Computational Colour Science using MATLAB®

Wiley-IS&T Series in Imaging Science and Technology

Series Editor:
Michael A. Kriss

Consultant Editor:
Lindsay W. MacDonald

Reproduction of Colour (6th Edition)
R.W.G. Hunt

Colour Appearance Models (2nd Edition)
Mark D. Fairchild

Colorimetry: Fundamentals and Applications
Noburu Ohta and Alan R. Robertson

Color Constancy
Marc Ebner

Color Gamut Mapping
Ján Morovič

Panoramic Imaging: Sensor-Line Cameras and Laser Range-Finders
Fay Huang, Reinhard Klette and Karsten Scheibe

Digital Color Management (2nd Edition)
Edward J. Giorgianni and Thomas E. Madden

The JPEG 2000 Suite
Peter Schelkens, Athanassios Skodras and Touradj Ebrahimi (Eds.)

Color Management: Understanding and Using ICC Profiles
Phil Green (Ed.)

Fourier Methods in Imaging
Roger L. Easton, Jr.

Measuring Colour (4th Edition)
R.W.G. Hunt and M.R. Pointer

The Art and Science of HDR Imaging
John McCann and Alessandro Rizzi

Computational Colour Science using MATLAB (2nd Edition)
Stephen Westland, Caterina Ripamonti and Vien Cheung

Published in Association with the Society for Imaging Science and Technology

Computational Colour Science using MATLAB®

Second edition

STEPHEN WESTLAND
University of Leeds, UK

CATERINA RIPAMONTI
University College London, UK

and

VIEN CHEUNG
University of Leeds, UK

WILEY
A John Wiley & Sons, Ltd., Publication

Registered office: John Wiley & Sons, Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, United Kingdom

Library of Congress Cataloguing-in-Publication Data

Westland, Stephen.
Computational colour science using MATLAB. – 2nd ed. / Stephen Westland, Caterina Ripamonti, Vien Cheung.
p. cm.
Summary: "Describes how to implement colour science in a way that most other texts barely touch upon"–Provided by publisher.
Includes bibliographical references and index.
ISBN 978-0-470-66569-5 (hardback)
1. Colorimetry. 2. MATLAB. I. Ripamonti, Caterina. II. Cheung, Vien, 1977- III. Title.
QC495.8.W47 2012
535.60285′53--dc23

2012009212

A catalogue record for this book is available from the British Library.

ISBN: 978-0-470-66569-5

Set in 10/12 pt Times by Laserwords Private Limited, Chennai, India

Contents

Acknowledgements

This book makes extensive use of the MATLAB® programme, which is distributed by The MathWorks, Inc. We are grateful to The MathWorks for permission to include extracts of this code.

For MATLAB® product information, please contact:

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760–2098, USA
Email: info@mathworks.com
Web: www.mathworks.com

The code in this book was produced with Bitstream Vera Sans Monospaced font[1]. One of the nice features of this font is that it clearly distinguishes between a zero (0) and a capital O (O). To use this font in MATLAB®, selecting the Custom item under Fonts in the preferences tree. Select the component to change the font for, select the category of font to change and finally select a font in the combo box.

The authors would like to thank Dr Peter Rhodes for his advice and comments regarding methods to plot representations of the chromaticity diagram and for the use of the DigiEye systems for the images used in the case study of camera characterisation; Marjan Vazirian for the colour measurements of the case study of CRT characterisation.

[1] Available from ftp.gnome.org/pub/GNOME/sources/ttf-bitstream-vera/1.10/

About the Authors

Stephen Westland

Stephen Westland was awarded his BSc and PhD from the University of Leeds. In 1986 he joined Courtaulds Research as a Colour Physicist before returning to academia in 1990 to work at the University of Keele. He worked as a post-doctoral researcher and lecturer in the Department of Communication and Neuroscience where he taught and researched colour measurement, human colour vision, computational imaging and image processing. In 1990 he was appointed as a Reader in Colour Imaging at the Colour Imaging Institute of the University of Derby. He was appointed as Professor of Colour Science and Technology in the School of Design at the University of Leeds in 2003 where he currently teaches and researches.

He has published more than 100 refereed papers in the areas of colour imaging, colour management, colour physics and colour design. He has been active in professional bodies that are concerned with colour. He is an active participant in conferences organised by the Society of Imaging Society and Technology (IS&T) and served on several organisational and technical committees. In 2008 he was awarded a Fellowship of the Society of Dyers and Colourists and the Davies Medal from the Royal Photographic Society for his research on digital colour imaging.

Caterina Ripamonti

Caterina Ripamonti was awarded her BA in Experimental Psychology from the University of Trieste (Italy) and her PhD in Vision Science from the University of Derby (UK). She worked as a post-doctoral fellow at the University of Pennsylvania (US), and at the University of Cambridge (UK). Since 2005, she has been working as a senior research fellow at the Institute of Ophthalmology, University College London (UK). From 2006–2008 she was appointed as Lecturer in Visual Communication at Universitá Suor Orsola Benincasa (Italy) where she taught human vision and visual communication to graduate students. She is also an honorary research fellow at Moorfields Eye Hospital (UK), and the Membership Secretary of The Colour Group (GB). Her research interests are in human colour vision, spatial and temporal properties of normal and defective vision of children and adults, and applied aspects of colour science related to human factors.

Vien Cheung

Vien Cheung completed her first degree in Textile Chemistry at The Hong Kong Polytechnic University (Hong Kong), an MSc in Colour Imaging at the University of Derby (UK) and a PhD in Colour and Imaging Science at University of Leeds (UK). She currently holds an academic post as Lecturer in Colour Theory in the School of Design at the University of Leeds.

Her research interests centred in colour applied to art, design, imaging and vision. She is active in professional bodies including The Colour Group (Great Britain), the Society of Dyers and Colourists (UK), the Society of Imaging Science and Technology (USA) and the Tsinghua Art & Science Colour & Imaging Institute (China). She is also a member of the International Editorial Panel of Coloration Technology. She received the Selwyn Award from The Royal Photographic Society in 2008 for her research in colour imaging and imaging technology; and a Silver Medal from the Society of Dyers and Colourists in 2011 in recognition of her contributions to education, the Society and in the interests of the allied industries.

1
Introduction

1.1 Preface

The growing importance of colour science in the manufacturing industry has resulted in the availability of many excellent text books: existing texts describe the history and development of the CIE system (Wyszecki and Stiles, 1982; Hunt, 1998; Ohta and Robertson, 2005; Schanda, 2007; Hunt and Pointer, 2011), the prediction of colour difference (McDonald, 1997a; Berns, 2000; Luo, 2002a) and colour appearance (Fairchild, 2005), the relationship of the CIE system to the human visual system (Wandell, 1995; Kaiser and Boynton, 1996), and applications of colour science in technology (Green and MacDonald, 2002; Green, 2010). However, the field of colour science is becoming ever more technical and although practitioners need to understand the theory and practice of colour science they also need guidance on how to actually compute the various metrics, indices and coordinates that are useful to the practicing colour scientist. *Computational Colour Science Using MATLAB®* was published to address this specific need (Westland and Ripamonti, 2004). It described methods and algorithms for actually computing colorimetric parameters and for carrying out applications such as device characterisation, transformations between colour spaces and computation of various indices such as colour differences. There are a number of reasons why a second edition has now been published. Firstly, the last decade has seen a number of developments that are important but which were not included in the first edition; secondly, some notable topics were omitted from the first edition and are now included as additional chapters in this edition; thirdly the toolbox was originally written to emphasise clarity (for teaching purposes) but somewhat at the expense of performance (the authors now feel that a better balance between clarity and performance can be achieved and therefore all of the MATLAB® code has been rewritten); fourthly, the presentation of the text has been rewritten to provide a more

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.

logical and consistent presentation; fifthly, the comprehensive use of colour throughout the second edition provides opportunities to include topics that were more difficult to include in the first edition.

In preparing this edition, and the previous edition, a difficult decision was required on what level of existing knowledge we assume the reader has. However, this book is not intended as a primer for colorimetry and the CIE system. It is clear that a number of excellent texts that address this purpose already exist. Therefore, we assume a reasonable understanding of the main principles of the CIE system of colorimetry although a brief revision aid is provided for those readers who may find this useful. Readers who wish to explore the theoretical and historical backgrounds of the topics covered by this book are encouraged to review the alternative texts mentioned above and referred to within this text. We have written this book primarily for master's and doctoral students undertaking research in colour science since this is the book that we would have liked to have had access to when we undertook our own doctoral research. However, we are confident that computer programmers, colour-image engineers and academics will find this book and the associated MATLAB® code useful.

Finally, we note that the term *colour science* could be defined quite broadly to include topics such as colour chemistry, materials science, imaging science and a myriad of industrial applications that involve colour. Our definition of colour science is the *perception, measurement and communication of colour*.

1.2 Why Base this Book on MATLAB®?

This book describes algorithms and mathematical procedures in colour science and illustrates these procedures using the numerical software tool called MATLAB®. MATLAB® provides several features that make it suitable for the implementation of algorithms in general and colour-science algorithms in particular, and results in code that is easily understandable by readers even if they have relatively little experience of writing software. These features include the use of operations upon vectors and matrices to enable compact code that avoids the excessive use of looping procedures, the provision of a massive library of functions that the MATLAB® programmer can call upon, and the ease of use of graphics functions to enable the user to easily and effectively visualise complex data structures.

Most computer languages are very dependent upon a variety of 'looping' procedures to execute summations or to implement iterative techniques whereas MATLAB® enables these types of operations to be performed with a fraction of the code that would otherwise be required. For example, if we have two variables x and y that each consist of five entries and we wish to compute the product of the corresponding entries and then sum the results to yield a single number, we might write code that in BASIC looks like the following:

```
sum = 0
FOR i = 1 TO 5
sum = sum + x(i)*y(i)
NEXT i
```

In MATLAB® these four lines of BASIC code shown could be written as:

```
sum = 0
for i = 1:5
sum = sum + x(i)*y(i)
end
```

Note the small differences in syntax between the two languages. However, in MATLAB® we can also use the elegant equivalent code thus:

```
sum = x*y;
```

Expressed in terms of linear algebra MATLAB® will perform the inner product of the 1×5 row vector **x** and the 5×1 column vector **y**. In the MATLAB® environment it is not necessary to specify how many entries the variables contain, as long as the dimensions of these variables define a valid matrix operation (in this case the row vector needs as many entries as the column vector). A variable in MATLAB® can represent a single number, a row or column vector, or a matrix (or array, as matrices are sometimes called). The operation given, for example, by:

```
y = 2*x;
```

will assign to **y** twice the value of **x** if **x** defines a single number, but twice the value of every element in **x** if **x** is a vector or a matrix. The compact nature of MATLAB® code allows complex and sophisticated algorithms to be explained and demonstrated with clarity and accuracy. Moreover, the computation of many colorimetric terms is ideally suited to a language that expresses variables in terms of matrices and vectors. For example, the calculation of CIE tristimulus values is essentially the inner product of certain matrices (typically the inner product of a 1×31 row vector with a 31×3 matrix when the calculations are being carried out at 31 wavelength intervals, as is common). Some procedures are best explained or implemented using loops, however, and for these situations MATLAB® does provide *for* and *while* looping structures which work broadly in a way that will be familiar to programmers who are used to languages such as C or BASIC.

The second strength of MATLAB® is that it includes an encyclopaedic collection of subprograms, called M-files, for the solution of nearly any numerical problem. Although this book is not principally concerned with generic numerical analysis, but rather with particular colorimetric algorithms, the M-files that are available with MATLAB® are useful for many computations in colour science. MATLAB® provides many functions (such as those with the ability to solve systems of simultaneous linear equations) and if it was necessary to spend time explaining these in detail or writing code to implement them

it would detract from the main focus of this book which is colour science. Readers may wish to refer to other text books (e.g. Press *et al*., 1993) that address implementations of numerical analysis in programming languages such as C if they wish to convert the code in this book into other programming languages.

Perhaps MATLAB's® most spectacular feature is its capability to display graphics. Two- or three-dimensional graphs are easily constructed by even a novice MATLAB® user. Thus:

```
x = [1 2 3 4 5];
y = [3 5 7 9 11];
plot(x,y)
```

is sufficient code to plot a graph of the five values in the vector **y** against those in the vector **x**. Experienced programmers will find it trivial to construct sophisticated and informative graphs and the ability to almost effortlessly visualise data is one of the main advantages of using MATLAB® in a research environment. MATLAB® allows the user to answer complex 'what if?' questions with just a handful of code lines. MATLAB® is therefore an ideal experimental or prototyping language even if it lacks the run-time speed of some other languages such as C.

MATLAB® can be confusing, however, for users new to programming or who do not have a reasonable understanding of linear algebra. For this reason, Chapter 3 provides a gentle introduction to MATLAB® and Chapter 2 provides a basic introduction to linear algebra and the notation that is used throughout this book. Where possible, the code that is presented has been written for clarity rather than for efficiency or speed of computation to allow readers to understand the computational principles involved and to be able to implement them in a wide variety of programming languages. In general, special MATLAB® commands have been avoided, even though their use may have made the code more efficient, to reduce the effort that would be required to translate the code into a language such as C or C++. One exception, however, is the backslash operator, which is described in Chapter 3. Programmers who wish to use languages other than MATLAB® may wish to create their own version of the backslash operator in order to easily translate the code within this book. All of the MATLAB® code contained within this book can be downloaded from the MATLAB® website: www.mathworks.com/matlabcentral/.

1.3 A Brief Review of the CIE System of Colorimetry

Light is a term that we use to describe electromagnetic radiation, in the approximate wavelength range 360–780 nm, to which the human visual system is sensitive. When we observe the light reflected from surfaces in a scene, or when we look directly at the light emitted by light sources, we experience the sensation of colour. Colour is just one attribute of a complex and not fully understood set of properties that define the appearance of the world. Surfaces interact with light in a complex and varied way that includes processes of absorption, scattering, refraction and diffraction but it is the light that is reflected by the surfaces in a scene that we use to identify those surfaces by

their colour. The reflectance properties of surfaces can be defined by spectral reflectance factors that are normally measured at regular intervals in the visible spectrum of radiation. The reflectance factor of an object at a certain wavelength (or wavelength interval) is the proportion of light at that wavelength that is reflected by the object and is never less than zero and only occasionally greater than unity. The term *surface reflectance factor* is used by some authors but this is somewhat confusing since it could imply that the light is reflected at the air/material surface of the object. Although a small amount of light (typically about 4% for inks and paints) is reflected at the surface, the majority of reflected light results from scattering processes that occur within the body of an object after the light has passed through the air/material surface. Commercially available reflectance spectrophotometers are able to measure reflectance factors (typically at intervals of 10 nm in the range 400–700 nm though some instruments extend their measurements to shorter or longer wavelengths). The quantity and quality of light that we see when we look at a point in a scene clearly depends upon the spectral power distribution of the illuminating source and the reflectance properties of the scene at that point. Our visual systems detect the reflected light using the light-sensitive sense organs or retinas that form the inner lining of the back of the eyeball. Light enters the eye through the pupil and is focused onto the retina by the lens. The retina consists of a mosaic of specialised cells called rods and cones that contain pigments that respond to light. The chemical changes that take place when the visual pigments in the rods and cones absorb light initiate electrochemical impulses that are subsequently processed by a neural network of brain cells and which eventually lead to the excitation of cells in various specialised areas of the outer region of the brain known as the cortex. It is still unknown where in the brain colour perception actually occurs, if indeed it occurs in any localised area, but activity in the visual cortex at the back of the brain (the occipital lobe) is strongly implicated. In Chapter 7, we will examine the properties of the visual system in more detail, as we discuss the way cone responses combine their outputs into post-receptoral channels and how to compute both receptoral and post-receptoral receptoral responses. In the present section, we provide only a minimal summary of the retinal processes, before methods for the measurement of colour are outlined.

Human vision is mediated exclusively by rods at low levels of illumination (below $0.001\ \text{cd/m}^2$) also referred to as scotopic or night vision. At medium or mesopic levels of illumination (between $0.001 - 10\ \text{cd/m}^2$) both rods and cones can contribute to vision. At higher or photopic levels of illumination (above $10\ \text{cd/m}^2$), vision is mediated solely by the responses of the cones of which there are three types with sensitivities peaking at 420 nm (short wavelengths), 530 nm (medium wavelengths) and 560 nm (long wavelengths), termed S, M and L cones respectively (Bowmaker, 2002). The three classes of cones are not distributed evenly throughout the retina (Williams *et al*., 1981). For example, the central or foveal region is occupied predominantly by L and M cones and there are approximately twice as many L cones as there are M cones. The S cones are rare throughout the retina but are more concentrated in a ring around the fovea. The retina contains several layers of cells and the signals generated by the transduction of light into chemical and electrical energy in the cones activate the bipolar and ganglion cells before leaving the eye via the optic nerve. The cones and rods each provide a univariant response (Rushton, 1972; Wandell, 1995) and the consequence of this is that the individual classes of cones and rods are colour blind. That is, the scotopic visual system can

only perceive shades of grey and the responses of single photoreceptor types considered independently are also incapable of wavelength discrimination. At least two types of photoreceptor are required for colour vision. Under mesopic and photopic conditions, the visual system achieves colour vision by analysing the relative responses of at least two types of photoreceptor.

The Commission Internationale de l'Éclairage (CIE) developed a system for the specification of colour stimuli that was recommended for widespread use in 1931. Perhaps the most important step that allowed this development was the understanding of additive colour mixing. Thus, all colour stimuli can be matched by the additive mixture of three appropriately chosen primaries. In the 1920s it had long been recognised that the amounts or intensities of the primaries required to match a given stimulus effectively form a specification of the stimulus in terms of the primaries that are used. The amounts of the primaries used for any given stimulus are commonly known as the tristimulus values. It is possible to determine the tristimulus values for any given stimulus using a device known as a split-field or bipartite colorimeter. In such a device an observer views a bipartite field. On one side of the field the stimulus is displayed; on the other side the additive mixture of the three primaries is displayed. The observer adjusts the intensities of each of the three primaries until the additive mixture is visually indistinguishable from the stimulus. Under the matching condition the field appears uniform to the observer and the tristimulus values can be read off from the device and recorded.

The measurement of the colour-matching functions of observers was a critical feature in the development of the 1931 CIE system of colorimetry since they allowed the computation of the tristimulus values for a known stimulus without the need to view the stimulus in a bipartite colorimeter. The colour-matching functions are the amounts of three primaries required to match one unit of intensity of a single wavelength of light, and were recorded for small wavelength intervals throughout the visible spectrum. If red, green and blue primaries are used and these are denoted by the symbols [R], [G] and [B], and the tristimulus values are represented by the symbols R, G and B, then it is possible to write an equation to denote the matching condition thus:

$$S \equiv R[R] + G[G] + B[B] \tag{1.1}$$

In this Equation the symbol $\equiv$ means 'is matched by' and the stimulus that is being matched is denoted by the symbol S. If the tristimulus values are measured separately for each wavelength in the visible spectrum then we obtain the tristimulus values as functions of wavelength λ thus: $R(\lambda)$, $G(\lambda)$, and $B(\lambda)$. These three functions of wavelength are called colour-matching functions. The additivity and linearity of colour matches allow an important property: if a stimulus S_1 is matched by R_1, G_1, B_1 and a stimulus S_2 is matched by R_2, G_2, B_2 then it is possible to predict in advance the tristimulus values that define a match to the stimulus defined by the additive mixture $S_1 + S_2$. Thus we can simply write:

$$S_1 + S_2 \equiv (R_1 + R_2)[R] + (G_1 + G_2)[G] + (B_1 + B_2)[B] \tag{1.2}$$

Since any real stimulus can be considered to be the sum of energy at many different wavelengths, it is possible to predict the tristimulus values for any stimulus in a similar way (without having to resort to physically determining a visual match for that stimulus using a bipartite colorimeter) given that the colour-matching functions are known.

