum. Given a reconstructed spectrum, S_{pre}, a new spectrum, U_{pre}, can be computed as:

$$U_{\text{pre}}(\lambda) = \begin{cases} -S_{\text{pre}}(\lambda) & \text{if } S_{\text{pre}}(\lambda) < 0 \\ 0 & \text{otherwise.} \end{cases} \tag{7.7}$$

U_{pre} is designed to hold the negative energy that may be present in S_{pre} and to find the contribution of the negative energy. The XYZ coordinates of U are then calculated as:

$$X_{\text{pre}} = -\int_{\lambda} U_{\text{pre}}(\lambda)\bar{x}(\lambda)d\lambda$$
$$Y_{\text{pre}} = -\int_{\lambda} U_{\text{pre}}(\lambda)\bar{y}(\lambda)d\lambda$$
$$Z_{\text{pre}} = -\int_{\lambda} U_{\text{pre}}(\lambda)\bar{z}(\lambda)d\lambda$$

X_{pre}, Y_{pre} and Z_{pre} are later used in the calculation of a new matrix, $\mathbf{W}$, which can be calculated using Equation 7.4.

Using the newly calculated $\mathbf{W}$, a new spectrum V is created and can be defined as follows:

$$V(\lambda) = w_1F_1(\lambda) + w_2F_2(\lambda) + w_3F_3(\lambda). \tag{7.8}$$

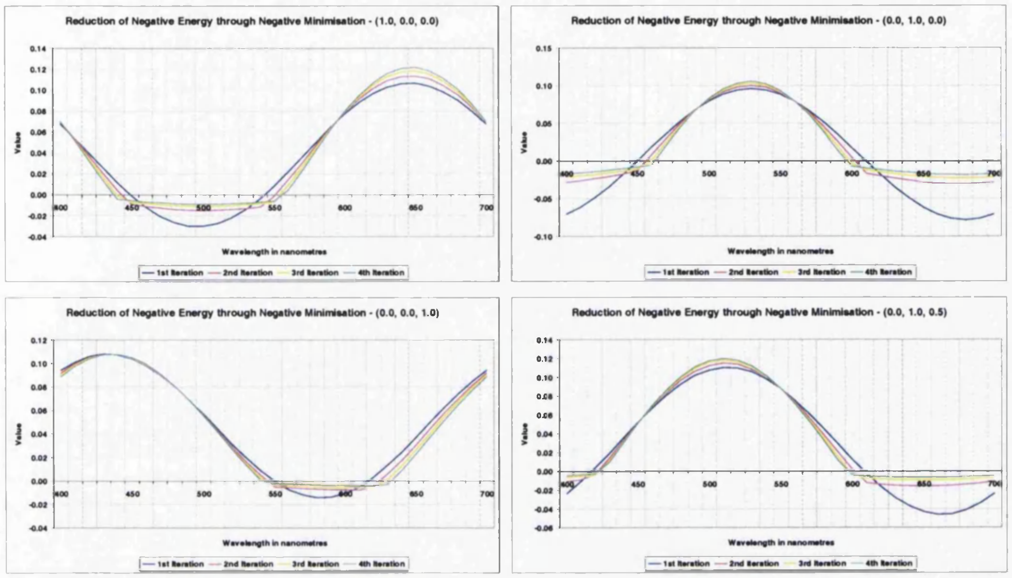


Figure 7.3: Four iterations of the reduction of the negative energy through negative minimisation on four different RGB coordinates.

$F_1(\lambda)$, $F_2(\lambda)$ and $F_3(\lambda)$ are the Fourier basis that has been defined in Equation 7.2. The new reconstructed spectrum, S_{cur}, can be computed as:

$$S_{\text{cur}}(\lambda) = S_{\text{pre}}(\lambda) + U(\lambda) + V(\lambda). \tag{7.9}$$

The modification steps are implemented as iterations and are computed until all of the negative energy has been removed or the sum of the current negative energy is not greater than the previous sum. Figure 7.3 shows the results of four iterations of the reduction of the negative energy on four different RGB colours of $(1.0, 0.0, 0.0)$, $(0.0, 1.0, 0.0)$, $(0.0, 0.0, 1.0)$ and $(0.0, 1.0, 0.5)$. The four RGB colours are chosen as they belong to a group of highly saturated colours and are therefore more likely to contain negative energy in their reconstructed spectra. In all of the four cases, it is observed that as the number of iterations increases the area of negative energy decreased.

7.5 Reduction of the Negative Energy through Shifting Fourier Basis

This section discusses the second modification that can be implemented on the Fourier basis for reducing the negative energy in a reconstructed spectrum. The reduction method explored in this section uses similar concepts as those presented in the previous section, but with a shifting Fourier basis. The Fourier basis changes according to the number of

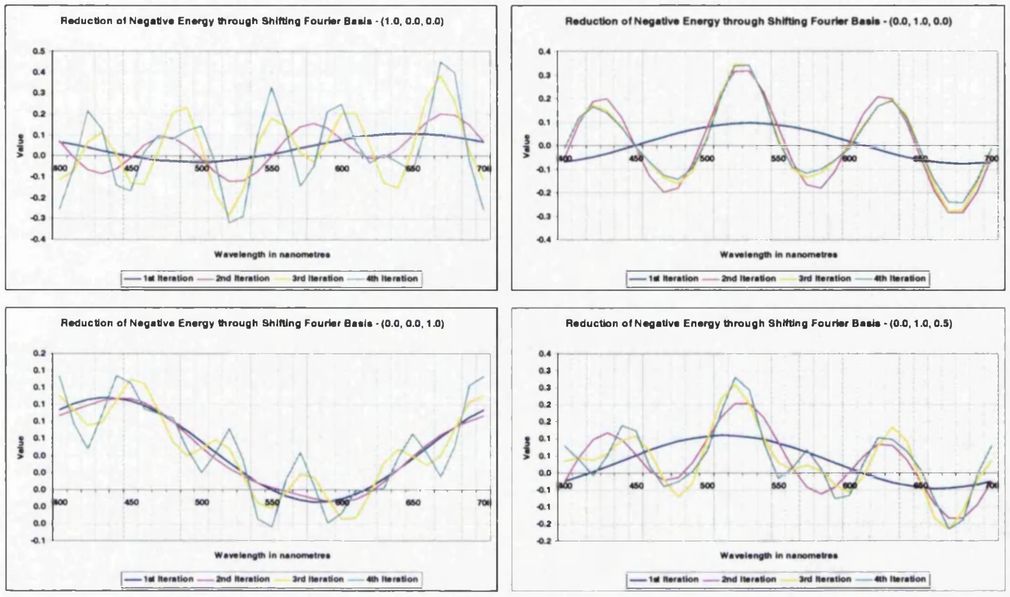


Figure 7.4: Four iterations of the reduction of the negative energy through shifting Fourier basis on four different RGB coordinates.

iterations and can be defined as:

$$F_{\text{pre},1}(\lambda) = 1.0$$
$$F_{\text{pre},2}(\lambda) = \sin\left(2\pi\frac{(\lambda - \lambda_{min})}{(\lambda_{max} - \lambda_{min})} \times (2i-1)\right)$$
$$F_{\text{pre},3}(\lambda) = \cos\left(2\pi\frac{(\lambda - \lambda_{min})}{(\lambda_{max} - \lambda_{min})} \times (2i-1)\right)$$

where i represents the number of current iteration and λ_{min} and λ_{max} are the minimum and maximum wavelengths of the visible range.

The final spectrum, S_{cur}, can be calculated as:

$$S_{\text{cur}}(\lambda) = S_{\text{pre}} + \frac{(U(\lambda) + V(\lambda))}{(2i-1)}$$

where $U(\lambda)$ and $V(\lambda)$ are as defined in Equations 7.7 and 7.8, respectively.

Similar to the previous reduction method, the functions are implemented as iterations and are calculated until all of the negative energy has been removed or the sum of the current energy is not greater than the previous sum. Figure 7.4 shows the results of four iterations of the implemented negative energy reduction on four different RGB colours of $(1.0, 0.0, 0.0)$, $(0.0, 1.0, 0.0)$, $(0.0, 0.0, 1.0)$ and $(0.0, 1.0, 0.5)$. It is observed that the spectra constructed using the shifting Fourier basis have a more interesting curve than the ones constructed using typical Fourier basis. In all of the four cases, it can be concluded that as the number of iterations increases, the area of negative energy also decreases.

7.6 Evaluation of Results

This section presents the results of the implementation of the conversion algorithms discussed in Sections 7.3.1, 7.3.2, 7.4 and 7.5. The conversion algorithms are implemented using two sets of input variables: a collection of reflectance data sets and a collection of RGB coordinates. The main objectives of the implementation are:

1. To investigate which of the colour-to-spectrum algorithms provide us with the best results with respect to the least amount of negative energy and
2. To examine which of the colour-to-spectrum algorithms are able to provide us with a reconstructed spectrum that best resembles the original spectrum [War04].

Throughout the section ,the algorithms implemented will be referred to as follows:

- **Method A:** Fourier Basis.
- **Method B:** Fourier Basis with negative energy reduction through negative minimisation.
- **Method C:** Fourier Basis with negative energy reduction through shifting Fourier Basis.
- **Method D:** Gaussian Basis.

The evaluation begins with the implementation of the colour-to-spectrum algorithms on a collection of reflectance data sets. Later in the section, the different reflectance data sets that are used in the implementations are examined and a discussion on the reconstructed spectra based on the two objectives is carried out. Finally, in Section 7.6.2 the implementation of the colour-to-spectrum conversions on a collection of RGB coordinates is examined and discussed.

7.6.1 Spectra Generated from Reflectance Data Sets

Reflectance Data Sets

The reflectance data sets that are used through the implementation comprise of a range of materials that are either synthetically made, such as denim and a Swiss army knife, or objects that can be found in nature (*e.g.,* grass and soil). The reflectance data sets can be briefly described as follows:

Agfa Color Reference Agfa Color Reference [Uni04] is created based on the IT8 standard and comprises of 288 patches of sample targets. The sample targets are created to represent the colour space and a full spectrum gamut.

Munsell Munsell reflectance are based on the *Munsell Color System* [Mun05, Mun15, War04], which has been divided into a three-dimensional space of hue, lightness and chroma.

Objects Objects [War04] is the reflectance spectra of a collection of everyday materials. The materials comprise of synthetically made objects and objects that can be found in nature.

Nature Nature [Uni04] is the reflectance spectra of samples collected from nature, such as flowers and leaves.

Dupont chips Dupont chips [War04] is the reflectance spectra of a collection of Dupont paint chips.

Macbeth Color Checker Macbeth Color Checker [MMD76, War04] is a commercially available product comprising of matte painted surfaces.

Light sources Light sources [War04] comprises of CIE standard illuminant and cool-white fluorescent bulb.

Table 7.1 presents the reflectance data sets together with its number of samples. Overall, the colour-to-spectrum algorithms are examined against 892 reflectance samples.

Reflectance Data Set	**No. of Sample**
Agfa Color Reference	288
Munsell	64
Objects	170
Nature	218
Dupont Chips	120
Macbeth Color Checker	24
Light Sources	8

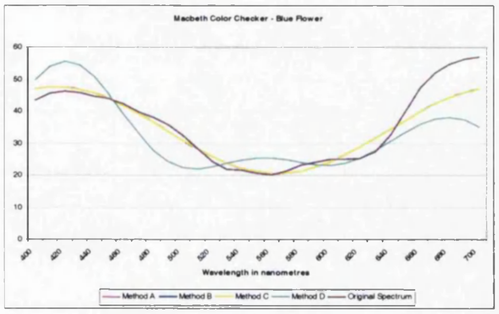
Table 7.1: Reflectance data sets with its number of samples.

Discussion

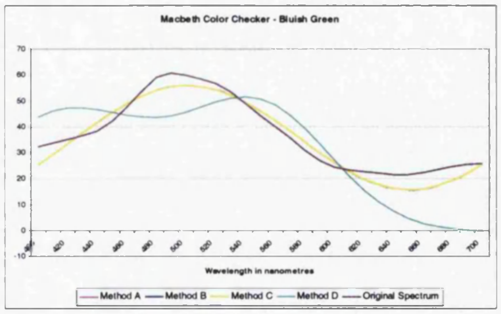
Table 7.2 shows the detailed results obtained for the implementations of the conversions algorithms using the Macbeth Color Checker as the input variables. The Macbeth Color Checker is chosen as it comprises not only the additive and subtractive primaries but also common natural colour such as skin colours and sky [Pas06]. The XYZ (or RGB) coordinates are generated using Equation(s) 7.1 (or 7.3.1). In this discussion, Objective 1 of the implementation is focused on. In order to fulfill the aim of Objective 1, the algorithms are examined as to which of them are able to reconstruct spectra that would contain the least amount of negative energy. The evaluation can be performed by examining the number of negative samplings and the amount of negative energy, which are represented by **Negs** and $\sum$ **Negs**, respectively. In addition, investigations are also carried out to examine the difference between the original spectrum and the reconstructed spectrum, which is represented by $\sum$ **Diff**. The three variables are later used as guidelines in finding which of the algorithms are able to reconstruct a spectrum with the least amount of negative energy.