In fact, experiments were carried out prior to the publication of the CIE system by two groups of workers headed by Wright in 1929 and Guild in 1931 (see Wyszecki and Stiles, 1982) to determine colour-matching functions. The two groups of workers used different primaries and consequently the two sets of colour-matching functions were different. This raises an interesting question of whether the colour-matching functions are arbitrary, given that there is a very wide choice in the selection of the primaries. Certainly, the actual tristimulus values obtained for a given stimulus are arbitrary in that they would be different if a different set of primaries was chosen. However, the matching condition is valid no matter which primaries are selected subject to some simple criteria (for example, the primaries must be independent; in other words, it must not be possible to match one of the primaries using an additive mixture of the other two). This means that if two stimuli are a visual match and are specified by the same tristimulus values under Guild's system, then the two stimuli would also match under Wright's system. Furthermore, the two stimuli would be a match under a system defined by any other set of three independent primaries.

It is possible to convert tristimulus values from one system to another by a simple linear transform (see Chapter 2). It is also possible to compute the colour-matching functions for a set of known primaries given the colour-matching functions of another set of primaries. Thus, in 1931, the CIE transformed the two sets of colour-matching functions obtained from experiments carried out by Wright and Guild into a single set of colour-matching functions and reassuringly found good agreement between the two sets of data. The CIE system as we know it today is based upon a transformation of the original colour-matching functions averaged from Guild and Wright to a set of primaries known as X, Y and Z. The colour-matching functions are known for each wavelength and are therefore represented by $\bar{x}(\lambda)$, $\bar{y}(\lambda)$ and $\bar{z}(\lambda)$.

Although it is commonly stated that any colour can be matched y three primaries, this statement requires careful understanding. If we restrict ourselves to using three fixed (and real) primaries and if we only consider positive amounts of the three primaries then the statement is certainly false. It is impossible to select three real primaries and use these in additive mixture to match all possible colour stimuli. If the primaries are chosen carefully then a good proportion of the possible colours can be matched, but not all of them. Therefore, when Wright and Guild carried out their experiments they found that negative amounts of one of the primaries were sometimes required. Practically, this meant that in the split-field colorimeter one of the primaries would be added to the stimulus, the resultant additive mixture then being matched by the other two primaries. Under these conditions, when a negative amount of the red primary was being used, we could rewrite Equation 1.1 as:

$$S + R[R] \equiv G[G] + B[B] \tag{1.3}$$

or, its mathematical equivalent:

$$S \equiv -R[R] + G[G] + B[B] \tag{1.4}$$

from which the notion of negative amounts of a primary is apparent. The implication of this is that in the original colour-matching functions of Guild and Wright some of the entries were negative and some of the tristimulus values would be negative. For largely

pragmatic reasons, the CIE decided to transform the colour-matching functions to a set of primaries for which all of the entries would be positive and all of the tristimulus values would be positive for all real colours. This was achieved mathematically by defining a set of primaries (XYZ) that were more saturated than any real colour. The CIE XYZ primaries are sometimes referred to as imaginary primaries since they cannot be realised by any real lights.

The CIE also defined standard illuminants – tables of spectral power distributions – that can be used to compute the colour signal for a surface given the spectral reflectance factor of the surface. The introduction of tables of illuminants allowed the computation of tristimulus values for surface colours as well as for self-luminous colours. A practical equation for computing the CIE 1931 tristimulus values for a surface with spectral reflectance $P(\lambda)$ under an illuminant of relative spectral power $E(\lambda)$ is:

$$\begin{aligned} X &= k\Sigma E(\lambda)P(\lambda)\bar{x}(\lambda)d\lambda \\ Y &= k\Sigma E(\lambda)P(\lambda)\bar{y}(\lambda)d\lambda \\ Z &= k\Sigma E(\lambda)P(\lambda)\bar{z}(\lambda)d\lambda \end{aligned} \tag{1.5}$$

where $k = 100/(\sum \bar{y}(\lambda)E(\lambda)d\lambda)$.

At each wavelength interval the product $E(\lambda)P(\lambda)$ gives the amount of energy in the stimulus at wavelength λ and the amount of the primary required to match this is given by multiplying this product by the colour-matching function at that wavelength. In order to arrive at the amount of the primary required to match the stimulus it is only necessary to sum across all wavelengths (Equation 1.5). Notice that the implication of the normalising factor k is that the *absolute* spectral power distribution for the illuminant is not required so that, for surface colours at least, $Y = 100$ for a perfect white surface (for which $P(\lambda) = 1$ for all λ). Further, note that a perfect white surface will give $Y = 100$ for *any* illuminant E. This normalisation is reasonable given the processes of adaptation that take place in our everyday vision. In order to appreciate these processes, imagine a piece of white paper with reflectance 1 at all wavelengths and a piece of black coal with reflectance 0.01 at all wavelengths. Now consider viewing these two surfaces indoors (under an equal-energy light source with 100 units of light at each wavelength) and outdoors (under an equal-energy light source with 10 000 units of light at each wavelength). When viewed indoors the paper reflects 100 units of light at each wavelength whereas the coal reflects only 1 unit of light at each wavelength, but the amount of light reflected outdoors is 10 000 and 100 for the paper and coal respectively. Even though the paper reflects 100 times as much light outdoors as it does indoors, the colour appearance of the paper remains approximately constant under the two light sources. More surprisingly, the coal reflects as much light outdoors as the paper does indoors and yet the coal is veridically seen as black. This remarkable property of colour constancy is central to our whole visual experience (the example just given is strictly lightness constancy but can be extended to consider colour appearances under illuminants that vary in colour as well as intensity). The normalising factor k in the CIE system ensures that for a perfectly white surface the Y tristimulus value will always be 100 irrespective of the quantity and quality of the illuminant. One consequence of this normalisation is that it is only necessary to know the relative energy of the illuminant at each wavelength in order to compute CIE XYZ values.

The CIE (1931) colour-matching functions were derived from *RGB* colour-matching experiments that used a bipartite field that subtended 2° (in terms of visual angle) at the retina. A second set of colour-matching functions were measured and introduced in 1964 using a larger field size of 10°. The 1931 and 1964 colour-matching functions are based on the same *XYZ* primaries but exhibit some marked differences. One reason for this is that the distribution of cones (the light-sensitive cells in the eye) is not uniform across the retina (Williams *et al*., 1981). For example, it is known that there are no cones that contain short-wavelength-sensitive pigment in the central region of the retina known as the foveola. The present situation whereby there are two sets of colour-matching functions known as the 2-degree (1931) and the 10-degree (1964) standard observers has served the colour industry well over the last 70 years but is ultimately unsatisfactory. Users need to make a choice based upon which set of colour-matching functions best represents any given viewing situation. This presents problems from time to time when the size of the stimulus is not exactly 2° or 10°. The CIE is currently working towards the development of a set of colour-matching functions that vary continuously for a wide range of stimulus sizes.

The CIE *XYZ* tristimulus values specify a colour stimulus in terms of the visual system. It is often useful, however, to compute chromaticity coordinates *xy* from the tristimulus values thus:

$$x = X/(X + Y + Z)$$
$$y = Y/(X + Y + Z) \tag{1.6}$$

The chromaticity diagram is derived by plotting y against x and this provides a useful map of colour space. However, it should be noted that stimuli of identical chromaticity but different luminance are collapsed onto the same point in the 2D plane of the chromaticity diagram. One of the benefits of the chromaticity diagram is that, according to Grassman's law, additive mixtures of two primaries fall on a straight line joining the two points that represent the two primaries in the chromaticity diagram. If three primaries are used then the gamut of the additive system is given by a triangle, with the vertices defined by the chromaticities of the three primaries. The gamut of all physically realisable colours is contained by the convex shape of the spectral locus and a straight line that can be considered to be drawn between the two ends of the locus. It can readily be seen that this is so if one considers any real colour stimulus to consist of the additive sum of energy at individual wavelengths.

The CIE system of colorimetry is a system of colour specification. However, it has two limitations that it is important to understand. Firstly, the system was designed for colour specification rather than for colour appearance. The chromaticities of a perfect reflecting diffuser will change as the illumination changes. However, it has already been mentioned that the colour appearance of such a surface would be expected to remain approximately constant under quite large changes in illumination. Secondly, the system is perceptually nonuniform. For a given Euclidean distance between two points in *XYZ* space the magnitude of the perceptual colour difference between the two stimuli represented by those points can vary by an order of magnitude or more. This second limitation in particular has presented industrial practitioners of colorimetry with serious problems and even today these problems have not all been resolved. Although it is useful to be able to state that two stimuli are a visual match (under the strict conditions under

which the colour-matching functions were derived) if they have the same tristimulus values, it would perhaps be even more useful to be able to predict the visual difference between two stimuli whose tristimulus values are not identical. Ideally, we would like a uniform colour space in which equal distances in that space correspond to equal perceptual differences.

A major advance was made by the CIE in 1976 with the introduction of the CIELAB system of colour specification. This nonlinear transform of the XYZ values provided partial solutions to both the problems of colour appearance and colour difference. The transformation from tristimulus values to $L^*a^*b^*$ coordinates is given by:

$$\begin{aligned} L^* &= 116(Y/Y_n)^{1/3} - 16 \\ a^* &= 500\left[(X/X_n)^{1/3} - (Y/Y_n)^{1/3}\right] \\ b^* &= 200\left[(Y/Y_n)^{1/3} - (Z/Z_n)^{1/3}\right] \end{aligned} \tag{1.7}$$

where X_n, Y_n and Z_n are the tristimulus values of a specified white achromatic stimulus[1].

CIELAB provides a three-dimensional colour space where the a^* and b^* axes form one plane and the Lightness L^* axis is orthogonal to this plane. The CIELAB transform is generally used for surface colours (a separate transform, CIELUV, was provided for use with self-luminous colour stimuli such as those generated using additive colour-reproduction devices) and includes several interesting features.

Firstly, the inclusion of difference signals crudely models processes that are believed to take place in the human visual system. Thus, whereas the retina initially captures responses derived from the cone spectral sensitivities, these responses are combined at an early (retinal) stage of visual processing to provide a luminance signal and two opponent signals that can be described as being yellow-blue and red-green. Similarly, CIELAB represents colour stimuli as an achromatic signal (L^*) and two chromatic channels representing yellow-blue (b^*) and red-green (a^*).

Secondly, the normalisation by the illuminant achieves a colour space that makes better predictions of colour appearance than the tristimulus space from which it is derived (Hunt and Pointer, 2011). Thus, whereas the xy chromaticities of a perfect white surface vary with the illuminant, the CIELAB coordinates remain constant at $L^* = 100$, $a^* = b^* = 0$. CIELAB also allows the representation of a colour stimulus by dimensions of lightness, chrome and hue and it is therefore reasonable to describe CIELAB as a colour-appearance space whereas this label is not appropriate for tristimulus space which is strictly only for colour specification. However, if predictions of colour constancy using CIELAB are compared with empirical measurements of colour constancy then it is found that the predictions are quite poor in general. The field of colour appearance has been actively researched over the last decade in particular and several advanced colour spaces (e.g. CIECAM02) are now available for predicting colour appearance.

Thirdly, the nonlinear transform of tristimulus values in the CIELAB equations allows the Euclidean distance between two points in the new space to better predict the visual colour difference between the colour stimuli represented by those two

[1] Slightly different equations are used for certain stimuli. See Chapter 5 for the complete equations.

points. Consequently, the colour difference metric known as ΔE^*_{ab} and computed by the equation:

$$\Delta E^*_{ab} = \sqrt{(\Delta L^*)^2 + (\Delta a^*)^2 + (\Delta b^*)^2} \tag{1.8}$$

where ΔL^*, for example, denotes the difference in L^* between the two samples, has been used effectively to quantify colour difference in a wide range of industries.

Unfortunately, although CIELAB is more perceptually uniform than *XYZ* space it is still a long way from being perceptually uniform. Industrial practitioners of colour science would like to be able to apply a single tolerance on the value of ΔE^*_{ab} that defines the perceptibility or acceptability boundaries throughout colour space but this is not possible. The last two decades of the twentieth century saw a great deal of research into the development of more effective colour-difference formulae. The CMC formula (named after the Colour Measurement Committee of the Society of Dyers and Colourists) was introduced in 1983 and has been widely used in industry (Clarke, McDonald and Rigg, 1984). However, a new recommendation for colour difference was introduced by the CIE and is known as CIEDE2000 (Luo, Cui and Rigg, 2001). CIEDE2000, like its predecessor CMC, is not in itself a colour space (it computes colour difference starting from differences in CIELAB space) but rather describes a method for combining and weighting the differences that is more complex, and certainly more effective, than simply measuring the Euclidean distance.

Systems that are able to better predict colour difference and colour appearance are currently active areas of research for colour scientists in academia and industry. One of the factors driving research in these areas is the need to be able to effectively communicate colour between image-capture and image-reproduction devices. The proliferation of inexpensive colour capture and display systems, in addition to the increasing commercial use of colour on the internet, requires increased understanding of, and ability to predict, colour difference and colour appearance.

A wide range of readable and informative texts exist for the reader who would like to explore the background and methods of the CIE system in more detail (e.g. McDonald, 1997a; Berns, 2000; Ohta and Robertson, 2005; Schanda, 2007; Hunt and Pointer, 2011).

2

Linear Algebra for Beginners

2.1 Some Basic Definitions

A matrix is a rectangular array of numbers and the numbers in the array are called the entries in the matrix. A two-dimensional matrix with one dimension equal to 1 is sometimes called a row matrix (a matrix with only one row) or a column matrix (a matrix with only one column). A matrix with both dimensions equal to 1 is simply a single number which we can also call a scalar. It is conventional to denote matrices by boldface upper-case symbols and row or column vectors by lower-case symbols. So, for example, the matrix **A**, where:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.1}$$

is a 2×2 matrix with four entries. Since only the diagonal entries are non-zero we can state that **A** is a diagonal matrix (furthermore, a diagonal matrix whose diagonal entries are all 1 is also called an identity matrix).

Two matrices are defined to be equal if they have the same size and their corresponding entries are equal. If **A** and **B** are matrices of the same size, then the sum $\mathbf{A} + \mathbf{B}$ is the matrix obtained by adding the entries of **B** to the corresponding entries of **A**, and the difference $\mathbf{A} - \mathbf{B}$ is the matrix obtained by subtracting the entries of **B** from the corresponding entries of **A**. Only matrices of the same size can be added or subtracted. As an example, if we defined matrix **B** by:

$$\mathbf{B} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \tag{2.2}$$

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.

then we can write that:

$$\mathbf{A}+\mathbf{B}=\begin{pmatrix}2 & 2\\ 3 & 5\end{pmatrix} \tag{2.3}$$

and that:

$$\mathbf{A}-\mathbf{B}=\begin{pmatrix}0 & -2\\ -3 & -3\end{pmatrix} \tag{2.4}$$

If $\mathbf{A}$ is an $m \times r$ matrix and $\mathbf{B}$ is an $r \times n$ matrix, then the product $\mathbf{AB}$ is the $m \times n$ matrix whose entries are determined as follows. To find the entry in row i and column j of $\mathbf{AB}$, single out row i from matrix $\mathbf{A}$ and column j from matrix $\mathbf{B}$, multiply the corresponding entries from the row and column together and then sum the resulting products. Thus:

$$\mathbf{AB}=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix} \tag{2.5}$$

In order to see how the entries were created in $\mathbf{AB}$, note that for the (i,j) entry where $i=j=1$, we took the values 1 and 0 from the first row of $\mathbf{A}$ and the values 1 and 4 from the first column of $\mathbf{B}$ to yield the calculation $1 \times 1 + 0 \times 3 = 1$. Notice that multiplying matrix $\mathbf{B}$ by $\mathbf{A}$ resulted in matrix $\mathbf{AB}$ which was the same as $\mathbf{B}$. This special situation occurred because matrix $\mathbf{A}$ is the identity matrix. We can therefore note that multiplying a matrix by the identity matrix is like multiplying a scalar by unity. It should also be clear that a matrix $\mathbf{B}$ may only be multiplied by a matrix $\mathbf{A}$ if the number of columns in $\mathbf{A}$ is equal to the number of rows in $\mathbf{B}$.

2.2 Solving Systems of Simultaneous Equations

Imagine that we wish to solve a problem where we need to find the values of two variables, x and y, and we are given knowledge of two relationships between the two variables. For example, we might be told that the sum of the two variables is 6 and the difference between the two is 3. We can represent this problem by a pair of simultaneous equations thus:

$$\begin{aligned} x+y&=6\\ x-y&=3 \end{aligned} \tag{2.6}$$

Many readers will be familiar with this sort of problem and will possess the algebraic skills to rearrange these two equations into a form that enables one of the variables to be eliminated. In this trivial example, we can simply add the two equations together to give:

$$2x=9 \tag{2.7}$$

from which it is now obvious that $x=4.5$ and (by subsequent substitution) that $y=1.5$.

However, it is often convenient to represent the problem in matrix form. The two simultaneous (or coupled) linear equations can be written as a single matrix equation of the form:

$$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 6 \\ 3 \end{pmatrix} \tag{2.8}$$

We can further simplify the notation by writing:

$$\mathbf{Ap} = \mathbf{q} \tag{2.9}$$

where $\mathbf{q}$ and $\mathbf{p}$ are 2×1 column matrices and $\mathbf{A}$ is a 2×2 matrix. Notice that the 'inner' dimensions of the terms that are being multiplied together match; thus $\mathbf{Ap}$ is $(2 \times \underline{2})(\underline{2} \times 1)$. Matrices can only be multiplied together if their inner dimensions match in this way and matrix multiplication is sometimes referred to as computing the inner product. Notice also that the dimensions of the result of computing the inner product are given by the outer dimensions. Thus, the result of a $(\underline{2} \times 2)(2 \times \underline{1})$ multiplication is a 2×1 matrix.

Matrix notation is concise and provides an alternative way to arrive at a solution to Equations 2.6. In order to solve the problem we need to compute the inverse of the matrix $\mathbf{A}$. The inverse of a matrix $\mathbf{A}$ is denoted as $\mathbf{A^{-1}}$ and defined by:

$$\mathbf{I} = \mathbf{AA^{-1}} \tag{2.10}$$

where $\mathbf{I}$ is the identity matrix (or unit matrix). (Note that it is only possible to compute the inverse for matrices that are square and non-singular. A singular matrix, whose determinant is zero, cannot be inverted.)

We can now multiply both sides of Equation 2.9 by the inverse of $\mathbf{A}$ to give:

$$\mathbf{A^{-1}Ap} = \mathbf{A^{-1}q} \tag{2.11}$$

and since $\mathbf{A^{-1}A}$ is the identity matrix we can simplify to:

$$\mathbf{p} = \mathbf{A^{-1}q} \tag{2.12}$$

to give an equation that will provide a solution $\mathbf{p}$ to the simultaneous equations that were originally considered as Equation 2.1. All that is required is to be able to compute the inverse of matrix $\mathbf{A}$ and then compute the product of $\mathbf{A^{-1}}$ and $\mathbf{q}$.

In the example that has so far been considered (Equation 2.1) there are two variables (x and y) and two equations or constraints. Under these conditions the matrix $\mathbf{A}$ will be square and the system of equations can be solved unless $\mathbf{A}$ is singular. Another example of a situation where $\mathbf{A}$ is square and singular is given by Equation 2.13 thus:

$$\begin{aligned} x + y &= 6 \\ 2x + 2y &= 12 \end{aligned} \tag{2.13}$$

where in this case, the determinant of the system matrix $\mathbf{A}$ is zero and $\mathbf{A}$ has rank[1] 1.

[1] The rank of a matrix is the number of independent rows (or columns).