Sample	Method A			Method B			Method C			Method D		
	$\sum$ Diff	Negs	$\sum$ Negs	$\sum$ Diff	Negs	$\sum$ Negs	$\sum$ Diff	Negs	$\sum$ Negs	$\sum$ Diff	Negs	$\sum$ Negs
Blue Flower	87.523	0	0	87.523	0	0	87.523	0	0	205.733	0	0
Bluish Green	101.784	0	0	101.784	0	0	101.784	0	0	350.591	2	0.337
Orange	217.561	2	1.009	217.561	2	1.009	217.561	2	1.009	1014.606	0	0
Purplish Blue	162.679	0	0	162.679	0	0	162.679	0	0	297.909	0	0
Moderate Red	248.958	1	0.042	248.909	1	Small num	248.915	1	0.006	1795.584	0	0
Purple	109.603	0	0	109.603	0	0	109.603	0	0	175.839	0	0
Yellow Green	309.547	5	7.461	302.661	5	Small num	303.869	5	1.331	426.867	0	0
Orange Yellow	252.012	0	0	252.012	0	0	252.012	0	0	808.046	0	0
Blue	187.370	0	0	187.370	0	0	187.370	0	0	401.299	0	0
Green	197.487	10	47.219	149.887	10	Small num	162.704	10	12.65	276.880	8	40.370
Red	459.287	8	47.834	372.374	10	6.945	440.240	8	37.020	3976.808	0	0
Yellow	383.784	0	0	383.784	0	0	383.784	0	0	941.046	0	0
Magenta	243.651	0	0	243.651	0	0	243.651	0	0	2230.94	0	0
Cyan	131.148	0	0	131.148	0	0	131.148	0	0	421.287	9	99.021
White	58.685	0	0	58.685	0	0	58.685	0	0	70.365	0	0
Neutral .8	16.683	0	0	16.683	0	0	16.683	0	0	36.866	0	0
Neutral 6.5	9.261	0	0	9.261	0	0	9.261	0	0	23.477	0	0
Neutral .5	3.968	0	0	3.968	0	0	3.968	0	0	14.797	0	0
Neutral 3.5	1.684	0	0	1.684	0	0	1.684	0	0	6.749	0	0
Black	1.197	0	0	1.197	0	0	1.197	0	0	1.960	0	0
Dark skin	105.534	0	0	105.534	0	0	105.534	0	0	104.436	0	0
Light skin	313.734	0	0	313.734	0	0	313.734	0	0	377.804	0	0
Blue sky	129.976	0	0	129.976	0	0	129.976	0	0	79.408	0	0
Foliage	119.972	3	0.355	119.628	3	Small num	119.629	3	0.005	110.022	0	0
Mean	**160.545**	**1.208**	**4.330**	**154.637**	**1.292**	**0.331**	**158.050**	**1.208**	**2.168**	**589.555**	**0.792**	**5.822**

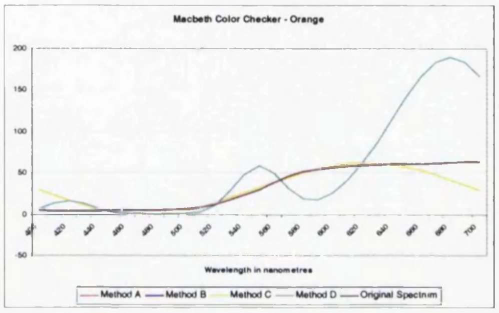
Table 7.2: The results of the Macbeth Color Checker based on the four methods.



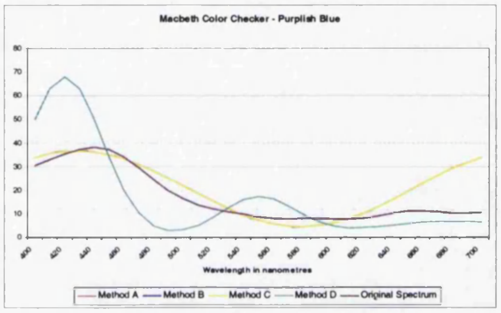
(a)



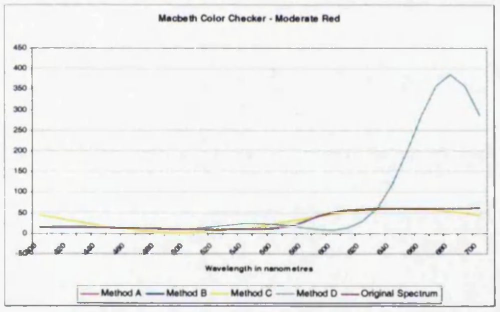
(b)



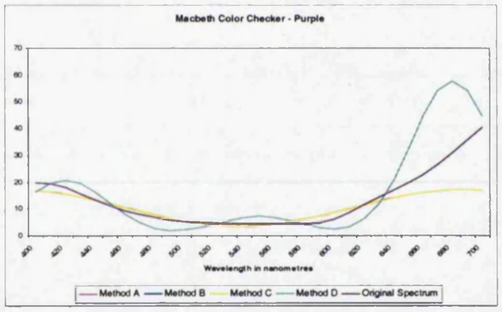
(c)



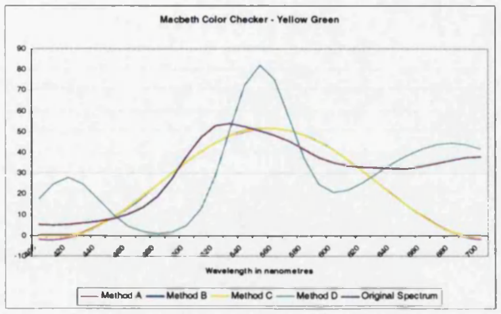
(d)



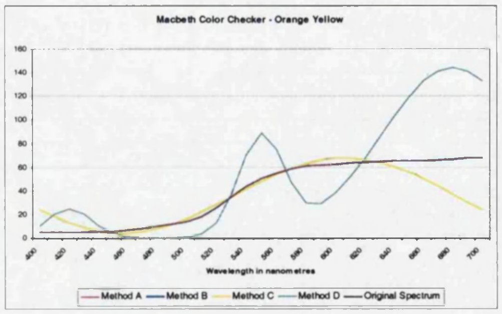
(e)



(f)

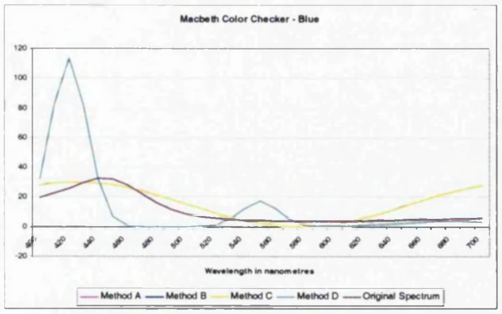


(g)

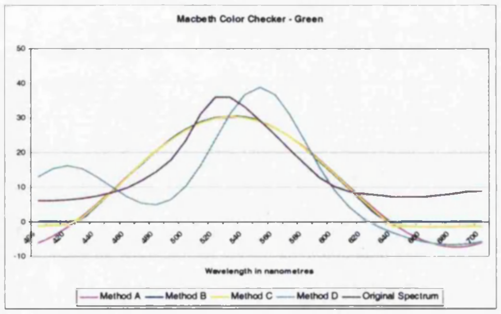


(h)

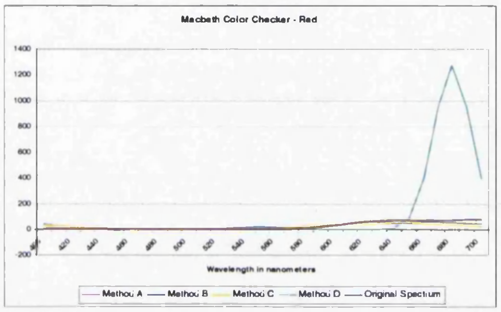
Figure 7.5: The spectra for the Blue Flower, Bluish Green, Orange, Purplish Blue, Moderate Red, Purple, Yellow Green and Orange Yellow for the Macbeth Color Checker.



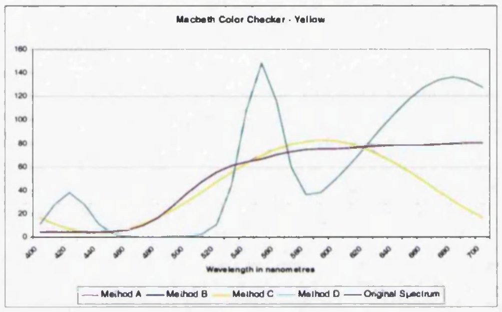
(a)



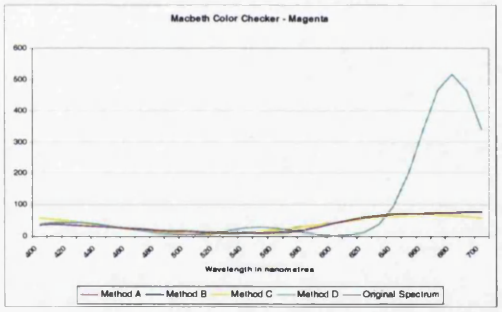
(b)



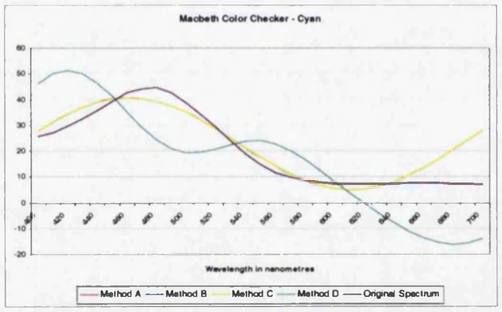
(c)



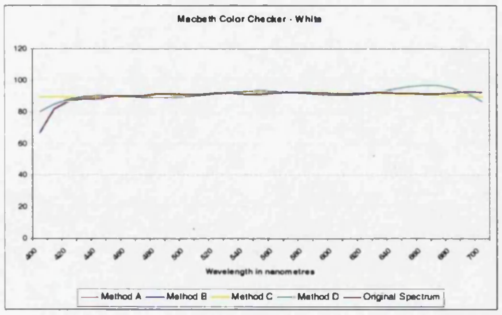
(d)



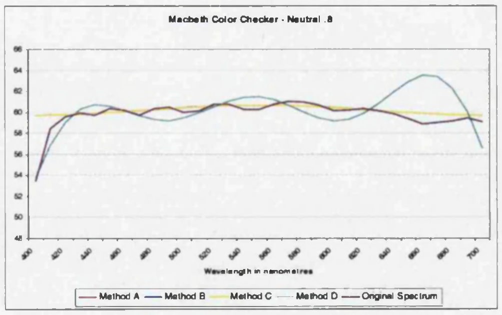
(e)



(f)

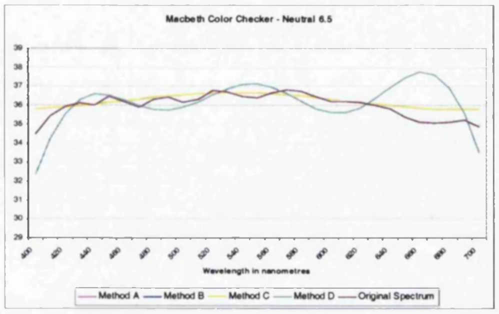


(g)

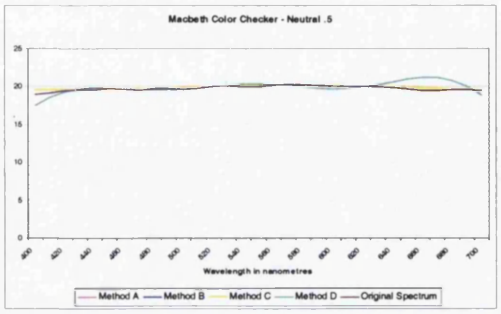


(h)

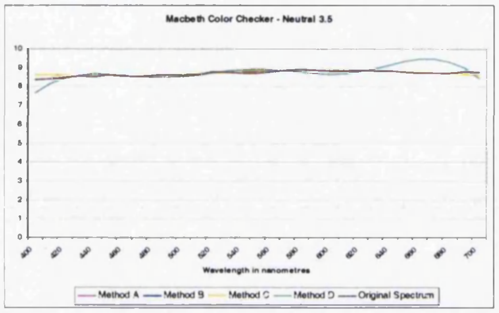
Figure 7.6: The spectra for the Blue, Green, Red, Yellow, Magenta, Cyan, White and Neutral .8 for the Macbeth Color Checker.



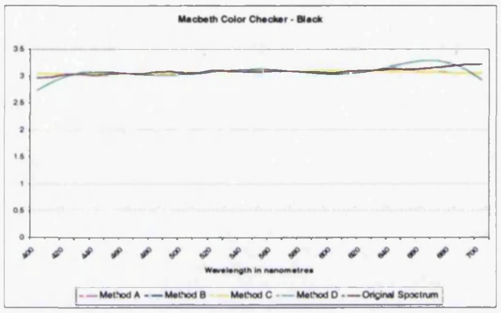
(a)



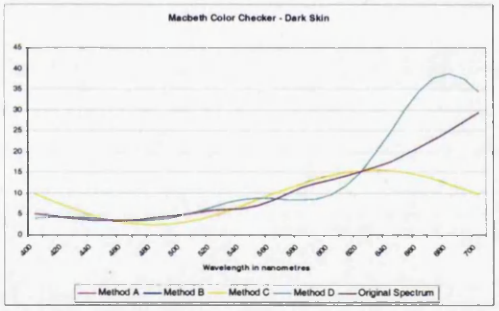
(b)



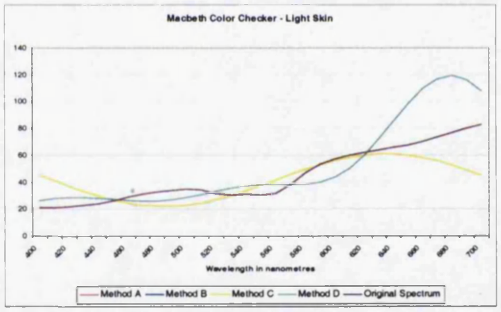
(c)



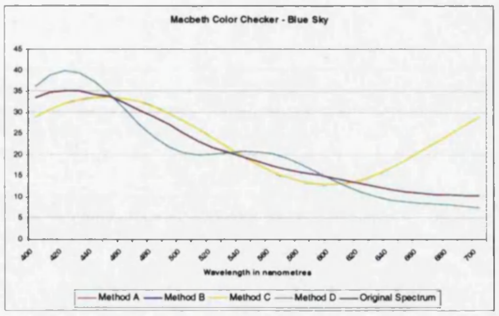
(d)



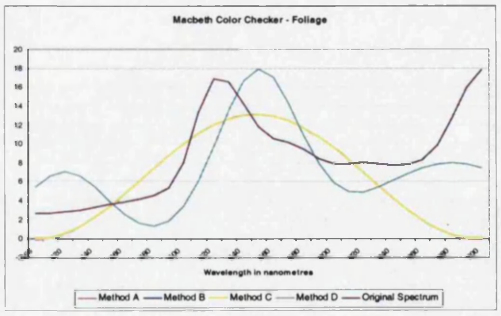
(e)



(f)



(g)



(h)

Figure 7.7: The spectra for the Neutral 6.5, Neutral .5, Neutral 3.5, Black, Dark Skin, Light Skin, Blue Sky and Foliage for the Macbeth Color Checker.