If there are more variables than equations then the system is said to be under-determined and cannot be solved. However, an interesting situation arises when the system is over-determined; that is, when there are more equations than variables. An example of an over-determined system is given by Equation 2.14 thus:

$$\begin{aligned} x + y &= 6 \\ x - y &= 3 \\ 2x - 3y &= 0 \end{aligned} \tag{2.14}$$

In a situation such as Equation 2.14 there are two possibilities. Firstly, that two of the equations are in fact mathematically identical (in this trivial condition one of the identical pair of equations can be ignored and the system can be solved in a similar manner to that described for Equation 2.1). Secondly, Equation 2.14 may, strictly speaking, not be a system of simultaneous equations at all; that is, the three equations are not simultaneously true. In this interesting situation it is possible to ignore one of the three equations and solve for the other two (however, the solution would be different depending upon which one of the three equations was ignored); a better solution can often be achieved by a process of optimisation using all three equations. In fact, this condition can often arise in engineering applications, where (for example) the scalars on the right hand side of Equation 2.14 are obtained by measurement and are subject to noise. Finding values for x and y that exactly satisfy none of the three equations but which approximately satisfy all three may be preferable. Using the matrix notation of Equation 2.9 to represent Equation 2.14 where $\mathbf{p}$ is a 2×1 column matrix, $\mathbf{A}$ is a 3×2 matrix, and $\mathbf{q}$ is a 3×1 column matrix, we need to solve for $\mathbf{p}$ so that the result of the product $\mathbf{Ap}$ is as close as possible to $\mathbf{q}$ (specifically, we need to minimise the root-mean-squared error between the 3×1 vectors $\mathbf{Ap}$ and $\mathbf{q}$). The difficulty, however, is that $\mathbf{A}$ is not a square matrix and therefore has no inverse. However, approximation methods can be used to compute the pseudoinverse of a non-square matrix and this procedure is denoted by the $+$ superscript symbol in this book, for example $\mathbf{A}^+$. It is therefore possible to solve Equation 2.14 thus:

$$\mathbf{p} = \mathbf{A}^+\mathbf{q}. \tag{2.15}$$

It can be shown that Equation 2.15 produces a least-squares solution for $\mathbf{Ap} - \mathbf{q}$.

2.3 Function Approximation

A linear transform is a type of function; a rule f that associates with each element in one set with one and only one element in another set (Anton, 1994). If f associates the element b with the element a then we write $b = f(a)$. A function may, for example, associate a four-dimensional real value $\mathbb{R}^4$ with a three-dimensional real value $\mathbb{R}^3$, in which case we say that f is a transformation from $\mathbb{R}^4$ to $\mathbb{R}^3$, or that f maps $\mathbb{R}^4$ into $\mathbb{R}^3$. We denote this by writing $f: \mathbb{R}^4 \rightarrow \mathbb{R}^3$.

Linear algebra can be used to find mappings between one set of data and another and this is sometimes called function approximation. In the following example we will

consider linear transforms of the type $f : \mathbb{R} \rightarrow \mathbb{R}$. Imagine that we suspect a relationship exists between the variable x and the variable y and that we collect N examples of the corresponding pairs (x_i, y_i) for $i = [1, N]$. Let us now suppose that the f may take the form $y = ax + b$ where a and b are scalars. We can now construct the linear transform:

$$\mathbf{y} = \mathbf{aX} \tag{2.16}$$

where $\mathbf{a}$ is a 1×2 row matrix containing the scalars a and b (which we would like to find), $\mathbf{y}$ is a $1 \times N$ row matrix containing the N values of y, and $\mathbf{X}$ is a $2 \times N$ matrix. The first row of $\mathbf{X}$ contains the N values of x and the second row of $\mathbf{X}$ contains 1s thus:

$$\begin{bmatrix} y_1 & y_2 & \cdots & y_N \end{bmatrix} = \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_N \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{2.17}$$

If we can compute the pseudoinverse[2] of $\mathbf{X}$ then we can find estimates for the scalars a and b thus:

$$\mathbf{a} = \mathbf{yX}^{+}. \tag{2.18}$$

Equation 2.18 produces a least-squares solution for $\mathbf{aX} - \mathbf{y}$. Note that $\mathbf{a}$ defines a linear transform between $\mathbf{y}$ and $\mathbf{X}$. However, the entries of $\mathbf{a}$ also define our fitted relationship $y = ax + b$. It is possible to use a similar process to fit nonlinear relationships between our variables x and y. For example, if we want to fit the data with the nonlinear relationship $y = ax^2 + bx + c$ then we construct a linear system as in Equation 2.16, but in this case $\mathbf{a}$ is a 1×3 row matrix containing the scalars a, b and c, and $\mathbf{X}$ is a $3 \times N$ matrix whose ith column contains the terms x_i^2, x_i and 1.

Function approximations of the form $f: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ are particularly useful in colour science. Imagine, for example, that we have a set of N camera *RGB* response values and we wish to find a linear transform between the corresponding N known *XYZ* tristimulus values. Formally, if we define $\mathbf{T}$ as the $3 \times N$ matrix of tristimulus values and $\mathbf{C}$ as the $3 \times N$ matrix of camera response values then we seek a transformation of type $\mathbb{R}^3 \rightarrow \mathbb{R}^3$. Thus, we require to find the coefficients a_{11} to a_{33} for the following system of equations:

$$\begin{aligned} X &= a_{1,1}R + a_{1,2}G + a_{1,3}B \\ Y &= a_{2,1}R + a_{2,2}G + a_{2,3}B \\ Z &= a_{3,1}R + a_{3,2}G + a_{3,3}B \end{aligned} \tag{2.19}$$

In matrix algebra we express this as:

$$\mathbf{T} = \mathbf{AC} \tag{2.20}$$

where $\mathbf{A}$ is a 3×3 system matrix that defines the transform. At least three examples of the transform are required. However, since the XYZ values are most likely obtained

[2] Although MATLAB® includes a function for computing the pseudoinverse of a nonsquare matrix it will be seen later that there are better ways for solving equations such as 2.17.

by measurement and subject to noise it is likely that better estimates for the matrix **A** are possible if $N > 3$. If $N = 3$ then we obtain **A** from $\mathbf{TC}^{-1}$ whereas if $N > 3$ we can obtain **A** from $\mathbf{TC}^{+}$.

If we would like to consider a nonlinear relationship between **C** and **T**, such as:

$$\begin{aligned} X &= a_{1,1}R + a_{1,2}G + a_{1,3}B + a_{1,4}R^2 + a_{1,5}G^2 + a_{1,6}B^2 \\ Y &= a_{2,1}R + a_{2,2}G + a_{2,3}B + a_{2,4}R^2 + a_{2,5}G^2 + a_{2,6}B^2 \\ Z &= a_{3,1}R + a_{3,2}G + a_{3,3}B + a_{3,4}R^2 + a_{3,5}G^2 + a_{3,6}B^2 \end{aligned} \tag{2.21}$$

this can again be expressed in linear algebra form:

$$\mathbf{T} = \mathbf{AD}. \tag{2.22}$$

In Equation 2.21, **T** is the $3 \times N$ matrix of tristimulus values, **A** is the 3×6 standard matrix, and **D** is a $6 \times N$ matrix whose ith column contains the terms $[R \quad G \quad B \quad R^2 \quad G^2 \quad B^2]$. In this case N must be 6 or greater.

3

A Short Introduction to MATLAB®

MATLAB® (*Mat*rix *Lab*oratory) is a numerical computing environment and fourth-generation[1] programming language. Versions of MATLAB® are available for almost all major computing platforms. The material in this book was produced and tested on a version (R2011_a) running on the Mac OS X environment but has also been tested on a Microsoft Windows environment. In addition to the MATLAB® system itself, Mathworks[2] offers sets of *Toolboxes*, containing MATLAB® functions for solving a number of important types of problems.

The simplest way to execute MATLAB® code is to type it in the Command Window, which is one of the elements of the MATLAB® Desktop. For all but the simplest of tasks, however, it is usually preferable to save a sequence of commands in a text file (with the .m file extension) usually created and edited using the MATLAB® Editor. It is also possible to create functions that extend the commands available and the toolbox presented in this book is in fact a set of functions.

3.1 Matrices

The key to using MATLAB® successfully lies in the user's ability to conceptualise data as square, rectangular, columnar and row matrices. Whereas most programming languages are based on ordinary algebra, whereby a symbol or name is used to represent a single numerical quantity, in MATLAB® every name is assumed to be a matrix and the names can be manipulated via the rules of matrix arithmetic.

[1] A fourth-generation programming language is a programming language or programming environment designed with a specific purpose in mind.
[2] www.mathworks.com/

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.

In order to illustrate the use of MATLAB® let us consider the problem defined by Equation 2.9 and examine how this problem could be solved using MATLAB®. To enter a 2×2 matrix called **A**, we can enter the following at the $\gg$ prompt of the Command Window.

```
A = [1 1; 1 -1];
```

Notice that the entries of **A** were entered within square brackets and that the rows were separated by a semi-colon. The final semi-colon at the end of the line is optional; if it is not present MATLAB® will echo the values of **A** to the Command Window when the Return key is pressed. To enter a 2×1 column matrix **q** we would write:

```
q = [6; 3];
```

In order to solve Equation 2.9 we need to multiple the column matrix **q** by the inverse of matrix **A**. A major feature of MATLAB® is that it provides many built-in high-level functions and the function `inv` returns the inverse of a square matrix. Thus typing:

```
inv(A)

ans =

0.5000       0.5000
0.5000      -0.5000
```

computes the inverse of **A** and displays it in the Command Window. At any time it is possible to type the command `whos` and this displays a list of the current variables and their dimensions; thus:

```
whos

  Name    Size    Bytes    Class

  A       2x2        32    double array
  ans     2x2        32    double array
  q       2x1        16    double array

Grand total is 10 elements using 80 bytes
```

Notice that since we did not assign the output of the `inv` command to any variable it was automatically assigned to the variable **ans**.

We can now compute the solution to Equation 2.9 by typing:

```
p = inv(A)*q

p = 4.5000
     1.5000
```

which gives, of course, the same result as that achieved by the substitution method that was briefly described in Chapter 2.

In the following sections, the basic properties of MATLAB® are briefly introduced and some of its useful functions are described.

3.2 Matrix Operations

Matrices may be added, subtracted and multiplied using the conventional symbols $+$, $-$ and $*$. Matrices may also be easily augmented thus:

```
M = [1 1; 1 -1];
M = [M; M]

M =
1     1
1    -1
1     1
1    -1
```

places a copy of the original matrix **M** below the current contents of **M** and hence produces a 4×2 matrix, whereas the command:

```
M = [M M]
```

would produce a 2×4 matrix. Two matrices can be joined, side by side, provided that they have the same number of rows. They can also be joined one on top of the other, provided they have the same number of columns.

The colon operator is a special feature in MATLAB® for constructing row vectors of evenly spaced values. The statement:

```
x = 1:6
x = 1   2   3   4   5   6
```

generates a row matrix **x** containing the integers from 1 to 6.

Individual elements of a matrix may be referenced by specifying their indices within parentheses. Thus:

```
M = [1 1; 1 -1];
x = M(1,1)

x = 1

y = M(2,:)

y = 1 -1
```

In the preceding statement the colon operator selects the whole row. Similarly, `y = M(:,2)` would select the whole of the second column. It is also possible to edit a single entry in a matrix by addressing it directly. Thus:

```
M = [1 1; 1 -1];
M(1,1) = 2;
M = 2  1
    1 -1
```

Notice that whole rows or columns can easily be selected and manipulated (copied, printed, operated upon). For example, the statement `M(1,:) = 2*M(1,:)` would double every entry in the first row of the matrix **M**.

MATLAB® provides many functions for entering and manipulating special matrices including `linspace, ones, eye, inv, length, diag`, and `size`. As an example, the command:

```
w = linspace(400,700,31);
```

generates a 31-dimensional row matrix containing the values 400, 410, 420, . . . , 700 evenly spaced between 400 and 700[3]. In order to discover the operation of other functions use the `help` command, for example, `help eye`.

A matrix may be easily transposed in MATLAB® using the ' operator. Thus, if **x** is a 3×1 column matrix then the command:

```
x = x';
```

will convert **x** into a 1×3 row matrix.

[3] Though w = [400:10:700] is equivalent in this case.

When entering matrices in MATLAB® names must begin with a letter, contain only letters or digits, and although they may be entered of any length, MATLAB® only retains the first 19 characters. While it is tempting to use names such as CameraResponses it is safer to choose instead camera_responses or to use the convention from the C language of capitalising only second and subsequent words, as in cameraResponses. During a MATLAB® session the values of all defined variables are stored in the workspace. The user may save the current list of variables and their associated values using the `save` command. The command `save myfile.mat`, for example, will save the workspace as a special MATLAB® file and this may be recovered during a new session using the `load` command.

The command `clear` will remove all user-defined variables from the workspace. The format of displayed numbers during a session can be changed using the `format` command. Finally, it is important to note that MATLAB® is case sensitive.

3.3 Solving Linear Systems

The `inv` operator has already been introduced for computing the inverse of a matrix when solving a pair of simultaneous equations. The `inv` command can only be used to invert matrices that are square. For nonsquare matrices MATLAB® provides the `pinv` command that computes a pseudoinverse. Whereas the inverse of a matrix $\mathbf{A}$ is denoted by the symbol $\mathbf{A^{-1}}$,the pseudoinverse is denoted by the symbol $\mathbf{A}^{+}$.

However, it is usually more efficient and accurate (Borse, 1997) to solve systems of simultaneous equations using Gaussian elimination or, equivalently, by using MATLAB's® slash (/) or backslash (\) division. Thus if $\mathbf{X}$ is the matrix that is required then $\mathbf{AX} = \mathbf{B}$ is solved by `X = A\B` and $\mathbf{XA} = \mathbf{B}$ is solved by `X = B/A`.

Thus Equation 2.9 may be solved by:

```
p = A\q;
```

For some linear systems solving using the `inv` and `pinv` commands will generate identical results to the backslash operator. However, for other linear systems the matrix that needs to be inverted is ill-behaved. Consider the systems illustrated by the upper diagrams in Figure 3.1. The top-left diagram shows a system of two simultaneous equations for which there is an exact solution (given by the intersection of the two lines). The equations that represent these two lines are neither contradictory nor simple multiples of each other. Whereas the top-right diagram shows two parallel lines for which there is no solution. The equations that represent these two lines are said to be inconsistent.

However, as the gradients of the two lines in the top-left diagram become more and more similar the computation of the exact solution can become difficult and can change quite markedly with small changes in the lines themselves. A set of two equations with two unknowns is termed ill-conditioned if a small change in any of the coefficients will change the solution set from unique to either infinite or empty (Borse, 1997).

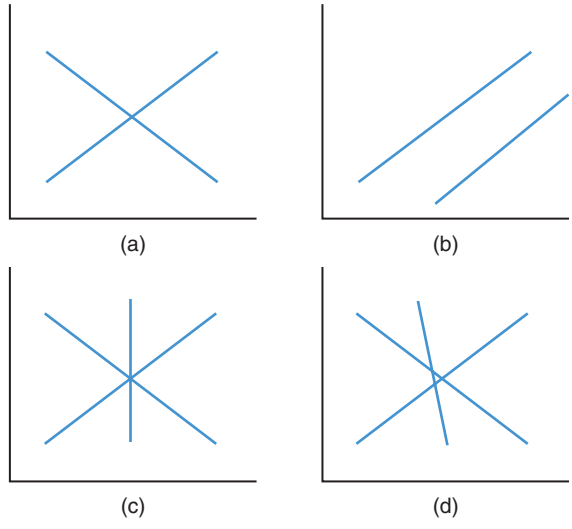

Figure 3.1 *(a) A system of two equations with an exact solution; (b) a system of two equations with no solution; (c) a system of three equations with an exact solution; (d) a system of three equations with an approximate solution.*

It is also interesting to consider the systems represented by the lower diagrams where two variables are represented by three equations (this is known as an over-determined system). The bottom-left diagram illustrates an over-determined system with an exact solution. However, there is no exact solution for the system represented by the bottom-right diagram but an approximate solution may be found. The system represented by the bottom-right diagram of Figure 3.1 is typical of many that are encountered during solutions to colorimetric problems.

Consider the following MATLAB® commands which represent how to solve a problem similar to that shown in the bottom-right diagram of Figure 3.1.

```
a = [1;  1;  2];
M = [1 -1; 1 1; 6 1];
x = M\a
```

The solution to this problem is given by $\mathbf{x} = (0.3846, -0.1026)$. However, if we multiply the top row of the system by a common factor, say 100, to yield the following related problem,

```
a = [100; 1; 2];
M = [100 -100; 1 1; 6 1];
x = M\a
```

then the solution reported is $\mathbf{x} = (0.4717, -0.5282)$. Note that if we represented these two problems graphically then they would be identical since multiplying an equation by a common factor throughout does not change it. The difference in the two solutions highlights an important property of the solution of such over-determined systems in that the solution provided by `pinv` or the backslash operator is a least-squares solution. That is, for $\mathbf{a} = \mathbf{M}\mathbf{x}$ the solution $\mathbf{x}$ is that which minimises the squares of the errors between actual values of the column matrix $\mathbf{a}$ and predicted values of $\mathbf{a}$ given $\mathbf{x}$. It is thus evident that multiplying one row of the system by a common factor will change the solution because it effectively changes the weight of that row in the solution. For the simple system considered the backslash and `pinv` operators generate identical solutions and this could be predicted in advance because the system is well conditioned. The condition number of a matrix is given by the MATLAB® command `cond`, thus:

```
M = [1 -1; 1 1; 6 1];
cond(M)
ans =4.4159
```

The condition number of a matrix $\mathbf{M}$ is defined as the product of the norm of $\mathbf{M}$ and the norm of the inverse matrix $\mathbf{M}^{-1}$ (Borse, 1997). When the condition number of a matrix is high it is especially important to use the backslash operator rather than `pinv`.

3.4 M-Files

A powerful property of MATLAB® is that it offers the user the ability to write scripts, known as M-files. Any simple text editor such as Notepad can be used to write an M-file but in later versions of MATLAB® an Integrated Development Environment (IDE) is provided with a special MATLAB® editor. Commands can be entered as a script in the same way that they would be entered into the Command Window. If the script is saved with the .m extension then the commands can be executed by simply typing the name of the script. For example, an M-file called test.m can be executed by typing `test` in the Command Window. For some scripts it can be useful to place the command `clear` as the first line in the script so that MATLAB® script is started from a clean environment. Of course, it is important to be careful to avoid using names for M-files that clash with any of MATLAB's® built-in functions or M-file functions. Comments may be placed in M-files by starting the line with the % symbol.

3.5 Using Functions in MATLAB®

Although it is possible to create quite complicated programs using combinations of M-files (since one M-file can call another) most users will at some stage wish to create their own functions. This can also be achieved using M-files. In fact, many of the toolbox functions in MATLAB® that perform some action on an arbitrary input are in fact scripts

stored as M-files. In order to see how a script can be used to generate a function the following example illustrates a function called treble that takes a single variable as input and produces three times that variable as the output:

```
function [out] = treble(in)
out = 3*in;
end
```

The text for the function *treble* should be saved in an M-file called treble.m[4]. The function is then available to the Command Window or to other M-files or functions and is simply called in any of the following ways:

```
treble(x)
y = treble(x);
[y] = treble(x);
```

The last of these formats is useful since it allows for a function to return more than one variable. Note also, that the function *treble* will operate on a single number, a row or column matrix or a matrix.

A wide variety of text books (Borse, 1997; Marchand, 1999) exist for the reader who wishes to become more familiar with MATLAB® before proceeding with the remainder of this book.

[4] Note that the final end statement in a function is strictly not required.