Reflectance	**Method A**				**Method B**			
	$\sum$ Diff	**Negs**	**$\sum$ Negs**	**No. Negs**	**$\sum$ Diff**	**Negs**	**$\sum$ Negs**	**No. Negs**
Agfa Color Reference	120.098	0.173	0.253	10	119.832	0.183	0	10
Munsell	1.653	1.328	0.059	13	1.581	1.547	0.006	13
Objects	1.465	0.659	0.022	22	1.439	0.747	0.004	22
Nature	6503.64	0.393	45.544	17	6443.216	0.475	12.362	17
Dupont Chips	1.914	2.867	0.148	45	1.767	3.325	0.057	45
Macbeth Color Checker	160.545	1.208	4.33	6	154.637	1.292	0.331	6
Light Sources	91.369	2.875	10.520	2	91.369	2.875	10.520	2

Table 7.3: Mean results for Methods A and B.

Reflectance	**Method C**				**Method D**			
	$\sum$ Diff	**Negs**	**$\sum$ Negs**	**No. Negs**	**$\sum$ Diff**	**Negs**	**$\sum$ Negs**	**No. Negs**
Agfa Color Reference	119.904	0.173	0.062	10	245.526	0.481	3.513	19
Munsell	1.621	1.469	0.031	13	6.589	1.344	0.117	11
Objects	1.455	0.676	0.014	22	2.713	0.247	0.032	5
Nature	6484.661	0.420	33.768	17	16049.014	0	0	0
Dupont Chips	1.853	3.058	0.103	45	9.133	1.717	0.172	24
Macbeth Color Checker	158.050	1.208	2.168	6	589.555	0.792	5.822	3
Light Sources	91.369	2.875	10.520	2	210.678	5.625	35.154	2

Table 7.4: Mean results for Methods C and D.

In Table 7.2, it can be observed that the poorest to the best methods are Method D, Method A, Method C and Method B. Method B can be considered as the best method as its reconstructed spectra contain the least amount of negative energy and that it also provides us with the lowest mean for the difference between the original and reconstructed spectra. Method D can be considered as the poorest method as its reconstructed spectra contain the largest amount of negative energy in comparison with the rest of the method. Method D can also be considered the poorest method as it has the highest mean for the difference between the original and reconstructed spectra. Overall, it can be said that Method B, which is the Fourier basis with negative minimisation, is the best method for colour-to-spectrum conversions and Method D, which is the Gaussian basis, can be considered as the poorest method for conversions algorithms.

Although Method D is considered to be the poorest method, it is observed that Method D has the lowest mean for the number of negative samplings. Another observation that can be made is that it does not produce any negative samplings for sample Orange, Moderate Red, Red and Foliage while the rest of the other methods do. Even though there are no negative samplings, the difference between the original and reconstructed spectra for the four samples is considerably higher in comparison with the rest of the methods.

The original and reconstructed spectra of the Macbeth Color Checker using the four methods are plotted in Figures 7.5, 7.6 and 7.5. The purpose of the graphs is to examine which of the algorithms are able to provide us with the closest representation of the original spectra (see Objective 2). From the graphs, Methods A, B and C can be considered as the best methods for providing a near representation of the actual spectra with 66.67% of the plotted graphs of a near representation of the actual spectra while Method D is seen as the poorest with only 25% of the plotted graphs are a near representation of the actual spectra. Although Methods A, B and C are considered as the best methods, they lack the rich curves that Method D able to produce, which can be seen in Figures 7.6(b), 7.7(h) and 7.6(h), in successfully mimicking the peaks in the spectra.

Tables 7.3 and 7.4 show the overall summary of the mean results of the four methods using the reflectance data set described in Section 7.6.1. The summary of the results confirmed that the poorest to the best methods are Method D, Method A, Method C and Method B. It can be seen that Methods B and C are able to reduce the amount of negative energy found in the spectra reconstructed using the Fourier basis. An observation is that even though Method D produced no negative sampling for the Nature reflectance data set, there is a significant difference between the original and reconstructed spectra.

7.6.2 Spectra Reconstructed from RGB coordinates

Table 7.5 shows the detailed results obtained for the implementation of the conversion algorithms using a collection of RGB coordinates as the input variables. It can be observed that the poorest to the best methods using a collection of RGB coordinates are Method D, Method A, Method C and Method B. As previously mentioned, Method B, which is the Fourier basis with negative elimination, can be considered as the best method as it provides us with the lowest mean for the amount of negative energy. Method D, which is the Gaussian basis, can be considered as the worst as it provides us with the highest mean for the amount

Sample	**Method A**		**Method B**		**Method C**		**Method D**	
	Negs	**$\sum$ Negs**	**Negs**	**$\sum$ Negs**	**Negs**	**$\sum$ Negs**	**Negs**	**$\sum$ Negs**
(0.0, 0.0, 0.0)	0	0	0	0	0	0	0	0
(0.5, 0.0, 0.0)	10	0.093	11	0.065	10	0.080	21	Small num
(1.0, 0.0, 0.0)	10	0.187	11	0.130	10	0.161	21	0.001
(0.0, 0.5, 0.0)	16	0.387	21	0.014	16	0.259	9	1.23
(0.0, 1.0, 0.0)	16	0.774	21	0.028	16	0.517	9	2.459
(0.5, 0.5, 0.0)	7	0.021	7	0.021	7	0.021	0	0
(1.0, 0.5, 0.0)	4	0.007	4	0.007	4	0.007	0	0
(0.5, 1.0, 0.0)	12	0.299	14	0.039	12	0.167	0	0
(1.0, 1.0, 0.0)	7	0.041	7	0.041	7	0.041	0	0
(0.0, 0.0, 0.5)	7	0.032	7	0.032	7	0.032	26	0.162
(0.0, 0.0, 1.0)	7	0.063	7	0.063	7	0.063	26	0.323
(0.5, 0.0, 0.5)	0	0	0	0	0	0	0	0
(1.0, 0.0, 0.5)	4	0.018	4	0.018	4	0.018	0	0
(0.5, 0.0, 1.0)	0	0	0	0	0	0	0	0
(1.0, 0.0, 1.0)	0	0	0	0	0	0	0	0
(0.0, 0.5, 0.5)	8	0.066	8	0.066	8	0.066	4	1.224
(0.0, 1.0, 0.5)	12	0.354	15	Small num	12	0.226	5	2.681
(0.0, 0.5, 1.0)	5	0.017	5	0.017	5	0.017	5	1.782
(0.0, 1.0, 1.0)	8	0.132	8	0.132	8	0.132	4	2.449
(0.5, 0.5, 0.5)	0	0	0	0	0	0	0	0
(1.0, 0.5, 0.5)	0	0	0	0	0	0	0	0
(0.5, 1.0, 0.5)	0	0	0	0	0	0	7	0.117
(0.5, 0.5, 1.0)	0	0	0	0	0	0	0	0
(1.0, 1.0, 0.5)	0	0	0	0	0	0	0	0
(0.5, 1.0, 1.0)	0	0	0	0	0	0	5	0.021
(1.0, 0.5, 1.0)	0	0	0	0	0	0	0	0
(1.0, 1.0, 1.0)	0	0	0	0	0	0	0	0
Mean	**4.926**	**0.092**	**5.556**	**0.025**	**4.926**	**0.067**	**5.259**	**0.461**

Table 7.5: The results of the RGB coordinates based on the four methods.

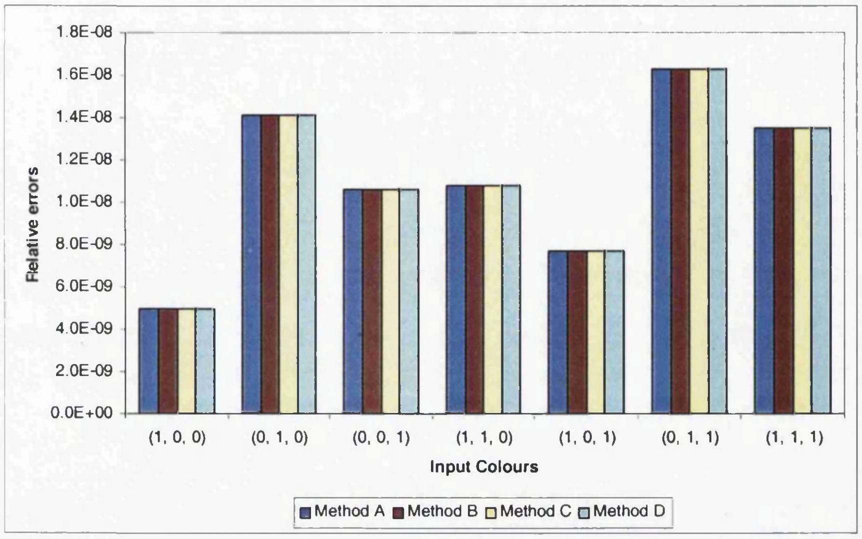


Figure 7.8: Relative errors between the input and output RGB coordinates.

of negative energy. The results also show that spectra reconstructed from RGB coordinates belonging to highly saturated colours are more likely to contain negative energy than that of low saturated colours [SFCD99].

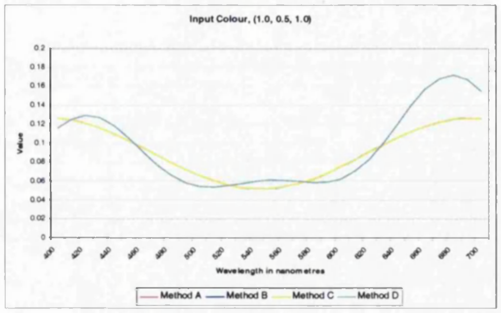
The relative errors of the four methods can be calculated by examining the input and output variables of the RGB coordinates. The relative errors can be calculated as follows [SFCD99]:

$$\epsilon = \frac{\sqrt{(c_1' - c_1)^2 + (c_2' - c_2)^2 + (c_3' - c_3)^2}}{\sqrt{c_1^2 + c_2^2 + c_3^2}}$$

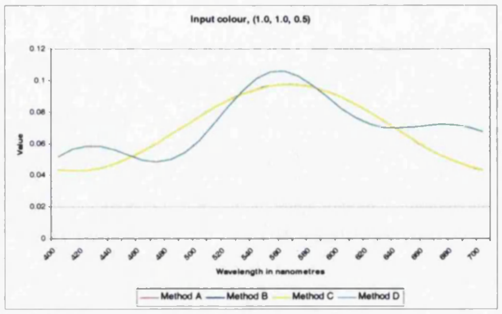
where (c_1, c_2, c_3) are the original input colours and (c_1', c_2', c_3') are the colours computed from the reconstructed spectra. Figure 7.8 shows the relative errors for the four methods. It is shown that the relative error for each of the methods are equal and are reasonably low. However, it should be noted that even though there may not be any relative error, the spectrum reconstructed may not be the same as the original spectrum but only a metamer of it [SFCD99]. Figure 7.9 shows some of the spectra that have been generated using the RGB coordinates.

7.7 Summary

This chapter began by emphasising the necessity and advantages of colour-to-spectrum conversions. Colour-to-spectrum conversions are required in order to overcome some of the limitations that can be found in an RGB representation. One of the advantages of colour-to-spectrum conversions is the ability to handle the rendering of physical optics phenomena such as light dispersion, interference and diffraction. The concept of metamerism plays an important role in the development of the colour-to-spectrum conversions. Most of the spec-



(a)



(b)

Figure 7.9: Some of the spectra that have generated using the RGB coordinates. The spectra reconstructed using Methods A, B and C are numerically similar.

tra that have been constructed by the colour-to-spectrum algorithms are only a metamer of the original spectrum.

In this chapter, a collection of colour-to-spectrum conversion algorithms have been investigated, in particular, the combination of linearly independent basis functions of *Fourier* [Gla89] and *Gaussian* [SFCD99]. In the course of the investigation, the negative energy problem that can occur in them has been highlighted. One of the solutions to the negative energy problem would be to reassign the negative values to zero. However, this would only introduce errors in the calculation of the reconstructed spectra. Two forms of modification to the Fourier basis have been introduced as a way of reducing the negative energy problem that may be present in the reconstructed spectra. We demonstrate that although the solutions to the negative energy problem do not generate perfect results, their results are an improvement on previous methods.

Chapter 8

Conclusions

Contents

8.1 Summary of contributions **163**
8.2 Future Work **165**

In this thesis, the issues of physically-based rendering and algebraic manipulation of volume model types have been addressed, with the focus on the algebraic properties and theory for volume data. This thesis provides a new approach to direct volume rendering that exhibits a better visual realism in its resulting images and an algebraic framework for modelling complex deformation in a constructive manner. We have achieved our objectives as outlined in Chapter 1 (see page 3). Specifically,

- Section 8.1.1 describes Objectives 1 and 2 of the new approach developed for direct volume rendering.
- Section 8.1.2 outlines Objectives 3, 4 and 5 of the comprehensive study on the algebraic framework of CVG and colour mixing techniques.
- Section 8.1.3 outlines Objectives 6, 7 and 8 of the algebraic framework for modelling complex deformation.
- Section 8.1.4 describes Objective 9 of the consideration and improvements on some of the existing colour-to-spectrum algorithms.

The introduction of a correct spectral volume rendering integral is essential in a graphical or visualisation system as it provides us with the capability to synthesise images in a manner that reflects the correct physical process of colour. The modelling and rendering of volumetric objects based on the Kubelka-Munk theory presents us with a mean of defining the volumetric relationship between opacity and transparency of a material and its absorption and scattering properties. In respect to the algebraic framework for modelling complex deformation, its introduction can be defined as a major extension of the concept of displacement mapping from a surface domain technique to a volume domain technique.

8.1 Summary of contributions

This thesis begins with literature reviews on the subjects of volume rendering and visualisation and colour models in computer graphics. The reviews form an integral part of this thesis as they provide the necessary groundwork that is required in our scientific investigations. The first review presents a survey of the commonly employed techniques in volume rendering and visualisation with the focus on object and volume representations, whereas the second review provides a general insight into colour models in computer graphics through the detailed examinations of its concepts and terminology. The work that we have contributed in the proposed visualisation of optical realism and the algebraic framework of volume deformation is described in the rest of the section.

This part of the thesis has been presented in the following publication:

- the Eurographics 2005 conference under the title *Spectral volume rendering based on the Kubelka-Munk theory* [AC05], in Dublin, Ireland.

8.1.1 A New Approach to Direct Volume Rendering

We have developed a new approach to direct volume rendering using a spectral volume rendering integral based on the Kubelka-Munk theory of diffuse reflectance. In the new approach, it has been shown that the Kubelka-Munk theory facilitates a correct spectral volume rendering integral that is suitable for both solid objects and amorphous matter in volume data sets. We have also shown that a volume rendering integral based on the Kubelka-Munk theory can provide volume visualisation with a higher level of accuracy in optical effects than the traditional volume rendering integral based on RGBα accumulation. Such high levels of accuracy can be seen in the comparisons between the two volume rendering integrals in the features of *general* and *selective absorptions* [JW76], which are simulated by the traditional volume rendering integral and the volume rendering integral based on the Kubelka-Munk theory, respectively.