4

Computing CIE Tristimulus Values

4.1 Introduction

In many manufacturing industries (such as plastics, textiles, and paints) trained experts known as colourists have traditionally been responsible for the assessment of the colour appearance of the colour match (Rich, 2002). Although this approach worked well for many years, in today's fast-moving global workplace more objective methods are required. Colorimetry attempts to capture the essence of colour perception and provides an objective procedure for accurate colour matching and reproduction. Tristimulus values are the basis of colorimetry and their accurate calculation is highly desired by industry for a wide range of applications. In order to compute the tristimulus values for a surface that is defined by a set of spectral reflectance factors it is necessary to specify an illuminant and a set of colour-matching functions. The spectral reflectance factors, the relative energy of the illuminant, and the colour-matching functions must be multiplied together at each wavelength and then summed. In some cases the reflectance factors are specified at a wavelength interval that is smaller or larger than the wavelength interval of the illuminant data or the colour-matching functions. This chapter reviews methods for computing tristimulus values from spectral reflectance data and considers the use of interpolation and extrapolation where appropriate.

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.

4.2 Colour-Matching Functions

The CIE (see Section 1.3 for a brief review) originally defined the tristimulus values in terms of a mathematical integration over wavelength λ thus:

$$X = k \int_{\lambda} E(\lambda)P(\lambda)\bar{x}(\lambda)d\lambda$$

$$Y = k \int_{\lambda} E(\lambda)P(\lambda)\bar{y}(\lambda)d\lambda$$

$$Z = k \int_{\lambda} E(\lambda)P(\lambda)\bar{z}(\lambda)d\lambda \tag{4.1}$$

where $E(\lambda)$ is the relative spectral power distribution of an illuminant, $\bar{x}(\lambda), \bar{y}(\lambda)$ and $\bar{z}(\lambda)$ are the colour-matching functions for the CIE 1931 or 1964 standard observers, $P(\lambda)$ is the spectral reflectance of an object surface (or point in a scene) and k is a normalising factor given by $k = 100/\int_{\lambda} E(\lambda)\bar{y}(\lambda)d\lambda$.

The integration was originally specified to be performed over the visible range of the electromagnetic spectrum. Unfortunately, analytical expressions for the colour-matching functions do not exist and so in practice it is not possible to calculate the tristimulus values according to Equation 4.1. Furthermore, it is typically the case that commercially available reflectance spectrophotometers record the reflectance spectrum $P(\lambda)$ at discrete intervals. It has therefore long been necessary to use numerical methods to approximate the integration. In 1986 the CIE adopted an alternative practice for calculating tristimulus values based upon numerical integration using wavelength intervals of 1 nm (CIE, 1986a). This leads to Equation 4.2 where the summation is carried out over the visible range of wavelengths as before:

$$X = k \sum_{\lambda=360}^{\lambda=830} E(\lambda)\bar{x}(\lambda)P(\lambda)$$

$$Y = k \sum_{\lambda=360}^{\lambda=830} E(\lambda)\bar{y}(\lambda)P(\lambda)$$

$$Z = k \sum_{\lambda=360}^{\lambda=830} E(\lambda)\bar{z}(\lambda)P(\lambda) \tag{4.2}$$

The colour-matching functions (for both the 1931 and 1964 standard observers) are provided with seven significant figures by the CIE in tabular form at 1-nm intervals between the wavelengths 360 nm and 830 nm in CIE Publication number S2 (CIE, 1986a). These are the official sets of colour-matching functions recommended by the CIE. However, for most practical applications it is suggested that an abridged set of colour-matching functions may be used at 5-nm intervals over the range 380 nm to 780 nm and these are provided in CIE Publication number 15.2 (CIE, 1986b).

The 1931 colour-matching functions are recommended whenever correlation with visual colour matching of fields of angular subtense between approximately 1° and 4° at the retina of the observer is desired. For larger angular subtenses the 1964 colour-matching functions are recommended.

The use of the 5 nm colour-matching functions necessitates that the spectral reflectance factors be known at intervals of 5 nm. For practical applications, the required data are often not available in an appropriate format because of abridgement (measurement at intervals greater than 5 nm) or truncation (lack of data at very short or very long wavelengths). Many modern reflectance spectrophotometers, for example, provide data at 10 nm intervals in the range 400 nm to 700 nm. For situations where the spectral data are abridged or truncated, the CIE recommends the use of interpolation and extrapolation respectively (CIE, 1986b) so that the measurement data are converted to 1 nm or 5 nm intervals. Note, however, that in practice, tables at 10 nm are frequently used.

If reflectance data are available at 5-nm intervals then the most accurate method to compute tristimulus values is to use the 5-nm colour-matching and illuminant data. Even if reflectance data are available at 10- or 20-nm intervals the 5-nm data can be used if interpolation methods are applied to the reflectance data. Another situation where interpolation methods may be important is where a user is computing tristimulus values for a specific non-CIE illuminant. A problem that the CIE has so far failed to solve is the disparity between illuminant spectral power distributions and light sources that serve to correspond to these illuminants. This is a particular problem with CIE illuminant D65 where although there are many lamps that are used as D65 simulators there is, in fact, no light source that replicates illuminant D65 exactly (Xu, Luo and Rigg, 2003). A practical solution to this problem is to measure the spectral power distribution of the actual light source used in a specific viewing cabinet, for example, and to use these measurements as the illuminant data in the colorimetric equations (Equation 4.2). This approach is sensible; unfortunately many commercial spectroradiometers provide radiance measurements at wavelength intervals of 4, 5, or 10 nm. Interpolation methods may be necessary to obtain the illuminant data at 5-nm intervals. Interpolation methods are now briefly discussed before alternative methods for computing tristimulus values are described.

4.3 Interpolation Methods

In the absence of a theoretical equation representing the data, a reasonable estimate of values between known values is found by making assumptions about the data set and applying mathematical principles based on those assumptions (CIE, 2005).

A straight line can be drawn to fit exactly through any two points, a parabola through any three points. More generally, an nth-degree polynomial can pass through any $n + 1$ points. Thus, if there are measurements of reflectance $P(\lambda)$ at n wavelengths an arbitrary $(n - 1)$th-degree polynomial,

$$P(\lambda) = a_1\lambda^{n-1} + a_2\lambda^{n-2} + \cdots + a_{n-1}\lambda + a_n \tag{4.3}$$

with n coefficients can be specified by the n independent relations. A method for finding the coefficients a_1 to a_n can be envisaged if we consider Equation 4.3 at each of the

n wavelengths simultaneously to give a linear system. For example, if we wish to fit reflectance data at four wavelengths λ_1 to λ_4, we can write:

$$\begin{aligned} P(\lambda_1) &= a_1\lambda_1^3 + a_2\lambda_1^2 + a_3\lambda_1 + a_4 \\ P(\lambda_2) &= a_1\lambda_2^3 + a_2\lambda_2^2 + a_3\lambda_2 + a_4 \\ P(\lambda_3) &= a_1\lambda_3^3 + a_2\lambda_3^2 + a_3\lambda_3 + a_4 \\ P(\lambda_4) &= a_1\lambda_4^3 + a_2\lambda_4^2 + a_3\lambda_4 + a_4 \end{aligned} \tag{4.4}$$

which represents four simultaneous equations and four unknowns. In terms of linear algebra (see Chapter 2) Equation 4.4 can be efficiently represented by Equation 4.5:

$$\mathbf{Ap} = \mathbf{q} \tag{4.5}$$

where $\mathbf{q}$ is a 4×1 column matrix of reflectance values, $\mathbf{p}$ is a 4×1 column matrix containing the coefficients a_1 to a_4, and $\mathbf{A}$ is a special 4×4 matrix known as the Vandermonde matrix. For the third-order polynomial example in Equation 4.4 the Vandermonde matrix would be constructed with the entries thus:

$$\begin{pmatrix} \lambda_1^3 & \lambda_1^2 & \lambda_1 & 1 \\ \lambda_2^3 & \lambda_2^2 & \lambda_2 & 1 \\ \lambda_3^3 & \lambda_3^2 & \lambda_3 & 1 \\ \lambda_4^3 & \lambda_4^2 & \lambda_4 & 1 \end{pmatrix}$$

The polynomial in Equation 4.3 is referred to as the Lagrange polynomial. It is trivial to solve Equation 4.5 for the coefficients $\mathbf{p}$ using MATLAB's® backslash operator. So, for example, imagine we have the following reflectance factors 0.40, 0.45, 0.48 and 0.49 at the wavelengths 400, 410, 420 and 430 nm respectively. The following MATLAB® code can be used to fit the Lagrange polynomial:

```
q = [0.40; 0.45; 0.48; 0.49];
A = [400^3 400^2 400 1; 410^3 410^2 410 1; 420^3 420^2 420 1;
     430^3 430^2 430 1];
p = A\q
```

which results[1] in:

```
p = -0.0000
    -0.0001
     0.0860
   -18.0000
```

[1] MATLAB® does its internal arithmetic in IEEE floating point precision using approximately 16 decimal digits, but the default display is only five decimal digits. Typing `format long` will reveal greater precision.

The performance of the polynomial can be tested at the four wavelengths using the command `A*p` which reveals that the predicted values are:

```
A*p
ans = 0.399999999999999
      0.449999999999999
      0.480000000000004
      0.489999999999998.
```

Notice that the fit is good but far from perfect. The problem is that matrix **A** is ill-conditioned. On the other hand, if we scale the wavelength values so that 400, 410, 420 and 430 are represented by 0, 0.25, 0.5 and 1.0, a more accurate fit is possible using the same commands. Thus:

```
clear
format long
q   = [0.40; 0.45; 0.48; 0.49];
AA  = [0^3 0^2 0 1; 0.25^3 0.25^2 0.25 1; 0.5^30.5^2 0.5 1;
       0.75^3 0.75^2 0.75];
p   = AA\q
ans = AA*p

p =  0.000000000000001
    -0.160000000000002
     0.240000000000000
     0.400000000000000

ans = 0.400000000000000
      0.450000000000000
      0.480000000000000
      0.490000000000000
```

Note that the solution to the equation is correct to 15 decimal places. The condition[2] number of matrix A is 7.64×10^{12} and for matrix AA is 1.63×10^{2}.

Alternatively, MATLAB® provides the functions `polyfit` and `polyval` that automatically fit and use polynomials respectively. Thus the following code fits a fifteenth-order Lagrange polynomial to 16 reflectance values representing measurements at 20 nm

[2] The MATLAB® command `cond` returns the so-called 2-norm condition number (the ratio of the largest singular value of the matrix to the smallest). Large condition numbers indicate a nearly singular matrix.

intervals in the range 400 nm to 700 nm:

```
r = [9.37 12.32 12.46 9.51 5.92 4.33 4.29 3.88 4.51
     10.92 27.50 49.67 69.59 81.73 88.19 86.05];
w = linspace(400,700,16);
[P,S] = polyfit(w,r,15);
x = linspace(400,700,301);
y = polyval(P,x);
```

The effect of the preceding code is illustrated in Figure 4.1(a) for a typical reflectance curve. The circle symbols show the original reflectance data at 20-nm intervals and the 20-nm data have been interpolated to yield the fitted line. The solid line shows the polynomial fit illustrated at intervals of 1 nm. During the execution of the `polyfit` command MATLAB® showed the warning:

```
about to call polyfit
Warning: Polynomial is badly conditioned. Remove repeated
     data points or try centering and scaling as described
     in HELP POLYFIT.
```

which indicates a problem with the solution of the matrix equation. This is the same problem of matrix conditioning described earlier. More information on this process can

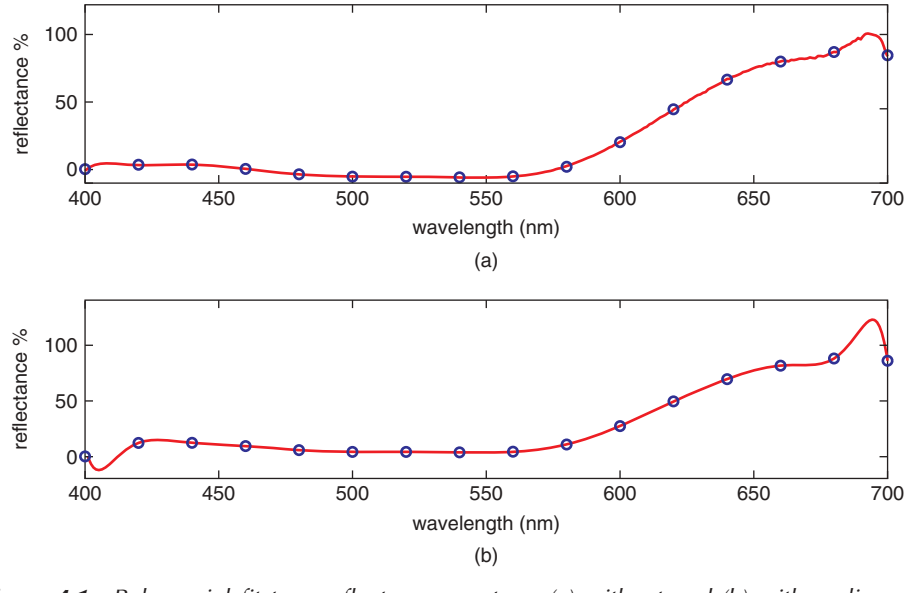

Figure 4.1 *Polynomial fit to a reflectance spectrum (a) without and (b) with scaling and centring.*

be found by typing `help polyfit`. The command `polyfit` provides automatic scaling and centring and the following code demonstrates how this is used. The effect of the normalising procedure can be seen by the solid line in Figure 4.1(b).

```
       r = [9.37 12.32 12.46 9.51 5.92 4.33 4.29 3.88 4.51
            10.92 27.50 49.67 69.59 81.73 88.19 86.05];
       w = linspace(400,700,16);
[P,S,mu] = polyfit(w,r,15);
       x = linspace(400,700,301);
       y = polyval(P,x,[],mu);
```

Notice, however, that even using the normalised wavelength scale[3] although the Lagrangian polynomial fits the 20-nm data exactly it would make quite poor estimates (particularly towards the two ends of the spectrum) if it was used to interpolate the 20-nm data to yield 10 nm intervals. The use of a Lagrangian polynomial in this way is not the best way to interpolate the spectral data; it may lead to wide excursions between the points and, moreover, a limited number of digits may lead to inaccuracies in the computed values (CIE, 2005).

This problem may alternatively be addressed by using a family of polynomials (each of which fits a relatively small number of points) rather than trying to fit the whole spectrum with a single polynomial. Repeated application of a low-order polynomial to a restricted range of the data is an example of a 'piecewise method'. CIE Publication Number 15.2 recommends that interpolation of spectra be carried out using a third-degree polynomial from neighbouring data within twice the measurement interval (CIE, 1986b). This means that a Lagrange interpolation formula should be used for four data points (two either side of the point to be interpolated). A later CIE publication (CIE, 2004) recommends one of four interpolation methods: (1) the third-order polynomial interpolation (Lagrange) from the four neighbouring data points around the point to be interpolated, or (2) a cubic spline interpolation formula, or (3) a fifth-order polynomial interpolation from the six neighbouring data points around the point to be interpolated, or (4) a Sprague interpolation.

The Sprague interpolation has been recommended[4] followed a study by a CIE working group (CIE, 2005). This essentially involves fitting a fifth-order polynomial between two points P_i and P_{i+1} but constraining the polynomial so that the gradients and curvature are consistent with the points P_{i-2}, P_{i-1}, and P_{i+2}, P_{i+3}. It turns out that the coefficients of the polynomial $a_0 - a_5$ are a linear transform of the points $P_{i-2} - P_{i+3}$ and this transform is provided by the CIE (CIE, 2005), thus:

$$a_0 = P_i$$
$$a_1 = (2P_{i-2} - 16P_{i-1} + 16P_{i+1} - 2P_{i+2})/24$$

[3] The variable x is replaced by $(x\text{-}mu(1))/mu(2)$.
[4] Strictly, the method is only recommended when the spectral data are uniformly spaced but this is almost always the situation for the vast majority of practical purposes.

$$
\begin{aligned}
a_2 &= (-P_{i-2} - 16P_{i-1} - 30P_i + 16P_{i+1} - P_{i+2})/24 \\
a_3 &= (-9P_{i-2} + 39P_{i-1} + 70P_i + 66P_{i+1} - 33P_{i+2} + 7P_{i+3})/24 \\
a_4 &= (13P_{i-2} - 64P_{i-1} + 126P_i - 124P_{i+1} + 61P_{i+2} - 12P_{i+3})/24 \\
a_5 &= (-5P_{i-2} + 25P_{i-1} - 50P_i + 50P_{i+1} - 25P_{i+2} + 5P_{i+3})/24
\end{aligned} \tag{4.6}
$$

where the coefficients define the polynomial:

$$
f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 \tag{4.7}
$$

The variable $x = (\lambda - \lambda_i)/(\lambda_{i+1} - \lambda_i)$ so that when $\lambda = \lambda_i$ and $\lambda = \lambda_{i+1}$ the value of x is 0 and 1 respectively. The polynomial should only be used to estimate reflectance values between λ_i and λ_{i+1}. The following code has been used to generate Figure 4.2 where the 20 nm data have to interpolated to generate data at intervals of 1 nm.

```
r = [9.37 12.32 12.46 9.51 5.92 4.33 4.29 3.88 4.51
     10.92 27.50 49.67 69.59 81.73 88.19 86.05];
w = linspace(400,700,16);
plot(w,r,'bo');
for i=3:13
a0 = r(i);
a1 = (2*r(i-2)-16*r(i-1)+16*r(i+1)-2*r(i+2))/24;
a2 = (-r(i-2)+16*r(i-1)-30*r(i)+16*r(i+1)-r(i+2))/24;
a3 = (-9*r(i-2)+39*r(i-1)-70*r(i)+66*r(i+1)-
     33*r(i+2)+7*r(i+3))/24;
a4 = (13*r(i-2)-64*r(i-1)+126*r(i)-
     124*r(i+1)+61*r(i+2)-12*r(i+3))/24;
a5 = (-5*r(i-2)+25*r(i-1)-50*r(i)+50*r(i+1)-
     25*r(i+2)+5*r(i+3))/24;
wave = 400+(i-1)*20;
w = linspace(wave+2,wave+18,19);
x = linspace(0.1,0.9,19);
y = a0+a1*x+a2*x.^2+a3*x.^3+
     a4*x.^4+a5*x.^5;
hold on
plot(w,y,'ro')
end
```

Of course, the Sprague interpolation method cannot be used for the data points at the ends of the spectrum; this is a limitation of all piecewise fitting methods. The CIE recommends a method for extrapolating four additional points, two at shorter wavelength intervals than available and two at longer wavelength intervals than available. The method is available in the CIE Technical Report (CIE, 2005) and is not described here. However, the following code segment implements the method and is used to generate Figure 4.3. The method recommended by the CIE is to use extrapolation to generate additional points at each end of the spectrum to allow the application of Sprague interpolation to the full range of data originally available.