A comparison between the spectral volume rendering integral based on the Kubelka-Munk theory and the previous spectral volume rendering integrals proposed by Noordmans *et al.* [NvS00] and Bergner *et al.* [BMD02, BMDF02, BMD04, BMTD05] has also been performed. The comparison examines the spectral volume rendering integrals in the form of its absorption and scattering properties together with the effects of the changing thickness of a medium. The results generated by the spectral volume rendering integral based on the Kubelka-Munk theory demonstrates depth-cueing in an optically correct manner in comparison with two existing rendering integrals, which have difficulties in relating opacity to scattering along the direction of S.

The volume rendering integral based on the Kubelka-Munk theory has provided us with the capability of separating the illumination process from the volume rendering integral. This has led to the consideration of utilising post-illumination to interrogate volume data sets with various light sources. Such a consideration has been demonstrated with a system called iPi (*interactive post-illumination*). We have also examined an approach for the design of spectral transfer functions for specifying absorption and scattering coefficients. We have demonstrated the optical realism in volume visualisation that can be achieved by the direct volume rendering based on the Kubelka-Munk with a combination of several natural and artificial colour data sets.

8.1.2 Extensive Study on Algebraic Properties of Volume-based Modelling and Physically-based Colour Mixing Techniques

We have conducted a comprehensive study on the algebraic framework of *Constructive Volume Geometry* (CVG). In the course of the study, we have generated a detailed analysis of the algebraic properties of the two existing example classes that can be found in CVG. The study of the algebraic properties revealed a number of undesired features in one of the example classes, such as the non-commutativity of the union and intersection operations. The non-commutativity of the operations allows us to consider and implement a collection of new operations for the example class, which was examined both algebraically and visually. In comparison with the previously defined union and intersection operations, the new operations fulfil the law of commutativity and therefore helps maintain the predictability and consistency in performing a series of operations on volume objects.

In addition to the new operations, we have also extended the algebraic framework of CVG to include a more complex example class of volume objects where its scalar fields can be used to defined physical properties of the object, such as absorption and scattering. This complex example class of volume object is desirable as it exhibits better algebraic properties in comparison with the two existing example classes.

We have also conducted a detailed examination of the different types of colour mixing algorithms. In the examination, we have revealed several alternative colour mixing algorithms that can be implemented in a graphics or visualisation system, such as the Bouguer-Beer's law of colour mixing and the Kubelka-Munk theory of colour mixing. Such forms of colour mixing algorithm are desirable as they provide us with the capability of utilising the physical properties of the objects, such as scattering and absorption, in comparison to the colour mixing based on the RGB colour space.

8.1.3 Algebraic Properties of Volume Deformation

We have proposed an algebraic framework for modelling constructive deformation in a constructive manner. The design of such an algebraic framework would not only enable deformation to become an integral part of a volume scene graph, but also it would facilitate constructive specification, inter-operation and direct rendering of volume deformation. By building on existing work of displacement mapping and spatial transfer functions, we have introduced the concept of *spatial displacement function* (SDF), which has been shown to be algebraically more sophisticated that its predecessors – in particular, spatial transfer functions. The algebraic superiority can be observed in the form of its identity and inverse properties.

In this algebraic framework, surface displacement mapping techniques have been successfully transformed to a volume technique that can be used to model complex volume deformation effects, such as opening, peeling and folding. From the perspective of volume deformation with SDFs, specifications of deformation can be modelled, combined and manipulated as conventional volume objects. We have also presented algorithms that allow direct rendering of volume scene graphs involving displacement objects.

The work on the algebraic properties of volume deformation is in preparation to be submitted to a journal.

8.1.4 Improvements on Colour-to-Spectrum Algorithms

We have conducted a comprehensive investigation into the colour-to-spectrum algorithms with the focus on the combination of linearly independent basis functions of *Fourier* [Gla89] and *Gaussian* [SFCD99]. In the investigation, we have highlighted some of the problems that may occur in the colour-to-spectrum algorithms. One such problem is the negative energy that can occur in its reconstructed spectrum, which contradicts the physical laws of nature. In our attempt to tackle the problem, we have examined and implemented some modifications that can be made to the existing conversion algorithms. The modifications are able to assist in the reduction of the amount of negative energy generated in the conversion algorithms. In comparison with the existing conversion algorithms, the amount of negative energy present in the reconstructed spectra using the modified algorithms is less than the ones in the existing algorithms.

8.2 Future Work

In this thesis, we have demonstrated physically-based rendering and algebraic manipulations of volume models with the focus on optical realism and volume deformation. In the course of the thesis, we have shown that it is possible to generate an image that displays a high level of optical realism. We have considered an approach to the design of spectral transfer functions, however, the design process of spectral transfer functions and the control of visual effects with a colour spectrum is a complex procedure. Moreover, the spectral colour space is a significantly large space. It is therefore possible to build on the work presented here by presenting an intuitive means of designing spectral transfer functions.

The volume deformation work suggests the great potential in our algebraic framework of volume deformation. In particular, volume scene graphs with deformation capability can be used as a technical framework at the lower layer of software for computer generated illustration. Therefore it is possible to integrate other advanced visualisation techniques into this framework.

With the work presented in this thesis, we hope that future researchers are further encouraged to achieve better realism and perception through optical correctness by the experimentation of complex illumination models.

Bibliography

[AC05] A. Abdul-Rahman and M. Chen. Spectral volume rendering based on the Kubelka-Munk theory. *Computer Graphics Forum*, 24(3):413 – 422, 2005.

[Ago76] M. K. Agoston, editor. *Algebraic Topology*. Marcel Dekker, New York, 1976.

[All80] E. Allen. Colorant formulation and shading. In F. Grum and C. J. Bartleson, editors, *Optical Radiation: Color Measurement*, volume 2, New York, 1980. Academic Press.

[ASK94] R. S. Avila, L. M. Sobierajski, and A. E. Kaufman. Visualizing nerve cells. *IEEE Computer Graphics and Applications*, pages 11 – 13, September 1994.

[Bar84] A. Barr. Global and local deformation of solid primitives. *Computer Graphics (Proc. SIGGRAPH 84)*, 18(3):21 – 30, July 1984.

[Bar86] A. Barr. Ray tracing deformed surfaces. *Computer Graphics (Proc. SIGGRAPH 86)*, 20(4):287 – 296, July 1986.

[Bar93] M. F. Barnsley, editor. *Fractals Everywhere*. Academic Press, Boston, 2nd edition, 1993.

[Bar00] K. Barnard. *Practical Colour Constancy*. PhD thesis, Simon Fraser University, 2000.

[Bau72] B. G. Baumgart. Winged-edge polyhedron representation. Technical Report STAN-CS-320, Stanford University, 1972.

[Bau75] B. G. Baumgart. A polyhedron representation for computer vision. In *Proc. National Computer Conference*, volume NCC 75, pages 589 – 596, 1975.

[BC96] M. Bro-Nielsen and S. Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. *Computer Graphics Forum*, 15(3):57–66, 1996.

[Bee52] A. Beer. Bestimmung der Absorption des rothen Lichts in farbigen Flüssigkeiten. *Ann. Phys. Chem.*, 86(2):78 – 90, 1852.

[Bee54] A. Beer. *Grundriß des photometrischen Kalküls*. Branschweig, 1854.

[Ber00] R. S. Berns. *Billmeyer and Saltzman's Principles of Color Technology*. Wiley-Interscience Publication, 3rd edition, 2000.

[BF94] D. H. Brainard and W. T. Freeman. Bayesian method for recovering surface and illuminant properties for photosensor responses. In *Proc. of IS & T and SPIE Symposium on Electronic Imaging Science & Technology*, volume 2179, pages 364 – 376, 1994.

[BFHT98] M. H. Brill, G. D. Finlayson, P. M. Hubel, and W. Thornton. Prime wavelengths and color imaging. In *Proc. 6th Color Imaging Conference: Color, Science, Systems and Applications*, pages 33 – 41, 1998.

[Bha93] N. Bhate. Application of rapid hierarchical radiosity to participating media. In B. Özgüc and V. Akman, editors, *ATARV-93: Advanced Techniques in Animation, Rendering and Visualization*, pages 43 – 53. Bilkent University, 1993.

[BHP46] F. Bloch, W. W. Hansen, and M. Packard. The nuclear induction experiment. *Physical Review*, 70:474 – 485, 1946.

[Bil63] F. W. Billmeyer, Jr. An objective approach to coloring. *Farbe*, 12:151 – 164, 1963.

[Bin71] T. Binford. Visual perception by computer. In *Proc. IEEE Conference on Systems and Control*, 1971.

[BK04] G. V. G. Baranoski and A. Krishnaswamy. An introduction to light interaction with human skin. *Revista de Informatica Teorica e Aplicada*, 11(1):33 – 62, 2004.

[BK05] G. V. G. Baranoski and A. Krishnaswamy. Simulation of light interaction with human skin. In *Tutorial Eurographics 2005*, 2005.

[Bli78] J. F. Blinn. Simulation of wrinkled surfaces. *Computer Graphics (Proc. SIGGRAPH 78)*, 12(3):286 – 292, 1978.

[Bli82a] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235 – 256, 1982.

[Bli82b] J. F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. *Computer Graphics*, 16(3):21 – 29, 1982.

[BLL96] R. Bajcsy, S. W. Lee, and A. Leonardis. Detection of diffuse and specular interface reflections and inter-reflections by color image segmentation. *International Journal of Computer Vision*, pages 241 – 271, 1996.

[Blo47] F. Blotiau. Les défauts d'additivitè de la colorimètric trichromatique. *Rev. d'Opt*, 23:193, 1947.

[Blo97] J. Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan Kaufmann, San Francisco, 1997. With C. Bajaj, J. Blinn, M. P. Cani-Gascuel, A. Rockwood, B. Wyvill and G. Wyvill.

[BM00] D. Bartz and M. Meißner. Voxels versus polygons: A comparative approach for volume graphics. In M. Chen, A. Kaufman, and R. Yagel, editors, *Volume Graphics*, pages 171 – 184, London, 2000. Springer-Verlag.

[BMD02] S. Bergner, T. Möller, and M. S. Drew. Spectral volume rendering. Technical Report SFU-CMPT-03/02-TR2002-03, School of Computing Science, Simon Fraser University, 2002.

[BMD04] S. Bergner, T. Möller, and M. S. Drew. A spectral engine for visualization. Technical Report SFU-CMPT-TR2004-06, School of Computing Science, Simon Fraser University, 2004.

[BMDF02] S. Bergner, T. Möller, M. S. Drew, and G. D. Finlayson. Interactive spectral volume rendering. In *Proc. IEEE Visualization*, pages 101 – 108, 2002.

[BMTD05] S. Bergner, T. Möller, M. Tory, and M. S. Drew. A practical approach to spectral volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 11(2):207 – 216, 2005.

[Bou29] P. Bouguer. *Essai d'optique sur la gradation de la lumiére*. Claude Jombert, Paris, 1729.

[Bou47] P. J. Bouma. *Physical aspects of colour: An introduction to the scientific study of colour stimuli and colour sensations*. Eindhoven Philips Gloeilampenfabrieken, 1947. Translated from Dutch by A. Wolters-Dawson.

[Bro98] M. Bro-Nielsen. Finite element modeling in medical VR. *Journal of IEEE*, 8(3):490 – 503, 1998.

[BW99] M. Born and E. Wolf. *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*. Cambridge University Press, 1999.

[BW01] K. Brodlie and J. Wood. Recent advances in volume visualization. *Computer Graphics Forum*, 20(2):125 – 148, 2001.

[BWL04] W. Baxter, J. Wendt, and M. C. Lin. IMPaSTo: a realistic, interactive model for paint. In *Proc. 3rd International Symposium on Non-photorealistic Animation and Rendering*, pages 45 – 148, 2004.

[CAS$^+$91] C. A. Curcio, K. A. Allen, K. R. Sloan, C. L. Lerea, J. B. Hurley, I. B. Klock, and A. H. Milam. Distribution and morphology of human cone photoreceptors stained with anti-blue opsin. *Journal of Comparative Neurology*, 312:610 – 624, 1991.

[CAS$^+$97] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin. Computer-generated watercolor. In *Proc. 24th Annual ACM SIGGRAPH Conference*, pages 421 – 430, 1997.

[CCC87] R. L. Cook, L. Carpenter, and E. Catmull. The Reyes image rendering architecture. *Computer Graphics (Proc. SIGGRAPH 87)*, 21(4):95 – 102, 1987.

[CF01] R. J. Campbell and P. J. Flynn. A survey of free-form object representation and recognition techniques. *Computer Vision and Image Understanding*, 81:166 – 210, 2001.

[CGMS00] P. Cignoni, F. Ganovelli, C. Montani, and R. Scopigno. Reconstruction of topologically correct and adaptive trilinear isosurfaces. *Computer & Graphics*, 24:399 – 418, 2000.

[Che95] E. V. Chernyaev. Marching cubes 33: Construction of topologically correct isosurfaces. Technical Report CN/95-17, Institute of High Energy Physics, 1995. Available as `http://wwwinfo.cern.ch/asdoc/psdir/mc.ps.gz`.

[Che01] M. Chen. Volume graphics. In A. Kent and J. G. Williams, editors, *Encyclopedia of Microcomputers*, pages 363 – 387, New York, 2001. Marcel Dekker.

[CIJ+05] M. Chen, S. Islam, M. W. Jones, P.-Y. Shen, D. Silver, S. J. Walton, and P. J. Willis. Deforming and animating discretely sampled object representations. In *Eurographics State of the Art Reports*, pages 113 – 140, Dublin, Ireland, 2005.

[CJT96] M. Chen, M. W. Jones, and P. Townsend. Volume distortion and morphing using disk fields. *Computers and Graphics*, 20(4):567 – 575, 1996.

[CM93] R. A. Crawfis and N. Max. Texture splats for 3D scalar and vector field visualization. In *Proc. IEEE Visualization*, pages 261 – 267, 1993.

[Coo84] R. L. Cook. Shade trees. *Computer Graphics*, 18:223 – 231, 1984.

[Cor63] A. M. Cormack. Representation of a function by its integral, with some radiological applications. *Journal of Applied Physics*, 34:2722 – 2727, 1963.

[Cor64] A. M. Cormack. Representation of a function by its integral, with some radiological applications ii. *Journal of Applied Physics*, 35:2908 – 2913, 1964.

[CSW+03] M. Chen, D. Silver, A. S. Winter, V. Singh, and N. Cornea. Spatial transfer functions – A unified approach to specifying deformation in volume modeling and animation. In *Proc. Volume Graphics 2003*, pages 35 – 44, Tokyo, Japan, 2003. Eurographics/ACM Publications.

[CT82] R. L. Cook and K. E. Torrance. A reflection model for computer graphics. *ACM Transactions on Graphics*, 1:7 – 24, 1982.

[CT98] M. Chen and J. V. Tucker. Constructive volume geometry. Technical Report CS-TR-98-19, University of Wales Swansea, July 1998.

[CT00] M. Chen and J. V. Tucker. Constructive volume geometry. *Computer Graphics Forum*, 19(4):281 – 293, 2000.

[CZK98] Y. Chen, Q. Zhu, and A. Kaufman. Physically-based animation of volumetric objects. In *Proc. IEEE Computer Animation '98*, pages 154 – 160, 1998.

[DCH88] R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *ACM SIGGRAPH Computer Graphics*, 22(4):65 – 74, 1988.

[DF92] M. S. Drew and B. V. Funt. Natural metamers. *Computer Vision Graphics and Image Processing: Image Understanding*, 56(2):139 – 151, 1992.

[DH96] J. Dorsey and P. Hanrahan. Modeling and rendering of metallic patinas. In *Proc. 23rd Annual ACM SIGGRAPH Conference*, pages 387 – 396, 1996.

[DH00] M. Doggett and J. Hirche. Adaptive view dependent tessellation of displacement maps. In *Proc. EG/SIGGRAPH Workshop on Graphics Hardware*, pages 59 – 66, Interlaken, Switzerland, 2000.

[DI93] M. M. D'Zmura and G. Iverson. Color constancy i: Basic theory of two-stage linear recovery of spectral descriptions for lights and surfaces. *Journal of the Optical Society of America A*, 10(10):2148 – 2165, 1993.

[DI94] M. M. D'Zmura and G. Iverson. Probabilistic color constancy. In M. M. D'Zmura, D. Hoffman, G. Iverson, and K. Romney, editors, *Geometric Representations of Perceptual Phenomena: Papers in Honor of Tarow Indow's 70th Birthday*. Laurence Erlbaum Associates, 1994.

[Die00] S. Dietrich. Elevation maps. Technical Report White Paper, NVIDIA Corporation, 2000.

[DJ05] C. Donner and H. W. Jensen. Light diffusion in multi-layered translucent materials. In *Proc. 24th Annual ACM SIGGRAPH Conference*, pages 1032 – 1039, 2005.

[DK96] S. Douglas and T. Kirkpatrick. Do color models really make a difference? In *Proc. SIGCHI Conference on Human Factors in Computing Systems*, pages 399 – 405, 1996.

[DNvK99] K. J. Dana, S. K. Nayar, B. van Ginneken, and J. J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1 – 34, 1999.

[Duf92] T. Duff. Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry. *ACM SIGGRAPH Computer Graphics*, 26(2):131 – 138, 1992.

[Dul06] Dulux. Dulux, 2006. `http://www.dulux.co.uk/`.

[Dun49] D. R. Duncan. The colour of pigment mixtures. *The Journal of Oil and Colour Chemists Association*, 32(7):296 – 321, 1949.

[Dür88] M. J. Dürst. Additional reference to "marching cubes". *Computer Graphics*, 22(5):243, April 1988. Letter.

[EA66] R. R. Ernst and W. A. Anderson. Application of Fourier transform spectroscopy to magnetic resonance. *Reviews of Scientific Instruments*, 37:93 – 102, 1966.

[Eur06] Eurographics. Eurographics digital library, 2006. `http://www.eg.org/EG/DL`.

[FFC82] A. Fournier, D. Fussell, and L. Carpenter. Computer rendering of stochastic models. *Communications of the ACM*, 25(6):371 – 384, 1982.

[FH00] G. D. Finlayson and S. Hordley. Improving gamut mapping color constancy. *IEEE Transactions on Image Processing*, 9(10):1774 – 1783, October 2000.

[FHH01] G. D. Finlayson, S. D. Hordley, and P. M. Hubel. Color by correlation: A simple, unifying framework for color constancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1209 – 1221, 2001.

[Fin95] G. Finlayson. *Coefficient Colour Constancy.* PhD thesis, Simon Fraser University, 1995.

[Fin96] G. D. Finlayson. Color in perspective. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):1034 – 1038, 1996.

[Fis83] K. P. Fishkin. Applying color science to computer graphics. Master's thesis, University of California, Berkeley, December 1983.

[FL01] R. Fattal and D. Lischinski. Variational classification for visualization of 3D ultrasound data. In *Proc. IEEE Visualization*, pages 403 – 410, 2001.

[FM99] G. D. Finlayson and P. M. Morovic. Metamer constrained colour correction. *The 7th Color Imaging Conference*, pages 26 – 31, November 1999.

[FM00] G. D. Finlayson and P. M. Morovic. Metamer crossovers of infinite metamer sets. *The 8th Color Imaging Conference*, pages 13 – 17, November 2000.

[For57] G. E. Forsythe. Generation and use of orthogonal polynomials for data fitting with a digital computer. *Journal of the Society for Industrial and Applied Mathematics*, 5:74 – 88, 1957.

[For88a] D. A. Forsyth. *Colour Constancy and its Application in Machine Vision.* PhD thesis, Oxford University, 1988.

[For88b] D. A. Forsyth. A novel approach to colour constancy. *IEEE International Conference on Computer Vision*, pages 9 – 17, 1988.

[Fra05] Fraunhofer IGD. Teleinvivo home page, 2005. `http://a7www.igd.fhg.de/images-video/teleinvivo/teleinvivo.html`.

[FS98] S. Fang and R. Srinivasan. Volumetric-CSG – A model-based volume visualization approach. In *Proc. Sixth International Conference in Central Europe on Computer Graphics and Visualization*, pages 88 – 95, 1998.

[FvDFH95] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics Principles and Practice.* Addison-Wesley, 2nd edition, 1995.

[GFWS96] W. F. Garrett, H. Fuchs, M. C. Whitton, and A. State. Real-time incremental visualization of dynamic ultrasound volumes using parallel bsp trees. In *Proc. IEEE Visualization*, pages 235 – 240, 1996.

[GH99] S. Gumhold and T. Hüttner. Multiresolution rendering with displacement mapping. In *Proc. EG/SIGGRAPH Workshop on Graphics Hardware*, pages 55 – 66, Los Angeles, CA, 1999.

[GHJ96] R. Geist, O. Heim, and S. Junkins. Color representation in virtual environments. *Color Research and Application*, 21:121 – 128, 1996.

[Gib97] S. Gibson. 3D chainmail: A fast algorithm for deforming volumetric objects. In *Proc. 1997 Symposium on Interactive 3D Graphics*, pages 149 – 154, April 1997.

[Gla89] A. S. Glassner. How to derive a spectrum from an RGB triplet. *IEEE Computer Graphics and Applications*, pages 95 – 99, 1989.

[Gla95] A. S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann, San Francisco, 1995.

[Gle88] J. Gleick, editor. *Chaos Making a New Science*. R. R. Donnelley & Sons Company, Harrisonburg, Virginia, 1988.

[GM97] S. F. F. Gibson and B. Mirtich. A survey of deformable modeling in computer graphics. Technical Report TR97-19, MERL Technical Report, 1997.

[GMN94] J. S. Gondek, G. W. Meyer, and J. G. Newman. Wavelength dependent reflectance functions. In *Proc. 21st Annual ACM SIGGRAPH Conference*, pages 213 – 220, 1994.

[GNHR98] A. Garcia-Beltran, J. L. Nieves, J. Hernandez-Andres, and J. Romero. Linear bases for spectral reflectance functions of acrylic paints. *Color Research and Application*, 23:39 – 45, 1998.

[Gou71] H. Gouraud. Computer display of curved surfaces. *IEEE Transactions on Computers*, C-20(6):623 – 629, 1971.

[Gra54] H. Grassmann. On the theory of compound colors. *Philosophical Magazine*, 7(4):254 – 264, 1854.

[GS99] N. Gagvani and D. Silver. Parameter controlled volume thinning. *Graphical Models and Image Processing*, 61(3):149 – 164, 1999.

[GS00] N. Gagvani and D. Silver. Animating the visible human dataset. In *Proc. the Visible Human Project Conference (2000)*, 2000.

[GS01] N. Gagvani and D. Silver. Animating volumetric models. *Graphical Models*, 63(6):443 – 458, 2001.

[GS04] S. Guy and C. Soler. Graphics gems revisited: Fast and physically-based rendering of gemstones. *ACM Transactions on Graphics*, 23(3):231 – 238, 2004.

[Gui31] J. Guild. The colorimetric properties of the spectrum. *Philosophical Transactions of the Royal Society of London, Series A*, 230:149 – 187, 1931.

[GW92] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, Reading, Massachusetts, 1992.

[Hal89] R. Hall. *Illumination and Color in Computer Generated Imagery*. Springer-Verlag, 1989.

[Har05] J. L. Harvey. Institute of non-newtonian fluid mechanics, 2005. `http://innfm.swan.ac.uk/innfm\_updated/index2.html`.

[HBS99] C. H. Ho, C. Basdogan, and M. A. Srinivasan. Efficient point-based rendering techniques for haptic display of virtual objects. *Presence*, 8(5):477 – 491, 1999.

[HDKS00] W. Heidrich, K. Daubert, K. Kautz, and H.-P. Seidel. Illuminating micro geometry based on precomputed visibility. In *Computer Graphics (Proc. SIGGRAPH 2000)*, pages 455 – 464. ACM Press, 2000.

[Her78] E. Hering. *Zur Lehre vom Lichtsinne.* Gerold & Sohn, 1878. Translated from German by L. M. Hurvich and D. Jameson, Outlines of a Theory of the Light Sense, Harvard University Press, Cambridge, MA, 1964.

[Her80] G. T. Herman, editor. *Image Reconstruction from Projections: The Fundamentals of Computerized Tomography.* Academic Press, 1980.

[HG83] R. A. Hall and D. P. Greenberg. A testbed for realistic image synthesis. *IEEE Computer Graphics and Applications*, pages 10 – 20, November 1983.

[Hil97] A. R. Hill. How we see colour. In R. McDonald, editor, *Colour Physics for Industry*, pages 426 – 513, Bradford, 1997. Society of Dyers and Colourists.

[HM92] C. S. Haase and G. W. Meyer. Modeling pigmented materials for realistic image synthesis. *ACM Transactions on Graphics*, 11(4):305 – 335, 1992.

[HNN98] S. A. Hagstrom, J. Neitz, and M. Neitz. Variations in cone populations for red-green color vision examined by mRNA. *Neuroreport*, 9:1963 – 1967, 1998.

[Hof92] C. M. Hoffmann, editor. *Geometric and Solid Modeling.* Morgan Kaufmann, San Mateo, CA, 1992.

[Hor84] B. K. P. Horn. Exact reproduction of colored images. *Computer Vision Graphics and Image Processing: Image Understanding*, 26:135 – 167, 1984.

[Hor05] J. P. Hornak. The basics of MRI, 2005. `http://astro.rit.edu/htbooks/mri/`.

[Hou72] G. Hounsfield. A method of an apparatus for examination of a body by radiation such as X-ray or gamma radiation, 1972. Patent Specification 1283915.

[Hou73] G. Hounsfield. Computerized transverse axial scanning (tomography): Part i description of system. *British Journal of Radiology*, 46(552):1016 – 1022, 1973.

[Hun98] R. W. G. Hunt. *Measuring colour.* Kingston-upon-Thames, 3rd edition, 1998.

[HZ74] E. Hecht and A. Zajac. *Optics.* Addison-Wesley, Reading, Massachusetts, 1974.

[IDSC04] S. Islam, S. Dipankar, D. Silver, and M. Chen. Temporal and spatial splitting of scalar fields in volume graphics. In *Proc. IEEE VolVis2004*, pages 87 – 94, Austin, Texas, October 2004. IEEE.

[IEE06] IEEE. IEEE Xplore, 2006. `http://ieeexplore.ieee.org/Xplore/dynhome.jsp`.

[JF99] G. M. Johnson and M. D. Fairchild. Full-spectral color calculations in realistic image synthesis. *IEEE Computer Graphics and Applications*, 19(4):47 – 53, 1999.

[JG78] G. H. Joblove and D. Greenberg. Color spaces for computer graphics. In *Proc. 5th Annual ACM SIGGRAPH Conference*, pages 20 – 25, 1978.

[Jon95] M. W. Jones. *The Visualisation of Regular Three Dimensional Volume Data*. PhD thesis, University of Wales, Swansea, July 1995.

[Jon99] M. W. Jones. International workshop on volume graphics, 1999. `http://www.swan.ac.uk/compsci/vg99/`.

[JW76] F. A. Jenkins and H. E. White. *Fundamentals of Optics*. McGraw-Hill, 4th edition, 1976.

[Kau00] A. E. Kaufman. State-of-the-art in volume graphics. In M. Chen, A. Kaufman, and R. Yagel, editors, *Volume Graphics*, pages 3 – 28, London, 2000. Springer-Verlag.

[KB96] P. K. Kaiser and R. M. Boynton. *Human Color Vision*. Optical Society of America, 1996.

[KCY93] A. Kaufman, D. Cohen, and R. Yagel. Volume graphics. *IEEE Computer*, 26(7):51 – 64, 1993.

[Kel97] P. A. Keller. *Electronic Color Displays, Measurements, Concepts, Techniques and Instrumentation*. John Wiley & Sons, 1997.

[Kli93] G. J. Klinker. *A Physical Approach to Color Image Understanding*. A K Peters, Massachusetts, 1993.

[KM31] P. Kubelka and F. Munk. Ein Beitrag zur Optik der Farbanstriche. *Zeitschrift für Technische Physik*, 12:593 – 601, 1931.

[KMH$^+$04] R. Keiser, M. Müller, B. Heidelberger, M. Teschner, and M. Gross. Contact handling for deformable point-based objects. In *Proc. Vision, Modeling and Visualisation (VMV)*, pages 315 – 322, November 2004.