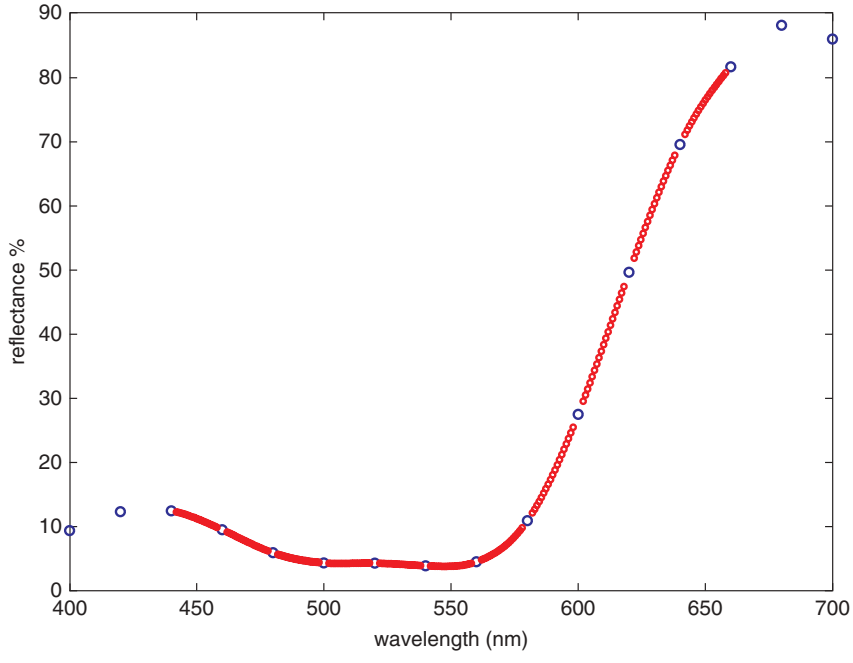

Figure 4.2 *Sprague interpolation of 20-nm data to intervals of 1 nm.*

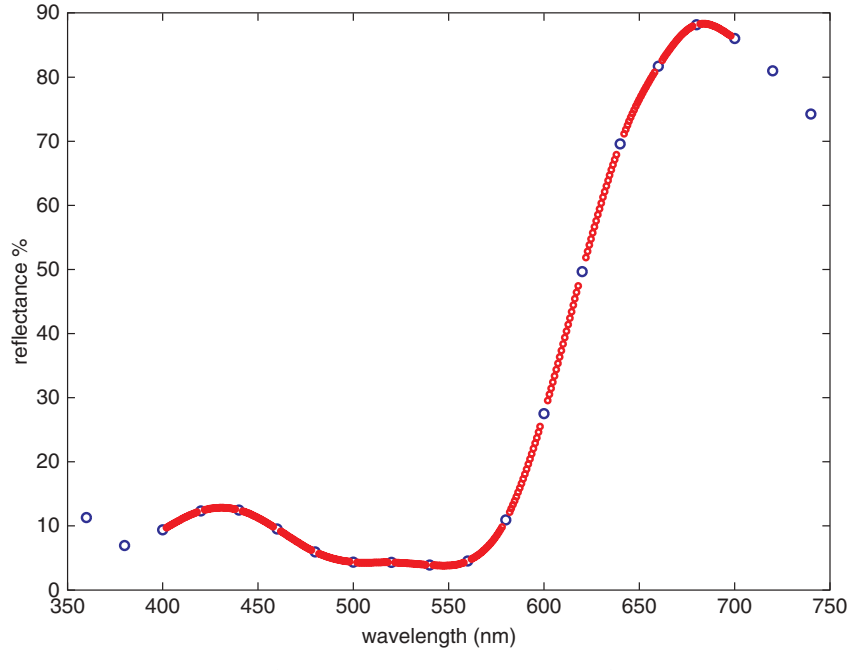

Figure 4.3 *Demonstration of CIE-recommended practice of extrapolation to allow the application of Sprague interpolation to the full data range available (in the figure, the two extrapolated values at each end of the spectrum are shown).*

The toolbox function sprague has been written to allow interpolation and should be used for reflectance spectra or illuminant data.

```
% ================================================
% *** FUNCTION sprague
% ***
% *** Interpolates an n by w matrix of spectra
% *** where n is the number of spectra
% *** f is an interpolation factor
% *** e.g. if f=2 the sampling rate is doubled
% ================================================

function [p] = sprague(spectra,f)
if (f<2 | ((f-floor(f))>0))
    disp('invalid f value - premature termination');
    p = 0;
return;
end

% set the parameters
c = [884 -1960 3033 -2648 1080 -180;
    508 -540 488 -367 144 -24;
    -24 144 -367 488 -540 508;
    -180 1080 -2648 3033 -1960 884];

[numSpectra lengthSpectra] = size(spectra);

for i=1:numSpectra
% select a spectrum
r = spectra(i,:);
% add the extra start and end points
k  = c(1,:);
p1 = (k*(r(1:6))')/209;
k  = c(2,:);
p2 = (k*(r(1:6))')/209;
k  = c(3,:);
p3 = (k*(r(11:16))')/209;
k  = c(4,:);
p4 = (k*(r(11:16))')/209;

                  r = [p1 p2 r p3 p4];
                  N = lengthSpectra+4;

p  = zeros(numSpectra,f*(N-5)+1);
xx = linspace(1/f,1-1/f,f-1);

for j=3:N-3
a0    = r(j);
```

```
a1    = (2*r(j-2)-16*r(j-1)+16*r(j+1)
                -2*r(j+2))/24;
a2    = (-r(j-2)+16*r(j-1)-30*r(j)+16*r(j+1)
                -r(j+2))/24;
a3    = (-9*r(j-2)+39*r(j-1)-70*r(j)+66*r(j+1)
                -33*r(j+2)+7*r(j+3))/24;
a4    = (13*r(j-2)-64*r(j-1)+126*r(j)
                -124*r(j+1)+61*r(j+2)-12*r(j+3))/24;
a5    = (-5*r(j-2)+25*r(j-1)-50*r(j)+50*r(j+1)
                -25*r(j+2)+5*r(j+3))/24;
y     = a0+a1*xx+a2*xx.^2+a3*xx.^3+a4*xx.^4
                +a5*xx.^5;
index = j-2;
p(i,(index-1)*f+1) = r(j);
p(i,(index-1)*f+1+1:(index-1)*f+1+f-1) = y;
end
p(i,f*(N-5)+1) = r(N-2);
end

end
% ======================================================
% *** END FUNCTION sprague
% ======================================================
```

The `sprague` function takes two arguments; the first is an $n \times w$ matrix of n spectra, each defined at w wavelengths, and the second is a factor f that indicates how much the spectra are interpolated (f must be an integer of 2 or greater). If f is 2 the sampling rate of the spectra is doubled, if f is 3 the sampling rate is trebled, and so on. An example of a valid function call is:

```
p = sprague(spectra,2);
```

The most commonly used values of f are likely to be 2 (which would convert 20-nm data to 10-nm data and 10-nm data to 5-nm data) and 10 (which would convert 10-nm data to 1-nm data). Note that the function checks to see whether the value of f that is passed is an integer >2. Note also the line:

```
p = zeros(numSpectra,f*(N-5)+1);
```

This pre-allocates sufficient memory for the matrix. This line is strictly not required. However, if memory is not pre-allocated in this way then MATLAB® will need to call a memory allocation routine many times in the `for` loop; this method of allocating memory 'on the fly' is not recommended and can result in much slower code.

4.4 Extrapolation Methods

A further problem that can occur when computing tristimulus values is that many reflectance spectrophotometers provide reflectance data in the range 400–700 nm and yet the 5 nm colour-matching functions, for example, are defined over 380–780 nm. It is possible to extrapolate the reflectance data and the method recommended by CIE is to extend the reflectance data by using the most extreme value as an estimate of all values beyond that extreme (CIE, 1986b). So, for example, if the calculation is being carried out at 10-nm intervals and the reflectance data are in the range 400–700 nm then values of reflectance at 710, 720, . . . , 780 nm are set equal to the value at 700 nm.

A similar procedure applies to the shorter wavelength. Although it could be suggested that more accurate extrapolation methods could be employed it should be remembered that extrapolation is far more dangerous than interpolation. Also, the fact that the colour-matching functions have very small values below 400 nm and above 700 nm means that the errors that result from the CIE method are generally very small and the risk of using sophisticated extrapolation techniques is not justified.

4.5 Correction for Spectral Bandpass

Reflectance spectrophotometers that, for example, provide measurement data at 10-nm intervals do not measure at single wavelengths (delta functions, for example, at 400, 410, 420, . . . , 700 nm). Rather the measurement that is reported at wavelength λ_i integrates energy between λ_{i-1} and λ_{i+1}. Consequently, the measured reflectance data P' need to be corrected to obtain the true reflectance data P. Stearns and Stearns (1988) and Venable (1989) have proposed methods for spectral bandpass correction and the Stearns-and-Stearns correction is given by Equation 4.8:

$$P_i = -\alpha P'_{i-1} + (1 + 2\alpha)P'_i - \alpha P'_{i+1} \tag{4.8}$$

where α is equal to 0.083 and where, if the wavelength being corrected is the first or last one in the sequence, Equation 4.9 is used:

$$P_i = (1 + \alpha)P'_i - \alpha P'_{i+1} \tag{4.9}$$

It is important to know, therefore, whether the spectral reflectance values from a given reflectance spectrophotometer have been corrected for spectral bandpass by the operating software in order that the correct tables of weights are used. The bandpass correction is not built-in to the CIE 1- and 5-nm data and therefore if these sets of colour-matching functions are used then it is important that the reflectance data are corrected for bandpass dependence.

```
% ================================================
% *** FUNCTION cband
% ***
% *** Stearns-Stearns bandpass correction
% *** operates on matrix P of dimensions n by m
% *** n spectra each at m wavelengths
% ================================================
```

```
function [p] = cband(p)

a = 0.083;
dim = size(p);
n = dim(1);

% the first and last wavelengths
p(1,:) = (1+a)*p(1,:) - a*p(2,:);
p(n,:) = (1+a)*p(n,:) - a*p(n-1,:);
% all the other wavelengths
for i=2:n-1
p(i,:) = -a*p(i-1,:) +(1+2*a)*p(i,:)
         -a*p(i+1,:);
end

end
% ================================================
% *** END FUNCTION cband
% ================================================
```

4.6 Tristimulus Values

Some practitioners prefer to use weighting tables where the terms $E(\lambda)\bar{x}(\lambda), E(\lambda)\bar{y}(\lambda)$ and $E(\lambda)\bar{z}(\lambda)$ as used in Equation 4.2 are pre-computed at each wavelength interval. These weighting tables can be computed from the CIE colour-matching functions and illuminants. The benefit to the user in using these tables is that Equation 4.2 can be replaced by Equation 4.10:

$$X = \sum_{\lambda=360}^{\lambda=830} W_X(\lambda)P(\lambda)$$

$$Y = \sum_{\lambda=360}^{\lambda=830} W_Y(\lambda)P(\lambda)$$

$$Z = \sum_{\lambda=360}^{\lambda=830} W_Z(\lambda)P(\lambda) \tag{4.10}$$

where the weight vectors $W_X(\lambda), W_Y(\lambda)$ and $W_Z(\lambda)$ also include the normalising constant k from Equation 4.2. The CIE recommends that such tables of weighting factors should be provided for the full range of wavelengths, 360–830 nm, so that they may be used for any degree of truncation by adding the weights at the unmeasured wavelengths to those at the extreme measured wavelengths.

A set of useful weights are provided by the *American Society for Testing and Materials* in E308-01 (ASTM, 2001). The E308-01 tables are provided only for the range

of wavelengths 360–780 nm but are suitable for most practical applications. They are provided at 10- and 20-nm intervals. The fact that the E308-01 tables are abridged to intervals of 10 and 20 nm has resulted in them being probably the most widely used method for computing tristimulus values because the 10-nm data, in particular, are suitable for direct use with reflectance data obtained from most reflectance spectrophotometers without interpolation. The ASTM publication provides the data in two main tables: ASTM Tables 5 should be used with reflectance data that have been corrected for the spectral bandpass of the instrument whereas ASTM Tables 6 have the spectral-bandpass correction built in and should be used with reflectance data that have not been corrected. The majority of reflectance spectrophotometers that are commercially available do not correct for the spectral bandpass of the instrument.

The ASTM tables of weights are available in hard copy or electronic form from the ASTM website www.astm.org.

The ASTM weights (Table 5) are provided in this toolbox as a mat file. The command `load weights.mat` loads the values of ASTM Table 5 and the `whos` command would reveal the following variables:

```
 Name    Size   Bytes  Class

a_31     43x3   1032   double
a_64     43x3   1032   double
c_31     43x3   1032   double
c_64     43x3   1032   double
d50_31   43x3   1032   double
d50_64   43x3   1032   double
d55_31   43x3   1032   double
d55_64   43x3   1032   double
d65_31   43x3   1032   double
d65_64   43x3   1032   double
d75_31   43x3   1032   double
d75_64   43x3   1032   double
f11_31   43x3   1032   double
f11_64   43x3   1032   double
f2_31    43x3   1032   double
f2_64    43x3   1032   double
f7_31    43x3   1032   double
f7_64    43x3   1032   double
```

The file weights.mat contains weights that represent the 1964 and 1931 colour-matching functions for the CIE illuminants, A, C, D50, D55, D65, D75, F2, F7 and F11. The white points of the illuminants are given (ASTM, 2001) by Table 1.

The function `r2xyz` computes *XYZ* tristimulus values using ASTM Table 5 colour-matching functions from the ASTM standard (ASTM, 2001). The reflectance data should be in the range [0, 1] rather than in per cent format and must be sampled at 10-nm intervals. Since ASTM Table 5 is used it is assumed that the reflectance data have been corrected for the spectral bandpass properties of the spectrophotometer that was used for

their measurement. A typical call for a reflectance spectrum sampled at 10-nm intervals in the range 400–700 nm would be:

```
[xyz] = r2xyz(p,400,700,'d65_64');
```

where **p** represents a 31 × 1 matrix. The second and third arguments relate to the shortest and longest wavelength available in the reflectance data. The fourth argument specifies the illuminant and observer combination to be used. The code for `r2xyz` is shown below:

```
% ====================================================
% *** FUNCTION r2xyz
% ***
% *** function [xyz] =r2xyz(p, startw, endw, obs)
% *** computes XYZ from reflectance p
% *** p is an n by w matrix of n spectra
% *** e.g. set obs to 'd65_64 for D65 and 1964
% *** the startw and endw variables denote first and
% *** last wavelengths (e.g. 400 and 700) for p which
% *** must be 10 nm data in the range 360–780
% ====================================================

function [xyz] = r2xyz(p, startlam, endlam, obs)

if((endw>780) | (startw<360) | (rem(endw,10)~=0)
    | (rem(start,10)~=0))
disp('startw and endw must be divisible by 10')
disp('wavelength range must be 360–780 or less');
return;
end
load weights.mat
% weights.mat contains the tables of weights
if strcmp('a_64',obs)
   cie = a_64;
elseif strcmp('a_31', obs)
   cie = a_31;
elseif strcmp('c_64', obs)
   cie = c_64;
elseif strcmp('c_31', obs)
   cie = c_31;
elseif strcmp('d50_64', obs)
   cie = d50_64;
elseif strcmp('d50_31', obs)
   cie = d50_31;
elseif strcmp('d55_64', obs)
   cie = d55_64;
```

```
elseif strcmp('d55_31', obs)
   cie = d55_31;
elseif strcmp('d65_64', obs)
   cie = d65_64;
elseif strcmp('d65_31', obs)
   cie = d65_31;
elseif strcmp('d75_64', obs)
   cie = d75_64;
elseif strcmp('d75_31', obs)
   cie = d75_31;
elseif strcmp('f2_64', obs)
   cie = f2_64;
elseif strcmp('f2_31', obs)
   cie = f2_31;
elseif strcmp('f7_64', obs)
   cie = f7_64;
elseif strcmp('f7_31', obs)
   cie = f7_31;
elseif strcmp('f11_64', obs)
   cie = f11_64;
elseif strcmp('f11_31', obs)
   cie = f11_31;
else
   disp('unknown option obs');
   disp('use d65_64 for D65 and 1964 observer');
return;
end

% check dimensions of p
dim = size(p);
N = ((endw-startw)/10 + 1);
if (length(p) ~= N)
   disp('dimensions of p inconsistent');
return;
end

% deal with possible truncation of reflectance
i = (startw - 360)/10 + 1;
if (i>1)
   cie(i,:) = cie(i,:) + sum(cie(1:i-1,:));
end
j = i + N - 1;
if (j<43)
   cie(j,:) = cie(j,:) + sum(cie(j+1:43,:));
end
cie = cie(i:j,:);
```

```
% the main calculation
xyz = (p*cie)*100/sum(cie(:,2));

end
% ================================================
% *** END FUNCTION r2xyz
% ================================================
```

Following some basic checks on the arguments to the function the issue of truncation is addressed in the `r2xyz` code. If the reflectance data are only available at 400 nm and higher, for example, then the values of the weights at wavelengths lower than 400 nm are added to the value of the weights at 400 nm. Summation of the product of the weights and the reflectance data is then performed at 400 nm and upwards. This is equivalent to extending the reflectance data below 400 nm using the value of reflectance at 400 nm. A similar process is carried out for the upper wavelength that is available. This procedure is in accordance with the CIE recommendation for dealing with truncated reflectance data.

4.7 Chromaticity Diagrams

Chromaticity coordinates are computed from tristimulus values according to Equations 4.11:

$$\begin{aligned} x &= X/(X+Y+Z) \\ y &= Y/(X+Y+Z) \\ z &= Z/(X+Y+Z) \end{aligned} \tag{4.11}$$

Given that $x + y + z = 1$, it is usual to quote just two of the coordinates (by convention, x and y are selected) in addition to one of the tristimulus values (Y is selected because, for the 1931 observer, it is equivalent to luminance which is expressed in units of cd/m^2). It is sometimes useful to be able to compute the tristimulus values from an xyY specification and this can be accomplished using Equations 4.12:

$$\begin{aligned} X &= (x/y)Y \\ Y &= (y/y) \\ Z &= ((1-x-y)/y)Y\,. \end{aligned} \tag{4.12}$$

The chromaticity coordinates provide a useful representation especially for additive colour-reproduction devices where the gamut of the device is defined by the polygon whose vertices are the chromaticities of the device primaries. Note, however, that such device gamuts are three-dimensional so, for example, for a colour monitor it will not be possible to obtain the full range of chromaticities at all luminance levels (Morovič, 2002).

The chromaticities of the spectral locus of a chromaticity diagram can be obtained directly from the tables of weights as shown in Equation 4.13:

$$x = W_X(\lambda)/W_X(\lambda) + W_Y(\lambda) + W_Z(\lambda))$$
$$y = W_Y(\lambda)/(W_X(\lambda) + W_Y(\lambda) + W_Z(\lambda)) \quad (4.13)$$

The weights in Equations 4.13 can be replaced by the colour-matching functions and in this case the chromaticity coordinates are computed for the appropriate observer and for the equal-energy illuminant (illuminant E). However, in order to generate a smooth spectral locus then 5-nm intervals or less are required.

It is sometimes desirable to generate a colour representation of the chromaticity diagram. Such a diagram would be useful for teaching and illustration purposes and also for technical papers and other publications. Of course, it must be understood that any such representation will be an illustration at best[5].

MATLAB's® `patch` command is useful for producing representations of various colour spaces. For example, Figure 4.4, a version of an RGB Maxwell triangle is produced using just the following code:

```
xy=[0 0; 1 0; 0.5 0.866];
col=[1 0 0; 0 1 0; 0 0 1];
patch('Vertices',xy, 'Faces',[1:size(xy,1)],
        'EdgeColor','none','FaceVertexCData',
        col,'FaceColor','interp');
axis off
```

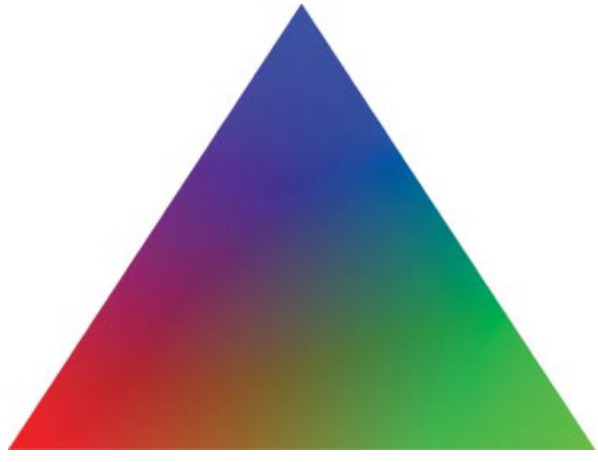

Figure 4.4 *Representation of an RGB colour-mixing triangle using the patch command.*

[5] The gamut of a digital display or printing system is smaller than the chromaticity gamut of all physically realisable colours. Therefore any chromaticity diagram that shows colours towards the spectral locus is, at best, an interesting representation rather than a colorimetrically accurate reproduction.