[KONN90] K. Kaneda, T. Okamoto, E. Nakamae, and T. Nishita. Highly realistic visual simulation of outdoor scenes under various atmospheric conditions. In *Proc. CG International 1990*, pages 117 – 131, 1990.

[KPHE02] J. Kniss, S. Premože, C. Hansen, and D. Ebert. Interactive translucent volume rendering and procedural modeling. In *Proc. IEEE Visualization*, pages 109 – 116, 2002.

[KS01] J. Kautz and H.-P. Seidel. Hardware accelerated displacement mapping for image based rendering. In *Proc. Graphics Interface*, pages 61 – 70, 2001.

[Kub48] P. Kubelka. New contributions to the optics of intensely light-scattering materials, part i. *Journal of the Optical Society of America*, 38:448 – 457, 1948.

[Kub54] P. Kubelka. New contributions to the optics of intensely light-scattering materials, part ii. nonhomogeneous layers. *Journal of the Optical Society of America*, 44:330 – 355, 1954.

[KV84] J. T. Kajiya and B. P. Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH Computer Graphics*, 18(3):165 – 174, 1984.

[KWE75] A. Kumar, D. Welti, and R. R. Ernst. NMR Fourier Zeugmatography. *Journal of Magnetic Resonance*, 18:69 – 83, 1975.

[KY95] Y. Kurzion and R. Yagel. Space deformation using ray deflectors. In *Proc. 6th EuroGraphics Workshop on Rendering 1995*, pages 21 – 32, Dublin, Ireland, June 1995. Eurographics.

[Lam60] J. H. Lambert. *Photometria sive de mensure et gradibus luminis colorum met umbrae*. Eberhard Klett, Augsburg, 1760. Latin: Photometry or on the Measurements and Grades (Degrees) of the Light of the Colors and the Shadow.

[Lan77] E. H. Land. The retinex theory of color vision. *Scientific American*, pages 108 – 129, 1977.

[LB03] A. Lopes and K. Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):16 – 29, 2003.

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

[Lee96] B. B. Lee. Receptive field structure in the primate retina. *Vision Research*, 36:631 – 644, 1996.

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

[Lev90] M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9(3):245 – 261, 1990.

[Lew26] G. N. Lewis. The conservation of photons. *Nature*, 2891(118):874 – 875, December 1926.

[LGL95] A. Lerios, C. Garfinkle, and M. Levoy. Feature-based volume metamorphosis. In *Computer Graphics (Proc. SIGGRAPH 95)*, pages 449 – 456. Addison-Wesley, Los Angeles, California, 1995.

[LH91] D. Laur and P. Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. *ACM SIGGRAPH Computer Graphics*, 25:285 – 288, 1991.

[LKHW03] A. E. Lefohn, J. M. Kniss, C. D. Hansen, and R. T. Whitaker. Interactive deformation and visualization of level set surfaces using graphics hardware. In *Proc. IEEE Visualization 2003*, pages 75 – 82. Seattle, WA, 2003.

[LM05a] S. Li and K. Mueller. Accelerated high-quality refraction computations for volume graphics. In *Volume Graphics*, pages 73 – 81, Stony Brook, 2005.

[LM05b] S. Li and K. Mueller. Spline-based gradient filters for high-quality refraction computations in discrete datasets. In *Proc. EUROGRAPHICS-IEEE VGTC Symposium on Visualization*, pages 215 – 222, 2005.

[LMH00] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *Computer Graphics (Proc. SIGGRAPH 2000)*, pages 85 – 94. ACM Press, 2000.

[Lop99] A. M. Lopes. *Accuracy in Scientific Visualization.* PhD thesis, University of Leeds, 1999.

[LTCK03] A. Liu, F. Tendick, K. Cleary, and C. Kaufmann. A survey of surgical simulation: applications, technology and education. *Presence*, 12(6), December 2003.

[LW90] M. Levoy and R. Whitaker. Gaze-directed volume rendering. In *Computer Graphics (Symposium on Interactive 3D Graphics)*, volume 24, pages 217 – 223, 1990.

[LW98] R. Lemos and B. Wyvill. A survey of layered construction techniques using deformable models in the animation of articulated figures. Technical Report TR1998-637-28, University of Calgary, 1998.

[Mac99] L. MacDonald. Using color effectively in computer graphics. *IEEE Computer Graphics and Applications*, pages 20 – 35, July/August 1999.

[Mal93] T. Malzbender. Fourier volume rendering. *ACM Transactions on Graphics*, 12(3):233 – 250, 1993.

[Man82] B. B. Mandelbrot, editor. *The Fractal Geometry of Nature.* W. H. Freeman and Company, San Francisco, 1982.

[Man97] T. Mandel, editor. *The Elements of User Interface Design.* John Wiley & Sons, New York, 1997.

[Max86] N. Max. Light difussion through clouds and haze. *Computer Vision, Graphics and Image Processing*, 33:280 – 292, 1986.

[Max88] N. Max. Horizon mapping: shadows for bump-mapped surfaces. *The Visual Computer*, 4(2):109 – 117, 1988.

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

[MB04] M. Mohammadi and R. S. Berns. Verification of the Kubelka-Munk turbid media theory for artist acrylic paint. Technical report, Spectral Color Imaging Laboratory Group, Munsell Color Science Laboratory, Rochester Institute of Technology, 2004.

[MC06] N. Max and M. Chen. Local and global illumination in the volume rendering integral. *Draft report (under review)*, 2006.

[McA04] A. McAndrew, editor. *An introduction to digital image processing with MATLAB*. Brooks Cole, 2004.

[MDM+02] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 49 – 54, San Antonio, Texas, 2002. ACM Press. 1-58113-573-4.

[Mea82] D. Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129 – 147, June 1982.

[Mey88] G. W. Meyer. Wavelength selection for synthetic image generation. *Computer Vision, Graphics and Image Processing*, 41:57 – 79, 1988.

[MG80] G. W. Meyer and D. P. Greenberg. Perceptual color spaces for computer graphics. In *Proc. 7th Annual ACM SIGGRAPH Conference*, pages 254 – 261, 1980.

[MGW01] T. Malzbender, D. Gelb, and H. Wolters. Polynormal texture maps. In *Computer Graphics (Proc. SIGGRAPH 2001)*, pages 519 – 528. ACM Press, 2001.

[MHC90] N. Max, P. Hanrahan, and R. Crawfis. Area and volume coherence for efficient visualization of 3D scalar functions. *Computer Graphics*, 24(5):27 – 33, 1990.

[MKN+04] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Cross, and M. Alexa. Point based animation of elastic, plastic and melting objects. In *Proc. ACM SIGGRAPH Symposium on Computer Animation*, pages 141 – 151, 2004.

[MMC99] K. Mueller, T. Möller, and R. Crawfis. Splatting without the blur. In *Proceedings of the conference on Visualization '99*, pages 363 – 370, 1999.

[MMD76] C. S. McCamy, H. Marcus, and J. G. Davidson. A color rendition chart. *Journal of Applied Photographic Engineering*, 2:95 – 99, 1976.

[MMS+91] H. P. Meinzer, K. Meetz, D. Scheppelmann, U. Engelmann, and H. J. Baur. The Heidelberg ray tracing model. *IEEE Computer Graphics and Applications*, 11(6):34 – 43, 1991.

[Moo45] P. Moon. Polynomial representation of reflectance curves. *Journal of the Optical Society of America*, 35:597 – 600, 1945.

[MS74] L. March and P. Steadman, editors. *The Geometry of Environment*. MIT Press, Cambridge, MA, 1974.

[MT92] K. Meinke and J. V. Tucker. Universal algebra. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume I, pages 189 – 411. Oxford University Press, 1992.

[MT96] T. McInerney and D. Terzopoulos. Deformable models in medical image analysis: A survey. *Medical Image Analysis*, 1(2):91 – 108, 1996.

[MTB03] M. J. McGuffin, L. Tancau, and R. Balakrishnan. Using deformations for browsing volumetric data. In *Proc. IEEE Visualization 2003*, pages 401 – 408. Seattle, WA, 2003.

[Mun05] A. H. Munsell. *A Color Notation.* Munsell Color Company, 1st edition, 1905.

[Mun15] A. H. Munsell. *Atlas of the Munsell Color System.* Wadsworth-Howland & Company, 1915.

[MW80] G. Markowsky and M. A. Wesley. Fleshing out wire frames. *IBM Journal of Research and Development*, 24(5):582 – 597, 1980.

[MW86] L. T. Maloney and B. A. Wandell. Color constancy: A method for recovering surface spectral reflectance. *Journal of the Optical Society of America A*, 3(1):29 – 33, January 1986.

[MY96] K. Mueller and R. Yagel. Fast perspective volume rendering with splatting by utilizing a ray-driven approach. In *Proc. IEEE Visualization*, pages 65 – 72, 1996.

[Nad00] D. R. Nadeau. Volume scene graphs. In *Proc. 2000 IEEE Symposium on Volume Visualization*, pages 49 – 56, 2000.

[Nas01] K. Nassau. *The physics and chemistry of color: The fifteen causes of color.* John Wiley & Sons, 2nd edition, 2001.

[Nat94] B. Natarajan. On generating topologically consistent isosurfaces from uniform samples. *The Visual Computer*, 11:52 – 62, 1994.

[NH91] G. M. Nielson and B. Hamann. The asymptotic decider: Resolving the ambiguity in marching cubes. In *Proc. IEEE Visualization*, pages 83 – 90, 1991.

[NHK$^+$85] H. Nishimura, M. Hirai, T. Kawai, T. Kawata, I. Shirakawa, and K. Omura. Object modeling by distribution function and a method of image generation. *Japan Electronics Communication Conference 1985*, J68-D(4):718 – 725, 1985. In Japanese.

[Nie97] G. Nielson. Tools for triangulations and tetrahedralizations and constructing functions defined over them. In G. Nielson, H. Hagen, and H. Muller, editors, *Scientific Visualization: Overviews, Methodologies, Techniques*, pages 429 – 525, Los Alamitos, CA, 1997. IEEE Computer Society.

[NM01] M. Nulkar and K. Mueller. Splatting with shadows. In K. Mueller and A. E. Kaufman, editors, *Proc. Joint IEEE TCVG and Eurographics Workshop (VolumeGraphics-01)*, pages 35 – 50, 2001.

[NMK$^+$05] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. Physically based deformable models in computer graphics. In *Eurographics State of the Art Reports*, Dublin, Ireland, 2005.

[NMN87] T. Nishita, Y. Miyawaki, and E. Nakamae. A shading model for atmospheric scattering considering luminous intensity of light sources. *Computer Graphics*, 21(4):303 – 310, 1987.

[NN98] J. Noh and U. Newmann. A survey of facial modeling and animation techniques. Technical Report TR1998-99-705, Integrated Media Systems Center, University of Southern California, 1998.

[NNJ43] S. M. Newhall, D. Nickerson, and D. B. Judd. Final report to the O.S.A. subcommittee on the spacing of the Munsell chips. *Journal of the Optical Society of America*, 33:385 – 418, 1943.

[Nob97] J. H. Nobbs. Colour-match prediction for pigmented materials. In R. McDonald, editor, *Colour Physics for Industry*, 1997.

[NvS00] H. J. Noordmans, H. T. M van der Voort, and A. W. M. Smeulders. Spectral volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 6(3):196 – 207, 2000.

[OBM00] M. M. Oliveira, G. Bishop, and D. McAllister. Relief texture mapping. In *Computer Graphics (Proc. SIGGRAPH 2000)*, pages 359 – 368. ACM Press, 2000.

[Oht71] N. Ohta. The color gamut available by the combination of subtractive color dyes. i. Actual dyes in color film (1) Optimum peak wavelengths and breadths of cyan, magenta and yellow. *Photographic Science and Engineering*, 15:399 – 415, 1971.

[OW82] R. D. Overheim and D. L. Wagner. *Light and Color*. John Wiley & Sons, 1982.

[OW91] R. J. Oddy and P. J. Willis. A physically based colour model. *Computer Graphics Forum*, 10:121 – 127, 1991.

[Par01] R. Parent. *Computer Animation: Algorithms and Techniques*. Morgan-Kaufmann, 2001.

[Pas05] A. A. Pasko. Hyperfun project, 2005. `http://cis.k.hosei.ac.jp/~F-rep/HF_proj.html`.

[Pas06] D. Pascale. RGB coordinates of the MacBeth ColorChecker, 2006. `http://www.babelcolor.com/`.

[PASS01] A. Pasko, V. Adzhiev, B. Schmitt, and C. Schlick. Constructive hypervolume modeling. *Graphical Models*, 63(6):413 – 442, 2001.

[PD84] T. Porter and T. Duff. Compositing digital images. *ACM SIGGRAPH Computer Graphics*, 18(3):253 – 259, 1984.

[Pee93] M. S. Peercy. Linear color representations for full spectral rendering. *ACM SIGGRAPH Computer Graphics*, 27(3):191 – 198, 1993.

[PH96] M. Pharr and P. Hanrahan. Geometry caching for ray-tracing displacement maps. In *Proc. Eurographics Rendering Workshop*, pages 31 – 40, 1996.

[PHL95] J. W. Patterson, S. G. Hoggar, and J. R. Logie. Inverse displacement mapping. *Computer Graphics Forum*, 14(5):129 – 139, 1995.

[Pho75] B. T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311 – 317, 1975.

[PKKG03] M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, 22(3):641 – 650, 2003.

[PLH$^+$90] P. Prusinkiewicz, A. Lindenmayer, J. S. Hanan, F. D. Fracchia, D. R. Fowler, M. J. M. de Boer, and L. Mercer, editors. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

[Poy96] C. A. Poynton. *A Technical Introduction to Digital Video*. John Wiley & Sons, 1996.

[PPP88] A. A. Pasko, V. V. Pilyugin, and V. N. Pokrovskiy. Geometric modeling in the analysis of trivariate functions. *Computer & Graphics*, 12(3/4):457 – 465, 1988.

[PSS77] E. M. Patterson, C. E. Sheldon, and B. H. Stockton. Kubelka-munk optical properties of a barium sulfate white reflectance standard. *Applied Optics*, 16(3):729 – 732, March 1977.

[PT90] B. A. Payne and A. W. Toga. Surface mapping brain function on 3D models. *IEEE Computer Graphics and Applications*, pages 33 – 41, September 1990.

[PTP45] E. M. Purcell, H. C. Torrey, and R. V. Pound. Resonance absorption by nuclear magnetic moments in a solid. *Physical Review*, 69:37 – 38, 1945.