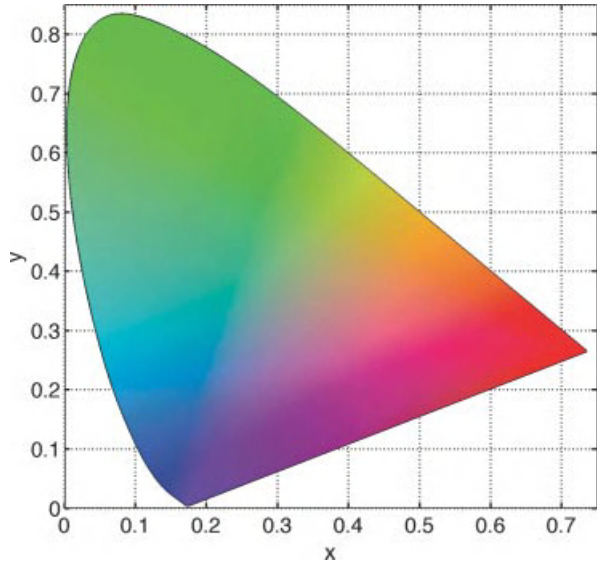

Figure 4.5 *Representation of the CIE chromaticity diagram using the toolbox function,* `cieplot`.

The toolbox function `cieplot` utilises the `patch` command and the sRGB colour space to produce a colour representation of the chromaticity diagram (see Figure 4.5). The MATLAB® code for `cieplot` is shown below:

```
% ======================================================
% *** FUNCTION cieplot
% ***
% *** function =cieplot()
% *** colour representation of chromaticity diagram
% ======================================================
function [] = cieplot()
% load spectral locus xy values at 1nm intervals
load locus.mat
plot(locus(:,1),locus(:,2),'k','LineWidth',2)
grid on; hold on
axis([0.0 0.75 0.0 0.85])
xlabel('x')
ylabel('y')
% plot the non-spectral locus
plot([locus(1,1) locus(end,1)], [locus(1,2)
     locus(end,2)],'k','LineWidth',2)
% chromaticity coordinates of spectrum locus
x = [0.175596 0.172787 0.170806 0.170085 0.160343
     0.146958 0.139149 0.133536 0.126688 0.115830
```

```
     0.109616 0.099146 0.091310 0.078130 0.068717
     0.054675 0.040763 0.027497 0.016270 0.008169
     0.004876 0.003983 0.003859 0.004646 0.007988
     0.013870 0.022244 0.027273 0.032820 0.038851
     0.045327 0.052175 0.059323 0.066713 0.074299
     0.089937 0.114155 0.138695 0.154714 0.192865
     0.229607 0.265760 0.301588 0.337346 0.373083
     0.408717 0.444043 0.478755 0.512467 0.544767
     0.575132 0.602914 0.627018 0.648215 0.665746
     0.680061 0.691487 0.700589 0.707901 0.714015
     0.719017 0.723016 0.734674 ]';
y = [0.005295 0.004800 0.005472 0.005976 0.014496
     0.026643 0.035211 0.042704 0.053441 0.073601
     0.086866 0.112037 0.132737 0.170464 0.200773
     0.254155 0.317049 0.387997 0.463035 0.538504
     0.587196 0.610526 0.654897 0.675970 0.715407
     0.750246 0.779682 0.792153 0.802971 0.812059
     0.819430 0.825200 0.829460 0.832306 0.833833
     0.833316 0.826231 0.814796 0.805884 0.781648
     0.754347 0.724342 0.692326 0.658867 0.624470
     0.589626 0.554734 0.520222 0.486611 0.454454
     0.424252 0.396516 0.372510 0.351413 0.334028
     0.319765 0.308359 0.299317 0.292044 0.285945
     0.280951 0.276964 0.265326 ]';
N = length(x);
i = 1;
e = 1/3;
steps = 25;
xy4rgb = zeros(N*steps*4,5,'double');
for w = 1:N % wavelength
   w2 = mod(w,N)+1;
   a1 = atan2(y(w)-e,x(w)-e);      % start angle
   a2 = atan2(y(w2)-e,x(w2)-e);    % end angle
   r1 = ((x(w)-e)^2 + (y(w)- e)^2)^0.5;% start radius
   r2 = ((x(w2)-e)^2 + (y(w2)-e)^2)^0.5; % end radius
   for c = 1:steps % colourfulness
      % patch polygon
      xyz(1,1) = e+r1*cos(a1)*c/steps;
      xyz(1,2) = e+r1*sin(a1)*c/steps;
      xyz(1,3) = 1 - xyz(1,1) - xyz(1,2);
      xyz(2,1) = e+r1*cos(a1)*(c-1)/steps;
      xyz(2,2) = e+r1*sin(a1)*(c-1)/steps;
      xyz(2,3) = 1 - xyz(2,1) - xyz(2,2);
      xyz(3,1) = e+r2*cos(a2)*(c-1)/steps;
      xyz(3,2) = e+r2*sin(a2)*(c-1)/steps;
      xyz(3,3) = 1 - xyz(3,1) - xyz(3,2);
      xyz(4,1) = e+r2*cos(a2)*c/steps;
      xyz(4,2) = e+r2*sin(a2)*c/steps;
      xyz(4,3) = 1 - xyz(4,1) - xyz(4,2);
```

```
      % compute sRGB for vertices
      rgb = xyz2srgb(xyz');
      % store the results
      xy4rgb(i:i+3,1:2) = xyz(:,1:2);
      xy4rgb(i:i+3,3:5) = rgb';
      i = i + 4;
   end
end
[rows cols] = size(xy4rgb);
f = [1 2 3 4];
v = zeros(4,3,'double');
for i = 1:4:rows
    v(:,1:2) = xy4rgb(i:i+3,1:2);
    patch('Vertices',v, 'Faces',f,'EdgeColor','none',
          'FaceVertexCData',xy4rgb(i:i+3,3:5),
          'FaceColor','interp')
    end
end
function [rgb] = xyz2srgb(xyz)
M = [3.2406 -1.5372 -0.4986; -0.9689 1.8758 0.0415;
     0.0557 -0.2040 1.0570];
[rows cols ] = size(xyz);
rgb = M*xyz;
for c = 1:cols
   for ch = 1:3
      if rgb(ch,c) <= 0.0031308
         rgb(ch,c) = 12.92*rgb(ch,c);
      else
         rgb(ch,c) = 1.055*(rgb(ch,c)^(1.0/2.4)) -0.055;
      end
      % clip RGB
      if rgb(ch,c) < 0
         rgb(ch,c) = 0;
      elseif rgb(ch,c) > 1
         rgb(ch,c) = 1;
      end
   end
end

end
% ======================================================
% *** END FUNCTION cieplot
% ======================================================
```

5

CIELAB and Colour Difference

5.1 Introduction

Although the system of colour specification introduced in 1931 by the CIE and augmented in 1964 has served the colour industry well, there remain a number of problems. One of the main problems is that in terms of visual perception it is very nonuniform. Equal changes in x, y or Y at various points in the colour space do not correspond to perceived differences of equal magnitude. Most attempts to develop more uniform spaces sought to find linear or nonlinear transforms of the tristimulus values or chromaticity coordinates to give a more uniform colour space. In 1976 the CIE recommended two new colour spaces for general use (CIE, 1986b): CIE $L^*a^*b^*$ and CIE $L^*u^*v^*$ also known as CIELAB and CIELUV respectively. CIELUV tends to be used to specify the colours of lights and other self-luminous sources whereas CIELAB tends to be used for the specification of surface colours. It is possible to compute a colour difference for two stimuli in CIELAB space by calculating the Euclidean distance in the space between the two points that represent the stimuli in the space (Equation 1.7). The CIELAB colour-difference formula has been used extensively for quality control in industry but its application is limited because although CIELAB space is more perceptually uniform than the tristimulus space on which it is based, it is still far from being perfectly uniform. The consequence of this is that for equal perceptual colour differences between pairs of samples, the values of CIELAB colour difference ΔE^*_{ab} computed between points representing the pairs in CIELAB space can vary by an order of magnitude. Since 1976, attempts to generate better metrics for the prediction of colour differences have concentrated on finding more sophisticated measures of distance. A summary of the developments is not given in detail here (see Smith, 1997; Berns, 2000; Luo, 2002a) but the formulae for the three key developments, CMC(l:c), CIE94, and CIEDE2000 are given. The CMC equation, developed in the early 1980s, was a key development in colour science and became a standard in certain countries and

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.

industries (Clarke, McDonald and Rigg, 1984). It was never adopted by the CIE as a standard, however, and by the early 1990s there was some concern that the formula might be overly complex and that its predictions might be poor in certain areas of colour space. The CIE recommended the CIE94 equation (Berns, 1993) for use before a concerted effort was made to develop a new formula. CIEDE2000 (Luo, Cui and Rigg, 2001) was developed following collaboration between scientists working in several countries and has now been adopted as a CIE recommendation for the prediction of small colour differences.

5.2 CIELAB and CIELUV Colour Space

Between 1940 and 1976 a great number of colour spaces, transformations of XYZ, were proposed as uniform colour spaces. Some of these, such as HunterLab and ANLAB were quite successful but in 1976 the CIE agreed upon two transformations that led to CIELAB and CIELUV.

The formulae (Hunt and Pointer, 2011) for computing CIELAB coordinates are given in Equations 5.1:

$$\begin{aligned} L^* &= 116 f(Y/Y_n) - 16 \\ a^* &= 500[f(X/X_n) - f(Y/Y_n)] \\ b^* &= 200[f(Y/Y_n) - f(Z/Z_n)] \end{aligned} \tag{5.1}$$

Where:
$f(I) = I^{1/3}$ if $I > (6/29)^3$ and
$f(I) = (841/108)(I) + 4/29$ if $I \leq (6/29)^3$.

In Equations 5.1, X_n, Y_n and Z_n are the tristimulus values for the appropriately chosen reference white. For surface colours the values of X_n, Y_n and Z_n are usually computed for the perfect reflecting diffuser and are therefore equivalent to the tristimulus values of the illuminant itself. Since white surfaces tend to look chromatically neutral under an illumination to which the visual system is adapted, the values of X_n, Y_n and Z_n are sometimes referred to as the neutral point. The axes L^*, a^* and b^* form a rectangular or Cartesian coordinate space where L^* represents Lightness, a^* represents redness-greenness, and b^* represents yellowness-blueness. Sometimes it is useful to represent colour stimuli in a cylindrical space and for these purposes it is possible to compute the polar coordinates C^*_{ab} and h_{ab} as shown in Equations 5.2 and 5.3:

$$C^*_{ab} = \sqrt{a^{*2} + b^{*2}} \tag{5.2}$$

$$h_{ab} = \tan^{-1}\left(b^*/a^*\right)(180/\pi) \tag{5.3}$$

where the term $180/\pi$ is necessary to convert the output of the inverse tan function from radians to degrees[1]. The polar coordinates are useful since the differences in the term C^*_{ab} can be correlated with differences in perceived chroma and differences in the hue term h_{ab} can be correlated with differences in perceived hue. Equations 5.2 and 5.3 can easily be inverted (Green, 2002a) thus:

[1] The first edition of this book included functions to convert between polar and Cartesian spaces. However, these are no longer needed since MATLAB® includes the functions `cart2pol` and `pol2cart` which do exactly this task.

$$a^* = C^*_{ab} \cos(h_{ab}\pi/180) \tag{5.4}$$
$$b^* = C^*_{ab} \sin(h_{ab}\pi/180). \tag{5.5}$$

The function `xyz2lab` computes CIELAB values from tristimulus values. A typical call would be:

```
[lab] = xyz2lab(xyz,'d65_64');
```

where **xyz** represents an $n \times 3$ matrix. The code for `xyz2lab` is shown below:

```
% ================================================
% *** FUNCTION xyz2lab
% ***
% *** function [lab] =xyz2lab(xyz, obs, xyzw)
% *** computes LAB from XYZ
% *** xyz is an n by 3 matrix
% *** e.g. set obs to 'd65_64 for D65 and 1964
% *** set obs to 'user' to use optional argument
% *** xyzw as the white point
% ================================================

function [lab] = xyz2lab(xyz,obs,xyzw)

if (size(xyz,2)~=3)
   disp('xyz must be n by 3'); return;
end
lab = zeros(size(xyz,1),size(xyz,2));

if strcmp('a_64',obs)
    white=[111.144 100.00 35.200];
elseif strcmp('a_31', obs)
    white=[109.850 100.00 35.585];
elseif strcmp('c_64', obs)
    white=[97.285 100.00 116.145];
elseif strcmp('c_31', obs)
    white=[98.074 100.00 118.232];
elseif strcmp('d50_64', obs)
    white=[96.720 100.00 81.427];
elseif strcmp('d50_31', obs)
    white=[96.422 100.00 82.521];
elseif strcmp('d55_64', obs)
    white=[95.799 100.00 90.926];
elseif strcmp('d55_31', obs)
    white=[95.682 100.00 92.149];
elseif strcmp('d65_64', obs)
    white=[94.811 100.00 107.304];
elseif strcmp('d65_31', obs)
    white=[95.047 100.00 108.883];
elseif strcmp('d75_64', obs)
    white=[94.416 100.00 120.641];
```

```
elseif strcmp('d75_31', obs)
    white=[94.072 100.00 122.638];
elseif strcmp('f2_64', obs)
    white=[103.279 100.00 69.027];
elseif strcmp('f2_31', obs)
    white=[99.186 100.00 67.393];
elseif strcmp('f7_64', obs)
    white=[95.792 100.00 107.686];
elseif strcmp('f7_31', obs)
    white=[95.041 100.00 108.747];
elseif strcmp('f11_64', obs)
    white=[103.863 100.00 65.607];
elseif strcmp('f11_31', obs)
    white=[100.962 100.00 64.350];
elseif strcmp('user', obs)
    white=xyzw;
else
   disp('unknown option obs');
   disp('use d65_64 for D65 and 1964 observer');
return;
end

lab = zeros(size(xyz,1),3);
index = (xyz(:,2)/white(2) > 0.008856);
lab(:,1) = lab(:,1) + index.*(116*(xyz(:,2)/
       white(2)).^(1/3) - 16);
lab(:,1) = lab(:,1) + (1-index).*(903.3*(xyz(:,2)/white(2)));

fx = zeros(size(xyz,1),3);
for i=1:3
    index = (xyz(:,i)/white(i) > 0.008856);
    fx(:,i) = fx(:,i) +
      index.*(xyz(:,i)/white(i)).^(1/3);
    fx(:,i) = fx(:,i) +
      (1-index).*(7.787*(xyz(:,i)/white(i)) + 16/116);
end

lab(:,2) = 500*(fx(:,1)-fx(:,2));
lab(:,3) = 200*(fx(:,2)-fx(:,3));

end
% ================================================
% *** END FUNCTION xyz2lab
% ================================================
```

Note that use of 'user' as the observer allows the user to input a third argument. Thus:

```
[lab] = xyz2lab(xyz,'user',xyzw);
```

where **xyzw** is a 1×3 row matrix containing the tristimulus values of a custom white point.

Also note that although the n sets of tristimulus values could be processed using a for loop this would be slow and unsatisfactory when the function is being used to process large amounts of data (as is often the case when processing colour image data)[2]. The lines:

```
index = (xyz(:,2)/white(2) > 0.008856);
lab(:,1) = lab(:,1) + index.*(116*(xyz(:,2)/
       white(2)).^(1/3) - 16);
lab(:,1) = lab(:,1) + (1-index).*(903.3*(xyz(:,2)/white(2)));
```

deserve special attention. This function could be written using a for loop so that each of the n samples is calculated in turn. However, this is not the most efficient way of performing this function in MATLAB® where matrix operations are optimised. Great care should be exercised in implementing conditional if statements when using matrix operations. The above code segment illustrates a neat way to handle conditional statements that would normally appear within a for loop. An $n \times 1$ column vector of zeros and ones called **index** is first generated and is then used to direct the conditional statements. This method is used repeatedly through this book.

The white points are taken from Table 5 of the ASTM standard (ASTM, 2001) and are reproduced in Table 5.2 (see later in this chapter). Note that the ASTM standard specifies that the white points listed at the bottom of each of the tables in the standard should be used for the values of X_n, Y_n and Z_n during computations where the neutral point is required. The listed white points sometimes differ from the check sums for each of the tables because the tabulated data are rounded to 3 decimal places. The quoted white points are more accurate than the sums of the rounded data in the columns. However, a consequence of the ASTM recommendation is that the CIELAB values for a sample with unit reflectance at every wavelength may not exactly satisfy $L^* = 100$, $a^* = b^* = 0$. For example, the XYZ values using ASTM Table 5.19 (illuminant D65 and 1964 observer) for a perfectly reflecting sample are [94.809 100.00 107.307] but the ASTM white point for that illuminant/observer is [94.811 100.00 107.304]. Consequently, the ratios X/X_n, Y/Y_n and Z/Z_n for a perfectly reflecting sample are not exactly unity and the CIELAB $L^*a^*b^*$ values returned from *xyz2lab* are [100.0000 −0.0035 −0.0019].

If the tristimulus values of the neutral are known, it is possible to invert Equations 5.1 according to the following equations:

If $L^* > 7.9996$

$$Y = Y_n \left((L^* + 16)/116\right)^3 \tag{5.6}$$

else:

$$Y = Y_n L^*/903.3.$$

[2] Although `xyz2lab` does not explicitly handle image data, an $m \times n \times 3$ matrix can be converted to an $mn \times 3$ matrix using the MATLAB® command `reshape`.

If $(a^*/500 + fy)^3 > 0.008856$

$$X = X_n \left(a^*/500 + fy\right)^3$$

else:

$$X = X_n \left(a^*/500 + fy - 16/116\right)/7.787.$$

If $(fy - b^*/200)^3 > 0.008856$

$$Z = Z_n \left(fy - b^*/200\right)^3$$

else:

$$Z = Z_n \left(fy - b^*/200 - 16/116\right)/7.787.$$

and where $fy = \left(Y/Y_n\right)^{1/3}$ if $Y/Y_n > 0.008856$ else: $fy = 7.787\left(Y/Y_n\right) + 16/116$.

The function `lab2xyz` computes tristimulus values from CIELAB. A typical call would be:

```
[xyz] = lab2xyz(lab,'d65_64');
```

where **lab** represents an $n \times 3$ matrix. The full code for `lab2xyz` is not shown here (because much of it is identical to the `xyz2lab` code). However, shown below is the key code segment that computes the tristimulus values:

```
% compute Y
index = (((lab(:,1)+16)/116).^3 > 0.008856);
xyz(:,2) = xyz(:,2) + index.*(white(2)*
     ((lab(:,1)+16)/116).^3);
xyz(:,2) = xyz(:,2) + (1-index).*
(white(2)*lab(:,1)/903.3);

% compute fy for use later
fy = xyz(:,2)/white(2);
index = (fy > 0.008856);
fy = zeros(size(lab,1),1);
fy = fy + (index).*(xyz(:,2)/white(2)).^(1/3);
fy = fy + (1-index).*(7.787*xyz(:,2)/white(2)
+ 16/116);

% compute X
index = ((lab(:,2)/500 + fy).^3 > 0.008856);
xyz(:,1) = xyz(:,1) + (index).*(white(1)*
     (lab(:,2)/500 + fy).^3);
xyz(:,1) = xyz(:,1) +
(1-index).*(white(1)*((lab(:,2)/500 + fy)
- 16/116)/7.787);
```

```
% compute Z
index = ((fy - lab(:,3)/200).^3 > 0.008856);
xyz(:,3) = xyz(:,3) + (index).*(white(3)*(fy
- lab(:,3)/200).^3);
xyz(:,3) = xyz(:,3) + (1-index).*(white(3)*((fy
- lab(:,3)/200) - 16/116)/7.787);
```

The equations for computing CIELUV coordinates are given as Equations 5.7:

If $Y/Y_n > 0.008856$

$$L^* = 116\left(Y/Y_n\right)^{1/3} - 16 \tag{5.7}$$

else:

$$L^* = 903.3\left(Y/Y_n\right),$$

and:

$$u^* = 13L^*\left(u' - u'_n\right)$$
$$v^* = 13L^*\left(v' - v'_n\right)$$

where u' and v' are the coordinates of the so-called uniform chromaticity space, CIE 1976 UCS, which is a linear transform of the more usual xy chromaticity space, thus:

$$\begin{aligned} u' &= 4X/(X + 15Y + 3Z) \\ v' &= 9Y/(X + 15Y + 3Z). \end{aligned} \tag{5.8}$$

The subscript n in Equations 5.7 again refers to the neutral point. It is, of course, also possible to compute polar coordinates for CIELUV thus:

$$C^*_{uv} = \sqrt{u^{*2} + v^{*2}} \tag{5.9}$$
$$h_{uv} = \tan^{-1}\left(v^*/u^*\right)(180/\pi). \tag{5.10}$$

The function `xyz2luv` computes CIELUV values from XYZ tristimulus values. A typical call would be:

```
[luv,up,vp] = xyz2luv(xyz,'d65_64');
```

where **xyz** represents an $n \times 3$ matrix. Note that the function returns an $n \times 3$ matrix **luv** containing the CIELUV values but that u' and v' are also returned, each as an $n \times 1$ column matrix[3]. The full code for `xyz2luv` is not shown here (because much of it is

[3] In MATLAB® whenever multiple outputs are returned from a function, typing the function on its own thus, `xyz2luv(xyz,'d65_64')` will output only the first of these arguments (in this case the vector **luv**) and assign it to the variable **ans**.

identical to the xyz2lab code). However, shown below is the key code segment that computes CIELUV and u' and v' values:

```
% compute u‘ v’ for sample
up = 4*xyz(:,1)./(xyz(:,1) + 15*xyz(:,2) + 3*xyz(:,3));
vp = 9*xyz(:,2)./(xyz(:,1) + 15*xyz(:,2) + 3*xyz(:,3));
% compute u‘ v’ for white
upw = 4*white(1)/(white(1) + 15*white(2) + 3*white(3));
vpw = 9*white(2)/(white(1) + 15*white(2) + 3*white(3));
index = (xyz(:,2)/white(2) > 0.008856);
luv(:,1) = luv(:,1) + index.*(116*(xyz(:,2)/white(2)).^(1/3) - 16);
luv(:,1) = luv(:,1) + (1-index).*(903.3*(xyz(:,2)/white(2)));
luv(:,2) = 13*luv(:,1).*(up - upw);
luv(:,3) = 13*luv(:,1).*(vp - vpw);
```

Whereas CIELAB was recommended for use with surface colours, CIELUV was recommended for use with self-luminous colours (surface colours are sometimes referred to as related colours since we rarely see a surface in isolation but rather as part of a scene). One of the reasons for this is that the CIELUV space retains a chromaticity diagram which is derived by plotting u^* against v^*. An approximately uniform chromaticity space is useful since the additive mixtures of two stimuli all lie upon the straight line in chromaticity space between the points that represent the two stimuli. However, in the last couple of decades CIELAB has become almost exclusively used for colour specification, and the vast majority of work on the prediction of colour difference and colour appearance has been based upon CIELAB. It has been noted that there seems no reason to use CIELUV over CIELAB (Fairchild, 1998).

5.2.1 A Representation of CIELAB Using MATLAB®

A function cielabplot has been developed to create a MATLAB® figure representation of CIELAB colour space. The problem of accurate colorimetric representation of the space is recognised but not reconciled by this code. The purpose of the code is simply to produce an aesthetically pleasing representation of the a^*–b^* plane of CIELAB colour space that could be used, for example, in presentations and reports. The code relies heavily on the MATLAB® function patch which is explained in a little more detail in Chapter 8.

```
% ================================================
% *** FUNCTION cielabplot
% ***
% *** function cielabplot
% *** creates a CIELAB figure representation
% *** see also cieplot
% ================================================
plot([0 0],[-60 60],‘k’,‘LineWidth’,2)
```

```
hold on
plot([-60 60],[0 0],'k','LineWidth',2)
axis([-60 60 -60 60])
gr = [0.7 0.7 0.7];
r = [.9 0 0];
g = [0 .9 0];
y = [.9 .9 0];
bl = [0 0 .9];
index = 0;
% first quadrant
index = index+1;
a = 50;
b = 0;
ab(index,:) = [a b];
col(index,:) = r;
for i = 5:5:85
    index = index+1;
    a = cos(i*pi/180)*50;
    b = sin(i*pi/180)*50;
    ab(index,:) = [a b];
    c = (a*r+(50-a)*y)/50;
    col(index,:) = c;
end
index = index+1;
a = 0;
b = 50;
ab(index,:) = [a b];
col(index,:) = y;
% grey
index = index+1;



a = 0;
b = 0;
ab(index,:) = [a b];
col(index,:) = gr;

patch('Vertices',ab, 'Faces',[1:size(ab,1)],
       'EdgeColor','none','FaceVertexCData',col,
       'FaceColor','interp')
clear ab;

index = 0;
% second quadrant
index = index+1;
a = 0;
b = 50;
ab(index,:) = [a b];
```

```
col(index,:) = y;

for i=95:5:175
    index = index+1;
    a = cos(i*pi/180)*50;
    b = sin(i*pi/180)*50;
    ab(index,:) = [a b];
    c=(b*y+(50-b)*g)/50;
    col(index,:) = c;
end
index = index+1;
a = -50;
b = 0;
ab(index,:) = [a b];
col(index,:) = g;
% grey
index = index+1;
a = 0;
b = 0;
ab(index,:) = [a b];
col(index,:) = gr;

patch('Vertices',ab, 'Faces',[1:size(ab,1)],
       'EdgeColor','none','FaceVertexCData',col,
       'FaceColor','interp')
clear ab;
index=0;
% third quadrant
index = index+1;
a = -50;
b = 0;
ab(index,:) = [a b];
col(index,:) = g;
for i=185:5:265
    index = index+1;
    a = cos(i*pi/180)*50;
    b = sin(i*pi/180)*50;
    ab(index,:) = [a b];
    c=(-b*bl+(50+b)*g)/50;
    col(index,:) = c;
end
index = index+1;
a = 0;
b = -50;
ab(index,:) = [a b];
col(index,:) = bl;
% grey
index = index+1;
a = 0;
```

```
b = 0;
ab(index,:) = [a b];
col(index,:) = gr;

patch('Vertices',ab, 'Faces',[1:size(ab,1)],
       'EdgeColor','none','FaceVertexCData',col,
       'FaceColor','interp')
clear ab;
index = 0;
% fourth quadrant
index = index+1;
a = 0;
b = -50;
ab(index,:) = [a b];
col(index,:) = bl;
for i=275:5:355
    index = index+1;
    a = cos(i*pi/180)*50;
    b = sin(i*pi/180)*50;
    ab(index,:) = [a b];
    c = (a*r+(50-a)*bl)/50;
    col(index,:) = c;
end
index = index+1;
a = 50;
b = 0;
ab(index,:) = [a b];
col(index,:) = r;
% grey
index = index+1;
a = 0;
b = 0;
ab(index,:) = [a b];
col(index,:) = gr;

patch('Vertices',ab, 'Faces',[1:size(ab,1)],
       'EdgeColor','none','FaceVertexCData',
       col, 'FaceColor','interp')
clear ab;
plot([0 0],[-60 60],'k','LineWidth',2)
plot([-60 60],[0 0],'k','LineWidth',2)

% ================================================
% *** END FUNCTION cielabplot
% ================================================
```

Figure 5.1 shows the result of running the `cielabplot` function.

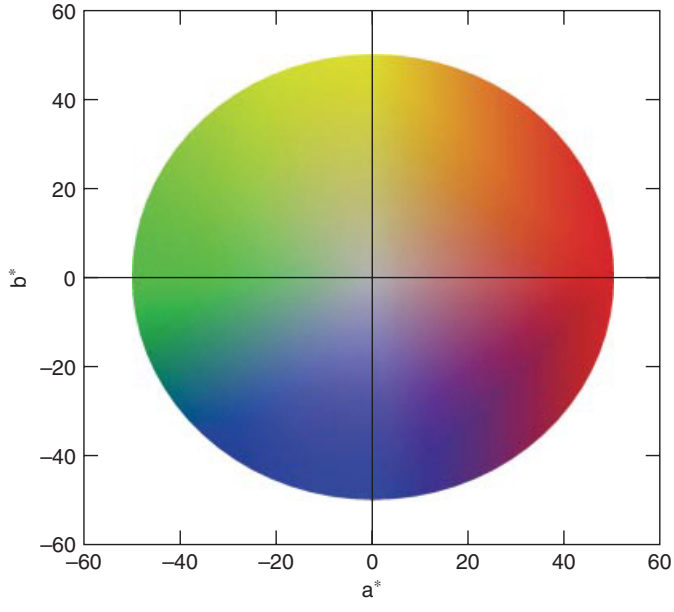

Figure 5.1 *Representation of CIELAB colour space using the toolbox function* ***`cielabplot`***.

5.3 CIELAB Colour Difference

The CIELAB space has become popular largely because of the associated colour-difference metric (Equation 5.11) that is computed as the Euclidean distance between two points in CIELAB space:

$$\Delta E^*_{ab} = \sqrt{(\Delta L^*)^2 + (\Delta a^*)^2 + (\Delta b^*)^2} \tag{5.11}$$

where:

$$\begin{aligned} \Delta L^* &= L^*_T - L^*_S \\ \Delta a^* &= a^*_T - a^*_S \\ \Delta b^* &= b^*_T - b^*_S \end{aligned}$$

and the subscripts refer to the standard S and the trial T. In industrial applications of colour difference it is common that one of the samples is a standard and the other is a sample or trial that is supposed to be a visual match to the standard.

An idea of the size of ΔE^*_{ab} units can be gained by considering that the difference between a perfect white ($L^* = 100$, $a^* = b^* = 0$) and a perfect black ($L^* = a^* = b^* = 0$) is 100 ΔE^*_{ab} units whereas industrial tolerances are usually about 1.0 ΔE^*_{ab} or CIELAB units.

If we imagine ΔE^*_{ab} to be computed from polar coordinates then we could write an equivalent equation in terms of ΔL^*, ΔC^*_{ab} and ΔH^*_{ab} thus:

$$\Delta E^*_{ab} = \sqrt{(\Delta L^*)^2 + \left(\Delta C^*_{ab}\right)^2 + \left(\Delta H^*_{ab}\right)^2} \tag{5.12}$$

where ΔH^*_{ab} is a difference in hue that is both commensurate with the other variables of CIELAB colour difference and orthogonal to both ΔL^* and ΔC^*_{ab}. Thus, whereas the other terms are computed as simple differences (ΔC^*_{ab} is simply the difference between C^*_{ab} of the standard and the trial) ΔH^*_{ab} is defined by equating Equations 5.11 and 5.12 to yield:

$$\Delta H^*_{ab} = \sqrt{\left(\Delta E^*_{ab}\right)^2 - (\Delta L^*)^2 - \left(\Delta C^*_{ab}\right)^2} \tag{5.13}$$

or simply (Seve, 1991):

$$\Delta H^*_{ab} = \sqrt{(\Delta a^*)^2 + (\Delta b^*)^2 - \left(\Delta C^*_{ab}\right)^2} \tag{5.14}$$

Sometimes an alternative method is used to compute a hue difference as given by Equation 5.13:

$$\Delta H^*_{ab} = C^*_{ab} \Delta h_{ab} \, (\pi/180) \tag{5.15}$$

where the term $\pi/180$ converts the difference in hue angle Δh_{ab} into radians. However, Smith (1997) notes that this method is only applicable for small colour differences away from the achromatic axis and Equations 5.13 and 5.14 are more generally applicable.

Normally when a colour difference is computed between two samples one of the samples is regarded as the standard and the other as the trial or batch. The components of the colour difference therefore have positive or negative sign and are computed as, for example, the chroma of the trial minus the chroma of the standard. Thus, if $\Delta C^*_{ab} > 0$ then the trial is stronger than the standard whereas if $\Delta C^*_{ab} < 0$ then the trial is weaker than the standard. Similarly, the trial can be lighter or darker than the standard depending upon the sign of the ΔL^* component. However, the definition of the hue component of colour difference as in Equation 5.12 leads to some ambiguity in the sign of ΔH^*_{ab}. By convention, it is regarded as positive if it indicates that in terms of hue the trial is anticlockwise from the standard and negative if it is clockwise.

The signs of the colour-difference components are most useful in determining colour-difference descriptors between a trial and a standard. Whereas the determination that the trial is either stronger or weaker and lighter or darker than the standard derives simply from the sign of the ΔL^* or ΔC^*_{ab}, the determination of hue difference descriptors is more complicated. The CIE recommend that two hue descriptors be assigned to any pair of samples. An illustration is given by the standard and trial samples represented in Figure 5.2.

In Figure 5.2 the chroma of the trial (represented by the circle symbol) is smaller than that of the standard (represented by the square symbol) and therefore we can deduce that the trial is weaker than the standard. To derive a hue difference descriptor we move from the standard to the trial in the hue circle and note the first two axes that are crossed. In the example illustrated by Figure 5.2, we move in a clockwise direction and pass through the red axis and then the blue axis. It is usual to then describe the trial as being redder/bluer than the standard. Note that the value of ΔH^*_{ab} would be assigned a negative sign since the trial is clockwise from the standard. Why should the trial be described as being redder/bluer? Would it not be simpler to use the closest axis and denote the trial as being redder? The answer is that the correct choice of hue descriptor is difficult

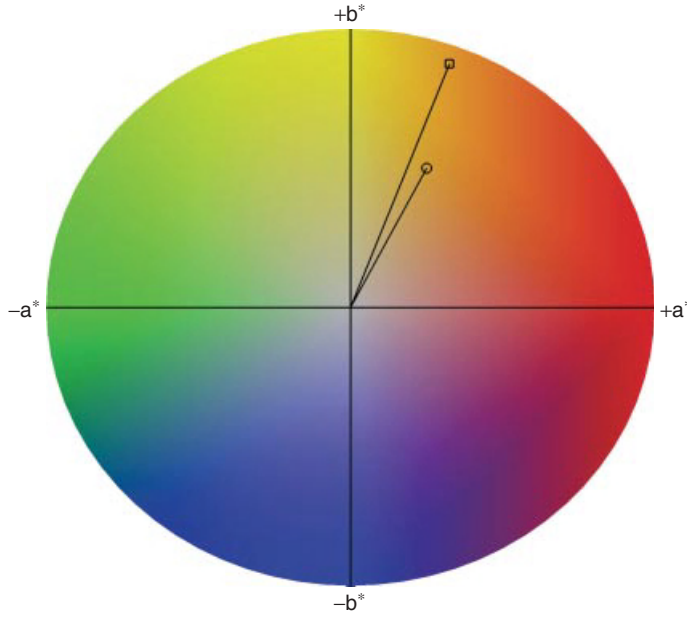

Figure 5.2 *Representation of CIELAB showing a standard (square) and trial (circle). The trial is redder than the standard even though its a* value is smaller.*

to predict in advance without knowledge of the colour appearance of the samples (the issue of colour appearance will be discussed in more detail later in Chapter 6). If, in Figure 5.2, the two samples appear yellow then it would indeed be reasonable to describe the trial as being redder than the standard. However, if the colour appearance of the two samples is essentially red, then it is not informative in terms of hue difference to describe the trial as being redder than the standard; rather, in this case we would say that the trial is bluer. For two samples in the first quadrant (that is, a^* and b^* are both positive) it is possible for the two samples to appear yellow (which is likely if the samples are close to the yellow axis) or red (which is likely if the samples are closer to the red axis) and therefore most computer programs that compute colour difference report both possible hue-difference terms (redder/bluer in the example).

Figure 5.2 emphasises why the polar coordinates C^*_{ab} and h_{ab} are often preferred to the Cartesian coordinates a^* and b^*. The fact that the trial has a smaller a^* value than the standard ($\Delta a^* < 0$) could be misinterpreted as indicating that the standard is redder than the trial and yet the opposite is true in terms of hue; the trial is redder/bluer than the standard. The possible error occurs because the dimensions of human colour perception are brightness, colourfulness and hue and these correlate with the polar coordinates lightness, chroma and hue. It can be misleading to consider differences in a^* and b^* in isolation since these confound differences in chroma and hue.

The function `cielabde` computes CIELAB colour differences between pairs of CIELAB coordinates. The function is thus called:

```
[de,dl,dc,dc] = cielabde(lab1,lab2);
```

where **lab1** and **lab2** are each $n \times 3$ matrices of the same size containing the CIELAB values for corresponding pairs. Note that the function returns four $n \times 1$ matrices containing the CIELAB colour differences and the component differences.

```
% ================================================
% *** FUNCTION cielabde
% ***
% *** function [de, dl, dc, dh] =cielabde(lab1,
       lab2)
% *** computes colour difference from CIELAB values
% *** using CIELAB formula
% *** inputs must be n by 3 matrices
% *** and contain L*, a* and b* values
% *** see also cmcde, cie94de, and cie00de
% ================================================
function [de,dl,dc,dh] = cielabde(lab1,lab2)

if (size(lab1,1)~=size(lab2,1))
   disp('inputs must be the same size'); return;
end
if (size(lab1,2)~=3 | size(lab2,2)~=3)
   disp('inputs must be n by 3'); return;
end

de = zeros(1,size(lab1,2));
dl = zeros(1,size(lab1,2));
dc = zeros(1,size(lab1,2));
dh = zeros(1,size(lab1,2));

dl = lab2(:,1)-lab1(:,1);
dc = (lab2(:,2).^2 + lab2(:,3).^2).^0.5 - (lab1(:,2).^2
       + lab1(:,3).^2).^0.5;
dh = ((lab2(:,2) - lab1(:,2)).^2 + (lab2(:,3)
       lab1(:,3)).^2 - dc.^2);
dh = (abs(dh)).^0.5;
dh = ((lab2(:,2) - lab1(:,2)).^2 + (lab2(:,3)
       lab1(:,3)).^2 - dc.^2);

% get the polarity of the dh term
dh = dh.*dhpolarity(lab1,lab2);
de = (dl.^2 + dc.^2 + dh.^2).^0.5;
end

function [p] = dhpolarity(lab1,lab2)
% function [p] =dhpolarity(lab1,lab2)
% computes polarity of hue difference
% p = +1 if the hue of lab2 is anticlockwise
% from lab1 and p =-1 otherwise
```

```
[h1,c1] = cart2pol(lab1(:,2), lab1(:,3));
[h2,c2] = cart2pol(lab2(:,2), lab2(:,3));

h1 = h1*180/pi;
h2 = h2*180/pi;
index = (h1> 180);
h1 = (1-index).*h1 + index.*(h1-180);
h2 = (1-index).*h2 + index.*(h2-180);

index = (h2<0);
h2 = (1-index).*h2 + index.*(h2+360);
p = (h2-h1);
index = (p==0);
p = (1-index).*p + index*1;
index = (p>180);
p = (1-index).*p + index.*(p-360);
p = p./abs(p);

end
```

Notice that the value of ΔH^*_{ab} is without sign because it results from a square-root function. The sub-function `dhpolarity` is therefore needed to ascertain whether the value should be positive or negative.