[PW84] A. Pentland and J. Williams. Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics (Proc. SIGGRAPH 89)*, 23(3):215 – 222, July 1984.

[RC01] D. Rodgman and M. Chen. Refraction in discrete raytracing. In A. Kaufman and K. Mueller, editors, *Proc. Volume Graphics*. Springer, New York, 2001.

[RC05] D. Rodgman and M. Chen. Regularised anisotropic nonlinear diffusion for rendering refraction in volume graphics. In E. Trucco and M. Chantler, editors, *Vision, Video and Graphics*, pages 219 – 228, 2005.

[RC06] D. Rodgman and M. Chen. Volume denoising for visualizing refraction. In G. P. Bonneau, G. M. Nelson, and T. Ertl, editors, *Scientific Visualization, Dagstuhl Scientific Visualization 2003*, pages 163 – 184. Springer, 2006.

[Ree83] W. T. Reeves. Particle systems - A technique for modeling a class of fuzzy objects. *ACM SIGGRAPH Computer Graphics*, 17(3):359 – 376, 1983.

[Req77] A. A. G. Requicha. Mathematical models of rigid solids. Technical Report Technical Memo 28, University of Rochester, 1977. Production Automation Project.

[Req80] A. A. G. Requicha. Representations for rigid solids: Theory, methods and systems. *ACM Computing Surveys*, 12(4):437 – 464, 1980.

[RF91] M. G. Raso and A. Fournier. A piecewise polynomial approach to shading using spectral distributions. In *Proc. Graphics Interface 91*, pages 40 – 46, 1991.

[Ric73] A. Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2):157 – 160, 1973.

[Rig97] B. Rigg. Colorimetry and the CIE system. In R. McDonald, editor, *Colour Physics for Industry*, pages 81 – 120, Bradford, 1997. Society of Dyers and Colourists.

[RMN03] D. Rudolf, D. Mould, and E. Neufeld. Simulating wax crayons. In *Proc. 11th Pacific Conference on Computer Graphics and Applications*, pages 163 – 172, 2003.

[RMN05] D. Rudolf, D. Mould, and E. Neufeld. A bidirectional deposition model of wax crayons. *Computer Graphics Forum*, 24(1):27 – 39, 2005.

[Roe04] S. Roettger. The volume library, 2004. `http://www9.cs.fau.de/Persons/Roettger/library`.

[RP98] G. Rougeron and B. Péroche. Color fidelity in computer graphics: A survey. *Computer Graphics Forum*, 17:3 – 15, 1998.

[RR78] D. Reddy and S. Rubin. Representation of three-dimensional objects. Technical Report CMU-CS-78-113, Carnegie-Mellon University, 1978. Computer Science Department.

[RSSG01] C. Rezk-Salama, M. Scheuering, G. Soza, and G. Greiner. Fast volumetric deformation on general purpose hardware. In *Proc. SIGGRAPH/Eurographics Graphics Hardware Workshop 2001*, pages 17 – 24. SIGGRAPH/Eurographics, 2001.

[RT87] H. Rushmeier and K. Torrance. The zonal method for calculating light intensities in the presence of a participating medium. *Computer Graphics*, 21(4):293 – 302, July 1987.

[RV83] A. A. G. Requicha and H. B. Voelcker. Solid modeling: Current status and research directions. *IEEE Computer Graphics and Applications*, 3:25 – 37, October 1983.

[Rva63] V. L. Rvačev. On the analytic description of certain geometric objects. *Reports of the Ukrainian Academy of Sciences*, 153:765 – 767, 1963.

[Sab88] P. Sabella. A rendering algorithm for visualizing 3D scalar fields. *ACM SIGGRAPH Computer Graphics*, 22(4):51 – 58, 1988.

[SACM06] M. Stokes, M. Anderson, S. Chandrasekar, and R. Motta. A standard default color space for the Internet - sRGB, 2006. `http://www.w3.org/Graphics/Color/sRGB`.

[San00] S. J. Sangwine. Color in image processing. *Electronics & Communication Engineering Journal*, pages 211 – 219, October 2000.

[Sau42] J. L. Saunderson. Calculation of the color of pigmented plastics. *Journal of the Optical Society of America*, 32:727 – 736, 1942.

[SB75] K. R. Sloan, Jr and R. Bajcsy. A computational structure for color perception. In *Proc. of the ACM Conference*, pages 42 – 45, 1975.

[SCB87] M. W. Schwarz, W. B. Cowan, and J. C. Beatty. An experimental comparison of RGB, YIQ, LAB, HSV and opponent color models. *ACM Transactions on Graphics*, 6(2):123 – 158, 1987.

[Sch05] A. Schuster. Radiation through a foggy atmosphere. *Journal of Astrophysics*, 21:1 – 22, 1905.

[SDF99] Y. Sun, M. S. Drew, and F. D. Fracchia. Representing spectral functions by a composite model of smooth and spiky components for efficient full-spectrum photorealism. In *IEEE Workshop on Photometric Modeling for Computer Vision and Graphics*, pages 4 – 11, June 1999.

[SFCD99] Y. Sun, F. D. Fracchia, T. W. Calvert, and M. S. Drew. Deriving spectra from colors and rendering light interference. *IEEE Computer Graphics and Applications*, pages 61 – 67, July/August 1999.

[SFD98] Y. Sun, F. D. Fracchia, and M. S. Drew. A composite model for representing spectral functions. Technical Report SFU-CMPT-TR1998-18, Simon Fraser University, 1998.

[SFD99] Y. Sun, F. D. Fracchia, and M. S. Drew. Rendering the phenomena of volume absorption in homogeneous transparent materials. In *Proc. 2nd Annual IASTED International Conference on Computer Graphics and Imaging*, pages 283 – 288, October 1999.

[SFDC01] Y. Sun, F. D. Fracchia, M. S. Drew, and T. W. Calvert. A spectrally based framework for realistic image synthesis. *The Visual Computer*, 17:429 – 444, 2001.

[SFF91] M. R. Stytz, G. Frieder, and O. Frieder. Three-dimensional medical imaging. *ACM Computing Surveys*, 23(4):421 – 499, 1991.

[SKB$^+$06] M. Stengert, T. Klien, R. Botchen, S. Stegmaier, M. Chen, and T. Ertl. Spectral volume rendering using GPU-based raycasting. *The Visual Computer*, 2006.

[SLG$^+$96] A. State, M. A. Livingston, W. F. Garrett, G. Hirota, M. C. Whitton, E. D. Pisano, and H. Fuchs. Technologies for augmented reality systems: Realizing ultrasound-guided needle biopsies. In *Proc. SIGGRAPH 96*, pages 439 – 446, 1996.

[Smi78] A. R. Smith. Color gamut transform pairs. In *Proc. 5th Annual ACM SIGGRAPH Conference*, pages 12 – 19, 1978.

[Smi84] A. R. Smith. Plants, fractals and formal languages. *ACM SIGGRAPH Computer Graphics*, 18(3):1 – 10, 1984.

[Smi99] B. Smits. An RGB to spectrum conversion for reflectances. *Journal of Graphics Tools*, 4(4):11 – 22, 1999.

[SMJ93] A. Stockman, D. I. A. MacLeod, and N. E. Johnson. Spectral sensitivities of the human cones. *Journal of the Optical Society of America A*, 10:2491 – 2521, 1993.

[SML96] W. Shroeder, K. Martin, and B. Lorensen, editors. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Prentice-Hall, Upper Saddle River, NJ, 1996.

[SO94] E. Steen and B. Olstad. Volume rendering of 3D medical ultrasound data using direct feature mapping. *IEEE Transactions on Medical Imaging*, 13(3):517 – 525, September 1994.

[Sob94] L. Sobierajski. *Global illumination models for volume rendering*. PhD thesis, State University of New York at Stony Brook, Aug 1994.

[SOM04] A. Sud, M. A. Otaduy, and D. Manocha. DiFi: Fast 3D distance field computer graphics hardware. *Computer Graphics Forum*, 23(3), 2004.

[SP86] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. *Computer Graphics (Proc. SIGGRAPH 86)*, 20(4):151 – 160, 1986.

[SP99] G. Schaufler and M. Priglinger. Efficient displacement mapping by image warping. In *Proc. Eurographics Rendering Workshop*, pages 175 – 186, 1999.

[SPC03] C. Sigg, R. Peikert, and M. Cross. Signed distance transform using graphics hardware. In *Proc. IEEE Visualization 2003*, pages 83 – 90, 2003.

[SS98] A. Stockman and L. T. Sharpe. Human cone spectral sensitivities: A progress report. *Vision Research*, 38(21):3198 – 3206, November 1998.

[SS04] V. Singh and D. Silver. Interactive volume manipulation with selective rendering for improved visualization. In *Proc. IEEE VolVis2004*, pages 95 – 102, Austin, Texas, October 2004. IEEE.

[SSC03] V. Singh, D. Silver, and N. Cornea. Real-time volume manipulation. In *Proc. Eurographics/IEEE TCVG Workshop on Volume Graphics*, pages 45 – 51, 2003.

[SSS92a] B. Smith, C. Spiekermann, and R. Sember. Numerical methods for colorimetric calculations: A comparison of integration methods. *Color Research and Application*, 17:384 – 393, 1992.

[SSS92b] B. Smith, C. Spiekermann, and R. Sember. Numerical methods for colorimetric calculations: Sampling density requirements. *Color Research and Application*, 17:394 – 401, 1992.

[SSS00] B. Smits, P. Shirley, and M. M. Stark. Direct ray tracing of displacement mapped triangles. In *Proc. Eurographics Rendering Workshop*, pages 307 – 318, 2000.

[ST90] P. Shirley and A. Tuchman. A polygonal approximation to direct scalar volume rendering. *Computer Graphics*, 24(5):63 – 70, 1990.

[ST97] G. Sharma and H. J. Trussell. Digital color imaging. *IEEE Transactions on Image Processing*, 6:901 – 932, July 1997.

[SW95] G. Sakas and S. Walter. Extracting surfaces from fuzzy 3D-ultrasound data. In *Proc. 22th Annual ACM SIGGRAPH Conference*, pages 465 – 474, 1995.

[Swe79] Swedish Standards Institute. Colour notation system, 1979. SS 01 91 00.

[Tai92] F. Taillefer. Fast inverse displacement mapping and shading in shadow. In *Proc. Graphics Interface Workshop on Local Illumination*, pages 53 – 60, 1992.

[THB+90] U. Tiede, K.H. Hoehne, M. Bomans, A. Pommert, M. Riemer, and G. Wiebecke. Investigation of medical 3D-rendering algorithms. *IEEE Computer Graphics and Applications*, 10(2):41 – 53, 1990.

[The05] The Imaging Center of Las Cruces Radiology Associates. MRI, ultrasound, mammography, CT scan - Las Cruces Radiology Services, 2005. `http://lc-radiology.com/index.html`.

[The06a] The Association for Computing Machinery. ACM portal, 2006. `http://portal.acm.org/portal.cfm`.

[The06b] The Colour Group, University of East Anglia. Colour constancy, 2006. `http://www.colour-research.com/`.

[The06c] The Colour Group, University of East Anglia. Metamerism, 2006. `http://www.colour-research.com/`.

[Tho73] W. A. Thornton. Matching lights, metamers and human visual response. *Journal of Color & Appearance*, 2(1):23 – 29, 1973.

[TL93] T. Totsuka and M. Levoy. Frequency domain volume rendering. In *Proc. 20th Annual ACM SIGGRAPH Conference*, pages 271 – 278, 1993.

[TN87] W. C. Thibault and B. F. Naylor. Set operations on polyhedra using binary space partitioning trees. *ACM SIGGRAPH Computer Graphics*, 21(4):153 – 162, 1987.

[TPB87] D. Terzopolous, J. Platt, and A. Barr. Elastically deformable models. *Computer Graphics (Proc. SIGGRAPH 87)*, 21(4):205 – 214, 1987.

[Tre53] P. W. Trezona. Addivity of colour equations. *Proceedings of the Physical Society*, B66:548 – 556, 1953.

[Tre54] P. W. Trezona. Addivity of colour equations ii. *Proceedings of the Physical Society*, B67:513, 1954.

[Tru81] J. R. Truckenbrod. Effective use of color in computer graphics. *Computer Graphics*, 15:83 – 90, August 1981.

[TW88] D. Terzopoulos and A. Witkin. Physically based models with rigid and deformable components. *IEEE Computer Graphics and Applications*, pages 41 – 51, November 1988.

[Uni04] University of Joensuu, Finland. Spectra databases, 2004. `http://spectral.joensuu.fi/databases/index.html`.

[VGI94] M. J. Vrhel, R. Gershon, and L. S. Iwan. Measurement and analysis of object reflectance spectra. *Color Research and Application*, 19:4 – 9, 1994.

[Vin92] J. Vince. *3-D Computer Animation*. Addison-Wesley, 1992.

[VS05] F. Vadakkumpadan and Y. Sun. Representing spectral functions using symmetric extension. In R. Eschbach and G. G. Marcu, editors, *Color Imaging X: Processing, Hardcopy and Applications (Proc. of SPIE and IS & T Electronic Imaging 2005)*, pages 10 – 21, 2005.

[VWS05] F. Vadakkumpadan, Q. Wang, and Y. Sun. Computer assisted spectral design and synthesis. In R. Eschbach and G. G. Marcu, editors, *Color Imaging X: Processing, Hardcopy and Applications (Proc. of SPIE and IS & T Electronic Imaging 2005)*, pages 53 – 64, 2005.

[Wan87] B. A. Wandell. The synthesis and analysis of color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):2 – 13, January 1987.

[War04] G. Ward. MGF material libraries, 2004. `http://radsite.lbl.gov/mgf/materials.html`.

[WE02] G. Ward and E. Eydelberg-Vileshin. Picture perfect RGB rendering using spectral prefiltering and sharp color primaries. In *Proc. 13th Eurographics Workshop on Rendering 2002*, pages 117 – 124, June 2002.

[Web98] S. Webb, editor. *The Physics of Medical Imaging*. Institute of Physics Publishing, Bristol, 1998.

[Wei85] K. Weiler. Edge-based data structures for solid modeling in curved-surface environments. *IEEE Computer Graphics and Applications*, pages 21 – 40, January 1985.

[Wes90] L. Westover. Footprint evaluation for volume rendering. *ACM SIGGRAPH Computer Graphics*, 24(4), 1990.

[Wes91] L. Westover. *SPLATTING: A parallel, feed-forward volume rendering technique*. PhD thesis, University of North Carolina at Chapel Hill, 1991.

[WG91] J. Wilhelms and A. Van Gelder. A coherent projection approach for direct volume rendering. *Computer Graphics*, 25(4):275 – 284, July 1991.

[WGG99] B. Wyvill, A. Guy, and E. Galin. Extending the CSG tree. Warping, blending and Boolean operations in an implicit surface modeling system. *Computer Graphics Forum*, 18(2):149 – 158, 1999.

[Whi04] R. Whitaker. Modeling deformable surfaces with level sets. *IEEE Computer Graphics and Applications*, pages 6 – 9, September-October 2004.

[Wil06] P. Willis. Projective alpha colour. *Computer Graphics Forum*, 25(3):557 – 566, 2006.

[Win02] A. S. Winter. *Field-based Modelling and Rendering*. PhD thesis, University of Wales Swansea, December 2002.

[WJ06] S. J. Walton and M. W. Jones. Volume wires: A framework for empirical nonlinear deformation of volumetric datasets. *Journal of WSCG*, 13:81 – 88, 2006.

[WM85] R. Wait and A. R. Mitchell, editors. *Finite element analysis and applications*. John Wiley & Son, Chichester, 1985.

[WM92] P. Williams and N. Max. A volume density optical model. In *Proc. 1992 Workshop Volume Visualization*, pages 61 – 68, 1992.

[WMW86] G. Wyvill, C. McPheeters, and B. Wyvill. Data structure for *soft* objects. *The Visual Computer*, 2:227 – 234, 1986.

[Woo85] T. Woo. A combinatorial analysis of boundary data structure schemata. *IEEE Computer Graphics and Applications*, pages 19 – 27, March 1985.

[WP00] Z. Wu and E. C. Prakash. Visual human animation. In M. Chen, A. E. Kaufman, and R. Yagel, editors, *Volume Graphics*, pages 243 – 252. Springer, London, 2000.

[Wri29] W. D. Wright. A re-determination of the trichromatic coefficients of the spectral colours. *Transaction of the Optical Society*, 30:141 – 164, 1929.

[WS82] G. Wyszecki and W. S. Stiles. *Color Science Concepts and Methods, Quantitative Data and Formulae*. Wiley-Interscience Publication, 1982.

[WS01] R. Westermann and C. Salama. Real-time volume deformations. *Computer Graphics Forum*, 20(3), 2001.

[WW92] A. Watt and M. Watt, editors. *Advanced Animation and Rendering Techniques*. Addison-Wesley, New York, 1992.

[WWT$^+$03] L. Wang, X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, and H.-Y. Shum. View-dependent displacement mapping. *ACM Transactions on Graphics (Proc. SIGGRAPH 2003)*, 22(3):334 – 339, 2003.

[WXS04] Q. Wang, H. Xu, and Y. Sun. Practical construction of reflections for spectral rendering. In *WSCG 2004*, Plzen, Czech Republic, February 2004.

[Yag96] R. Yagel. Towards real time volume rendering. In *Proc. GRAPHICON 96*, pages 230 – 241, 1996.

[YLS00] Y. Yang, F. Lin, and H. S. Seah. Fast multi-resolution volume rendering. In M. Chen, A. Kaufman, and R. Yagel, editors, *Volume Graphics*, pages 185 – 197, London, 2000. Springer-Verlag.

List of Figures

2.1 (a)-(c) Structured grids and (d) unstructured grid. 10

2.2 Classification of data types. 11

2.3 (a) A CT equipment and (b) CT scan [The05]. 11

2.4 (a) An MRI equipment and (b) MRI scan [The05]. 12

2.5 (a) An ultrasound scanner [Fra05] and (b) an ultrasound image [The05]. . . 13

2.6 Simulations of (a) viscoelastic flows and (b) fibre suspensions [Har05]. . . . 14

2.7 Cube indexing. 15

2.8 The 14 basic configurations of the marching cubes algorithm. 17

3.1 The electromagnetic spectrum [McA04]. 39

4.1 The overall reflection and transmittance for two layers, shown in terms of the reflection and transmittance of each individual layer [Kub54]. 54

4.2 A translucent sphere (radius = 1) with an amorphous layer (refraction index = 1) and a solid layer (refraction index = 1.03), rendered with different sampling distances. It is nontrivial for the naked eye to distinguish (b), (c) and (d), where most sampling errors are in fact attributed to refraction rather than the accumulation of reflectance and transmittance. Table 4.3 compares the results of applying the different sampling distances to the translucent sphere against the reference image in (d). The quantitative comparisons are generated using a `ppmdiff` program, which compares two images in the ppm format. 59

4.3 Discrete sampling of a uniform medium of homogeneous layers. 60

4.4 Effects of varying absorption (K), scattering (S) and thickness (x). In each prism, x changes from 0 (left) to 2 (right). Artificial spectra of K and S with a constant value for all wavelengths are used to minimise the chromatic effects in the visual appearance resulting from different reflectance (R) and transmittance (T). All prisms are lit by the same white light source. 62

4.5 Effects of varying absorption (K), scattering (S) and thickness (x). For (a), we compute the local reflectance R_{KM} and transmittance T_{KM} using K and S in the same way as in the Kubelka-Munk model, and set the local properties defined in Noordmans *et al.* [NvS00] as $R_{NVS} = R_{KM}$ and $\alpha_{NVS} = 1 - T_{KM}$. For (b), we set the basic reflectance defined in Bergner *et al.* [BMDF02] as $R_{BMD} = R_{KM}$, and compute the local absorption by setting $\tau(\lambda) = 0.5K(\lambda)\ln 10$. 63

4.6 The geometrical specification of the objects used in Figures 4.4, 4.5, 4.7 and 4.8. To help visualise the geometrical shapes of the objects, the refraction index of the prism is set to 1.03 and that of the spheres to 1.05. The base of the prisms in both (a) and (b) is a right-angle isosceles triangle with adjacent edges of length 2. Both spheres in (b) are of a radius 0.4. 64

4.7 Colour accumulation in the RGBα model, the spectral volume rendering integral by Noordmans *et al.* [NvS00] and Bergner *et al.* [BMDF02] and reflectance accumulation in the Kubelka-Munk theory. The latter is shown to be more optically correct. 66

4.8 Colour accumulation in the RGBα model, the spectral volume rendering integral by Noordmans *et al.* [NvS00] and Bergner *et al.* [BMDF02] and reflectance accumulation in the Kubelka-Munk theory. The latter is shown to be more optically correct. 67

4.9 Volume datasets rendered using the spectral volume rendering integral based on the Kubelka-Munk theory with measured and estimated natural spectra. . 69

4.10 Volume datasets rendered using the spectral volume rendering integral based on the Kubelka-Munk theory with measured and estimated natural spectra. . 70

4.11 The post-illumination pipeline. 71

4.12 Selected wavelengths of a reflectance map, which is computed using single-peak spectral transfer functions for both K and S, with $K_{min} = 0.0001$, $K_{max} = 0.5$, $S_{min} = 0.0001$, $S_{max} = 10$. 72

4.13 Post-illumination with single-peak light source. 73

4.14 Post-illumination with a white light source and multi-peak light sources. . . 73

4.15 Examples of various light sources available for post-illumination. 74

4.16 Post-illumination with multiple light sources. 75

4.17 Post-illumination at $\lambda = 450$, which is computed using single-peak spectral transfer functions for both K and S, with $K_{max} = 0.5$, $S_{max} = 10.0$ and varying values for K_{min} and S_{min} where K_{min} increases from 0.00 to 0.01 in a top to bottom order and S_{min} increases from 0.00 to 0.01 in a left to right order. 76

4.18 Post-illumination at $\lambda = 450$, which is computed using single-peak spectral transfer functions for both K and S, with $K_{min} = 0.0001$, $S_{min} = 0.0001$ and varying values for K_{max} and S_{max} where K_{max} increases from 10^{-10} to 100 in a top to bottom order and S_{max} increases from 1 to 987 in a left to right order. 77

4.19 The user interface of the iPi system. iPi was initially written in C and tested on a Linux environment. It was later rewritten in Microsoft Visual C# .Net development environment; and this version is shown here. 79

4.20 iPi – rendering engine. 81

4.21 iPi – spectral transfer function. 82

4.22 Post-illumination of orange data set with single-peak light source and a white light source. 83

4.23 The user interface of the pre-calculated volume data set. 85

5.1 Results of the union and intersection operations on two spheres. (a) and (b) demonstrate that the union operation of the 4-colour channel model is non-commutative while (c) and (d) demonstrate that the intersection operation of the 4-colour channel model is also non-commutative. The specifications of the objects are as defined in Table 5.4.1. 93

5.2 Results of the modified union operation on two spheres. The specifications of the objects are as defined in Table 5.4.1. 99

5.3 Results of the modified intersection operation on two spheres. The specifications of the objects are as defined in Table 5.4.1. 100

5.4 K, S, R and T are the four properties that are taken into consideration during the calculation of the Kubelka-Munk model. 101

5.5 Kubelka-Munk theory of colour mixing. The colours used in the implementation have been estimated from the MacBeth Color Checker [Pas06]. . . . 103

5.6 The visual representation of Grassmann's laws of colour mixing [Bou47]. . 104

5.7 Grassmann's third laws of colour mixing in the RGB colour space. The input colours have been estimated using the MacBeth Color Checker [Pas06]. 105

5.8 Colour blending of the 4-colour channel model. The input colours have been estimated using the MacBeth Color Checker [Pas06]. 107

5.9 Bouguer-Beer's law of colour mixing. The colours used in the implementation have been estimated from the MacBeth Color Checker [Pas06]. 108

6.1 Examples of combining three SDFs and STFs, **wave1**, **wave1′**, **wave2**, **wave2′**, **wave3**, and **wave3′**, using a CVG operation based on vector addition, and applying the composite SDF and STF to a scaled cubic volume object, **c**. 121

6.2 (c)-(d) Examples of using the *Times* operation to 'mask' a displacement object **open** prior to its application to a cubic volume object **c** and a spheric volume object **s**. The same lookup table was used to create $\mathbf{c}_m$ and $\mathbf{s}_m$. Because the displacement involves motion only in the x and y directions, the effects on the continuity of normals and thickness of the shell become noticeable on **s**. Despite such effects, the examples demonstrate that one can apply a mask to a displacement object according to some segmentation information. (e)-(f) Similar operations that are performed for SDFs can also be simulated with STFs operations. 122

6.3 Examples of applying two SDFs and STFs, **squeeze** and **open**, to a cubic operand object, **c**, in different hierarchical order. **c** is a procedurally-defined volume object, and has a translucent external shell and an opaque internal shell. 125

6.4 The volume scene graphs for Figures 6.3(d) and (e). 126

6.5 Displacement objects can be manipulated in the same way as conventional objects. In (a), a rectangular volume object (left) is deformed by a simple displacement object, with $\psi_x(p) = (0.3)y$ and $\psi_y(p) = \psi_z(p) = 0$ (middle). Application of a transfer function to $\psi_x(p)$ in the form of a lookup table, resulting in a layered deformation (right). In (b), a displacement object is applied to different parts of an operand object with three different geometric transformations. 127

6.6 General identity definition for the combinational ST objects. 128

6.7 General identity definition for the combinational SD objects. 129

6.8 General inverse definition for the combinational ST objects. 131

6.9 General inverse definition for the combinational SD objects. 132

6.10 Three generic types of SDFs. 134

6.11 A combination of **bend** and **twist** operations applied to a Bonsai tree using *type-L SDF*. 137

6.12 Three generic types of STFs. 138

7.1 Comparison between the delta function and Fourier basis on four different RGB coordinates. 146

7.2 Comparison between the Fourier and Gaussian basis on four different RGB coordinates. 148

7.3 Four iterations of the reduction of the negative energy through negative minimisation on four different RGB coordinates. 149

7.4 Four iterations of the reduction of the negative energy through shifting Fourier basis on four different RGB coordinates. 150

7.5 The spectra for the Blue Flower, Bluish Green, Orange, Purplish Blue, Moderate Red, Purple, Yellow Green and Orange Yellow for the Macbeth Color Checker. 154

7.6 The spectra for the Blue, Green, Red, Yellow, Magenta, Cyan, White and Neutral .8 for the Macbeth Color Checker. 155

7.7 The spectra for the Neutral 6.5, Neutral .5, Neutral 3.5, Black, Dark Skin, Light Skin, Blue Sky and Foliage for the Macbeth Color Checker. 156

7.8 Relative errors between the input and output RGB coordinates. 160

7.9 Some of the spectra that have generated using the RGB coordinates. The spectra reconstructed using Methods A, B and C are numerically similar. . . 161

List of Tables

1.1 Thesis nomenclature. 6

4.1 Spatial signature of the RGBα model [CT98]. 56
4.2 Spatial signature of the Kubelka-Munk model. 57
4.3 Quantitative comparison of difference sampling distances. 58
4.4 Attributes associated with the pre-computed volume data sets. 84
4.5 Attributes associated with the spectral transfer function. 84
4.6 System environment settings. 85
4.7 Attributes associated with the interrogative light source. 85

5.1 Spatial signature of the opacity model [CT98]. 89
5.2 CVG Algebra of the opacity model with new operations [CT98]. 90
5.3 A set of algebraic laws for the opacity channel model. 91
5.4 CVG Algebra of the 4-colour channel model [CT98, CT00]. 92
5.5 A set of laws for the 4-colour channel model. 94
5.6 The attribute specifications of the spatial objects shown in Figures 5.1, 5.2 and 5.3. 94
5.7 Spatial signature of the Kubelka-Munk model. 101

6.1 Examples of constructive operations on displacement objects. Except for *Inv*, they are all point-wise binary operations $\Phi(\mathbf{a},\mathbf{b})$, where $\mathbf{a} = (\psi_x, \psi_y, \psi_z)$ and $\mathbf{b} = (\psi_x, \psi_y, \psi_z)$. 118
6.2 Examples of constructive operations on transfer objects. The operations are all point-wise binary operations $\Phi(\mathbf{a},\mathbf{b})$, where $\mathbf{a} = (\Psi_x, \Psi_y, \Psi_z)$ and $\mathbf{b} = (\Psi_x, \Psi_y, \Psi_z)$. 123
6.3 Algebraic properties of operations on displacement and transfer objects. . . 124

7.1 Reflectance data sets with its number of samples. 152

7.2 The results of the Macbeth Color Checker based on the four methods. . . . 153
7.3 Mean results for Methods A and B. 157
7.4 Mean results for Methods C and D. 157
7.5 The results of the RGB coordinates based on the four methods. 159