5.4 Optimised Colour-Difference Formulae

5.4.1 CMC (*l*:*c*)

In the late 1970s a number of different formulae were being used by practitioners, one of which was known as the JPC79 formula (the name derives from J&P Coats whose laboratories developed the formula). The JPC79 formula was effective but was known to be deficient in some areas (Smith, 1997) and a revised version of the formula was published in 1984 by members of the Colour Measurement Committee of the Society of Dyers and Colourists (Clarke, McDonald and Rigg, 1984). This revised formula became known as CMC(l:c) which, like most modern optimised colour-difference formulae, is based upon the CIELAB colour-difference components ΔL^*, ΔC^*_{ab} and ΔH^*_{ab} thus:

$$\Delta E_{CMC\,(l:c)} = \sqrt{\left(\Delta L^*/lS_L\right)^2 + \left(\Delta C^*_{ab}/cS_C\right)^2 + \left(\Delta H^*_{ab}/S_H\right)^2} \tag{5.16}$$

where if $L^*_S \geq 16$ then $S_L = 0.040975L^*_S/\left(1 + 0.01765L^*_S\right)$
and if $L^*_S < 16$ then $S_L = 0.511$
and $S_C = 0.638 + 0.0638C^*_{ab,S}/\left(1 + 0.0131C^*_{ab,S}\right)$
$S_H = S_C\,(TF + 1 - F)\,.$

The terms T and F are given by:

$$F = \sqrt{\left(C^*_{ab,S}\right)^4 / \left(\left(C^*_{ab,S}\right)^4 + 1900\right)}$$
$$T = 0.36 + \left|0.4\cos\left(h_{ab,S} + 35\right)\right| \quad \text{if } h_{ab,S} \leq 164 \text{ or } h_{ab,S} \geq 345$$
$$T = 0.56 + \left|0.2\cos\left(h_{ab,S} + 168\right)\right| \quad \text{if } 164 < h_{ab,S} < 345.$$

The subscript S, as in $C^*_{ab,S}$, denotes that the terms S_L, S_C, S_H, F and T are computed using the CIELAB lightness, chroma and hue angle (in degrees) of the standard. The terms S_L, S_C and S_H define the lengths of the semi-axes of the tolerance ellipsoid at the position of the standard in CIELAB space in each of the three directions (S_L for lightness, S_C for chroma, and S_H for hue). The ellipsoids were fitted to visual tolerances determined from psychophysical experiments and the semi-axes in the CMC(l:c) formula are used to effectively convert these ellipsoids into spheres at each point in CIELAB space. The parametric terms l and c constitute an important feature of the formula. These parameters allow the relative tolerances of the lightness and chroma components to be modified. For the textile industry it was recommended that $l = c = 1$ for perceptibility decisions whereas for acceptability decisions it was recommended that $l = 2$ with $c = 1$. The reason for this difference is that it is considered that, in terms of acceptability, differences in lightness should be weighted to be half as important as differences in either chroma or hue. The CMC(l:c) formula has been widely used in a number of industries and was adopted, for example, as a British standard (BS 6923) and an AATCC test method (AATCC 173). However, it was never adopted as a CIE standard.

The function `cmcde` computes CMC(l:c) colour differences between pairs of CIELAB coordinates. The function is called thus:

```
[de,dl,dc,dc] = cmcde(lab1,lab2,sl,sc);
```

where **lab1** and **lab2** are each $n \times 3$ matrices of the same size containing the CIELAB values for corresponding pairs. Note that the function returns four $n \times 1$ matrices containing the CMC colour differences and the component differences.

```
% ================================================
% *** FUNCTION cmcde
% ***
% *** function [de, dl, dc, dh] =cmcde(lab1,lab2,sl,sc)
% *** computes colour difference from CIELAB values
% *** using CMC formula
% *** inputs must be n by 3 matrices
% *** and contain L*, a* and b* values
% *** see also cielabde, cie94de, and cie00de
function [de,dl,dc,dh] = cmcde(lab1,lab2,sl,sc)
```

```
if (size(lab1,1)~=size(lab2,1))
    disp('inputs must be the same size'); return;
end

if (size(lab1,2)~=3 | size(lab2,2)~=3)
    disp('inputs must be n by 3'); return;
end

if (nargin<4)
    disp('using default values of l:c')
    sl=1; sc=1;
end

de = zeros(1,size(lab1,2));
dl = zeros(1,size(lab1,2));
dc = zeros(1,size(lab1,2));
dh = zeros(1,size(lab1,2));

% first compute the CIELAB deltas
dl = lab2(:,1)-lab1(:,1);
dc = (lab2(:,2).^2 + lab2(:,3).^2).^0.5-(lab1(:,2).^2
       + lab1(:,3).^2).^0.5;
dh = ((lab2(:,2)-lab1(:,2)).^2 + (lab2(:,3)
       lab1(:,3)).^2 - dc.^2);
dh = (abs(dh)).^0.5;
dh = ((lab2(:,2)-lab1(:,2)).^2 + (lab2(:,3)
       lab1(:,3)).^2 - dc.^2);

% get the polarity of the dh term
dh = dh.*dhpolarity(lab1,lab2);

% now compute the CMC weights
Lweight = [lab1(:,1)<16]*0.511+(1
       [lab1(:,1)<16]).*(0.040975*lab1(:,1))./(1 +
       0.01765*lab1(:,1));
[c,h] = cart2pol(lab1(:,2), lab1(:,3));
Cweight = 0.638 + (0.0638*c)./(1 + 0.0131*c);
index = (164<h & h<345);
T = index.*(0.56 + abs(0.2*cos((h+168)*pi/180)));
T = (1-index).*(0.36 + abs(0.4*cos((h+35)*pi/180)));
F = ((c.^4)./(c.^4 + 1900)).^0.5;
Hweight = Cweight.*(T.*F + 1 - F);

dl = dl./(Lweight*sl);
dc = dc./(Cweight*sc);
dh = dh./Hweight;

de = (dl.^2 + dc.^2 + dh.^2).^0.5;
end
```

Note that the sub-function `dhpolarity` is not repeated here. See the function `cielabde` for details of this sub-function.

5.4.2 CIE 94

Berns (2000) and others argued that the complexity of the CMC equation and the use of large numbers of significant figures in its definition suggest a degree of precision that cannot be supported on statistical grounds. Detailed analyses of large sets of psychophysical data suggested that simple S_L, S_C and S_H weighting functions could be sufficient and this led to the publication of a new formula known as CIE94 (Berns, 1993). The CIE94 formula is given by:

$$\Delta^*_{94} = \sqrt{\left(\Delta L^*/(K_L S_L)\right)^2 + \left(\Delta C^*_{ab}/(K_C S_C)\right)^2 + \left(\Delta H^*_{ab}/(K_C S_H)\right)^2} \tag{5.17}$$

where $S_L = 1,$
$S_C = 1 + 0.045 C^*_{ab,S},$
$S_H = 1 + 0.015 C^*_{ab,S}.$

The parametric variables K_L, K_C and K_H are all set to unity and the values of S_C and S_H are computed using the CIELAB values of the standard as indicated. When neither sample can logically be deemed a standard the geometric mean of the two samples should be used.

The function `cie94de` computes CIE94 colour differences between pairs of CIELAB coordinates. The function is called thus:

```
[de,dl,dc,dc] = cie94de(lab1,lab2);
```

where **lab1** and **lab2** are each $n \times 3$ matrices of the same size containing the CIELAB values for corresponding pairs. Note that the function returns four $n \times 1$ matrices containing the `CIE94` colour differences and the component differences.

```
% ================================================
% *** FUNCTION cie94de
% ***
% *** function [de, dl, dc, dh] =cie94de(lab1,lab2)
% *** computes colour difference from CIELAB values
% *** using CIE94 formula
% *** inputs must be n by 3 matrices
% *** and contain L*, a* and b* values
% *** see also cielabde, cmcde, and cie00de
function [de,dl,dc,dh] = cie94de(lab1,lab2)

if (size(lab1,1)~=size(lab2,1))
    disp('inputs must be the same size'); return;
end
```

```
if (size(lab1,2)~=3 | size(lab2,2)~=3)
    disp('inputs must be n by 3'); return;
end

de = zeros(1,size(lab1,2));
dl = zeros(1,size(lab1,2));
dc = zeros(1,size(lab1,2));
dh = zeros(1,size(lab1,2));

% first compute the CIELAB deltas
dl = lab2(:,1)-lab1(:,1);
dc = (lab2(:,2).^2 + lab2(:,3).^2).^0.5-(lab1(:,2).^2
       + lab1(:,3).^2).^0.5;
dh = ((lab2(:,2)-lab1(:,2)).^2 + (lab2(:,3)
       lab1(:,3)).^2 - dc.^2);
dh = (abs(dh)).^0.5;
dh = ((lab2(:,2)-lab1(:,2)).^2 + (lab2(:,3)
       lab1(:,3)).^2 - dc.^2);

% get the polarity of the dh term
dh = dh.*dhpolarity(lab1,lab2);

% now compute the CIE94 weights
Lweight = 1.0;
[c,h] = cart2pol(lab1(:,2), lab1(:,3));
Cweight = 1.0 + 0.045*c;
Hweight = 1.0 + 0.015*c;

dl = dl/Lweight;
dc = dc./Cweight;
dh = dh./Hweight;

de = (dl.^2 + dc.^2 + dh.^2).^0.5;
end
```

Note that the sub-function `dhpolarity` is not repeated here. See the function `cielabde` for details of this sub-function. The use of the CIE94 colour-difference formula is no longer recommended by the CIE (Hunt and Pointer, 2011).

5.4.3 CIEDE2000

The CIE recommended for trial the CIEDE2000 colour-difference formula for the evaluation of small colour differences (Luo, Cui and Rigg, 2001). Note that CIELAB ΔE^*_{ab} remains the current CIE recommendation for the evaluation of large colour differences ($\Delta E^*_{ab} > 5$). Performance testing of CIEDE2000 suggests that it is significantly better than CIELAB and CIE94 (see Melgosa and Huertas, 2004; Melgosa, 2006; Shen and Berns, 2011).

The CIEDE2000 formula was agreed by a technical committee within Division 1 of the CIE (2001) and includes not only lightness, chroma and hue weighting functions, but also an interactive term between chroma and hue differences for improving the performance for blue colours and a scaling factor for the CIELAB a^* scale to improve the performance for colours close to the achromatic axis.

The new formula, referred to as ΔE_{00}, is given by Equation 5.18:

$$\Delta E_{00} = \left[\begin{array}{l} (\Delta L'/(K_L S_L))^2 + (\Delta C'/(K_C S_C))^2 + (\Delta H'/(K_H S_H))^2 \\ + R_T(\Delta C'/(K_C S_C))(\Delta H'/(K_H S_H)) \end{array} \right]^{0.5} \tag{5.18}$$

where $S_L = 1 + 0.015\left(L' - 50\right)^2 / \left(20 + \left(L' - 50\right)^2\right)^{0.5}$

$S_C = 1 + 0.045C'$

and $S_H = 1 + 0.015TC'$.

The terms $\Delta L'$, $\Delta C'$ and $\Delta H'$ are given by:

$$\Delta L' = L'_T - L'_S$$
$$\Delta C' = C'_T - C'_S$$
$$\Delta H' = 2\sqrt{C'_T C' S}\,\sin\left(\Delta h'/2\right)$$

where the subscripts S and T refer to the standard and trial respectively, and where:

$$\Delta h' = h'_T - h'_S$$
$$L' = L^*$$
$$a' = (1 + G)\,a^*$$
$$b' = b^*$$
$$C' = \sqrt{a'^2 + b'^2}$$

and:

$$h' = \tan^{-1}\left(b'/a'\right).$$

The terms G and T are calculated using:

$$G = 0.5 - 0.5\sqrt{C^{*7}_{ab}/\left(C^{*7}_{ab} + 25^7\right)}$$

and:

$$T = 1 - 0.17\cos(h' - 30) + 0.24\cos(2h') + 0.32\cos(3h' + 6) - 0.20\cos(4h' - 63)$$

Finally, the rotation term R_T given by:

$$R_T = -\sin(2\Delta\theta)R_C$$

where $R_C = 2\sqrt{C'^7/\left(C'^7 + 25^7\right)}$

and $\Delta\theta = 30\exp\left\{-\left(\left(h' - 275\right)/25\right)^2\right\}$.

Note that the arithmetic mean of the CIELAB values of standard and trial are used for computing the values of the terms such as S_L. The CIEDE2000 formula has been shown to outperform the CMC and CIE94 formulae by a large margin (Luo, Cui and Rigg, 2001) and it is the current CIE standard method for computing small colour differences.

Interestingly, the Lightness component of the formula is very different from those in earlier formulae. For example, the value of the function in the CMC formula increases markedly as L^* increases, implying that for equal differences in L^* the visual difference should be largest in the low L^* region. The Lightness correction in the CIE94 formulae, on the other hand, implied that the CIELAB L^* scale was correct, so that equal differences in L^* would yield equal visual differences no matter what the value of L^*. The S_L formula in CIEDE2000, however, was based upon new data (Heptinstall, 1999; Chou *et al*., 2001) so that S_L increases with L^* only for $L^* > 50$; for lower values of L^* the value of S_L decreases as L^* increases. It is still not at all clear why the new data upon which CIEDE2000 was based should have been so different from the data upon which the earlier formulae were based. Nevertheless, the evidence for CIEDE2000 is convincing and there is strong confidence that the new formula is reliable (Cui *et al*., 2001; Luo, 2002a). We note that an alternative form of the equation has been provided by Nobbs that apportions the colour difference into hue, chroma and lightness components (Nobbs, 2002; Hunt and Pointer, 2011).

The function `cie00de` computes colour differences according to the CIEDE2000 equation between pairs of CIELAB coordinates. The function is called thus:

```
[de,dl,dc,dc] = cie00de(lab1,lab2,sl,sc,sh);
```

where **lab1** and **lab2** are each $n \times 3$ matrices of the same size containing the CIELAB values for corresponding pairs. Note that the function returns four $n \times 1$ matrices containing the CIEDE2000 colour differences and the component differences.

```
% ================================================
% *** FUNCTION cie00de
% ***
% *** function [de,dl,dc,dh] = cie00de(lab1,lab2,sl,sc,sh)
% *** computes colour difference from CIELAB values
% *** using CIEDE2000 formula
% *** inputs must be n by 3 matrices
% *** and contain L*, a* and b* values
% *** see also cielabde, cmcde, and cie94de
function [de,dl,dc,dh] = cie00de(lab1,lab2,sl,sc,sh)

if (size(lab1,1)~=size(lab2,1))
    disp('inputs must be the same size'); return;
end
```

```
if (size(lab1,2)~=3 | size(lab2,2)~=3)
    disp('inputs must be n by 3'); return;
end
if (nargin<5)
    disp('using default values of l:c')
    sl=1; sc=1; sh=1;
end
de = zeros(1,size(lab1,2));
dl = zeros(1,size(lab1,2));
dc = zeros(1,size(lab1,2));
dh = zeros(1,size(lab1,2));

% convert the cartesian a*b* to polar chroma and hue
[c1,h1] = cart2pol(lab1(:,2), lab1(:,3));
[c2,h2] = cart2pol(lab2(:,2), lab2(:,3));
meanC = (c2+c1)/2;

% compute G factor using the arithmetic mean chroma
G = 0.5 - 0.5*(((meanC.^7)./(meanC.^7 + 25^7)).^0.5);

% transform the a* values
lab1(:,2) = (1 + G).*lab1(:,2);
lab2(:,2) = (1 + G).*lab2(:,2);

% recompute the polar coordinates using the new a*
[c1,h1] = cart2pol(lab1(:,2), lab1(:,3));
[c2,h2] = cart2pol(lab2(:,2), lab2(:,3));

% compute the mean values for use later meanC = (c2+c1)/2;
meanL = (lab2(:,1)+lab1(:,1))/2;
[a1,b1] = pol2cart(ones(1,size(h1,2)), h1);
[a2,b2] = pol2cart(ones(1,size(h2,2)), h2);
a = (a1+a2)/2;
b = (b1+b2)/2;
[c,meanH] = cart2pol(a,b);

% compute the basic delta values
dh = (h2-h1);
index = dh>180;
dh = (index).*(dh-360);
dh = dh + (1-index).*dh;
dh = 2*((c1.*c2).^0.5).*sin((dh/2)*pi/180);
dl = lab2(:,1)-lab1(:,1);
dc = c2-c1;

T = 1 - 0.17*cos((meanH-30)*pi/180) + 0.24*cos((2*meanH)*pi/180);
T = T + 0.32*cos((3*meanH + 6)*pi/180) - 0.20*
       cos((4*meanH - 63)*pi/180);
```

```
dthe = 30*exp(-((meanH-275)/25).^2);
rc = 2*((meanC.^7)./(meanC.^7 + 25^7)).^0.5;
rt = -sin(2*dthe*pi/180).*rc;

Lweight = 1 + (0.015*(meanL-50).^2)./((20 + (meanL-50).^2).^0.5);
Cweight = 1 + 0.045*meanC;
Hweight = 1 + 0.015*meanC.*T;

dl = dl./(Lweight*sl);
dc = dc./(Cweight*sl);
dh = dh./(Hweight*sh);

de = sqrt(dl.^2 + dc.^2 + dh.^2 + rt.*dc.*dh);
```

The CIEDE2000 formula allows for three parametric terms for Lightness, Chroma and Hue weightings respectively. The default values for these parameters are all set to unity.

It has been noted that the implementation of CIEDE2000, in particular, is non-trivial (Sharma *et al*., 2005). The computation of hue from a^* and b^* is indeterminate when $a^* = b^* = 0$ and Sharma *et al*. (2005) suggest that a suitable value for hue in this case is zero. The MATLAB® function `cart2pol` that is used in our implementation naturally returns zero for this condition. The implementation requires that the hue values of the standard and trial be averaged. But the arithmetic mean cannot be simply computed directly since this would give a mean hue of 185° for the two hues 20° and 350° whereas the true average hue would be 5°. The approach taken here is to use the method suggested by Sharma *et al*. (2005) implemented as the following code segment:

```
meanH = (h1+h2)/2;
% Identify positions for which abs hue diff exceeds
       180 degrees
meanH = meanH - (abs(h1-h2) > 180) *180;
% rollover ones that come -ve
meanH = meanH + (meanH < 0)*360;
% Check if one of the chroma values is zero, in which
       case set mean hue to the sum which is
       equivalent to other value
index = find(c1.*c2 == 0);
meanH(index) = h1(index)+h2(index);
```

In this code the procedure to compute the hue difference ΔH is:

$$\Delta H = 2\sqrt{C_T C_S}\sin(\Delta h/2) \tag{5.19}$$

where Δh is the hue of the trial minus the hue of the standard. This method gives a relatively simple way to compute ΔH but a correction is still required to ensure the

correct sign is always computed. The correction is to subtract 360 from Δh if $\Delta h > 180$. This is implemented as the following code segment:

```
dh = (h2-h1);
index = dh> 180;
dh = (index).*(dh-360) + (1-index).*dh;
dh = 2*((c1.*c2).^0.5).*sin((dh/2)*pi/180);
```

Users may wish to modify the scripts or to convert them into other programming languages.

The tristimulus values in Table 5.1 are for the 1964 observer and illuminant D65 (X_n = 94.811, Y_n = 100.000, Z_n = 107.304) Table 5.2 lists the colour difference values for the CIELAB, CMC(1,1), CIE94, and CIEDE2000 equations. In order to facilitate testing of any implementations of these colour-difference equations Table 5.3 has been provided which lists 10 pairs of samples that Luo *et al*. (2001) have designed for testing the CIEDE2000 equation. Table 5.3 provides more detailed information on the intermediate stages for the CIEDE2000 equation.

Table 5.1 *XYZ pairs for testing colour difference equations (Luo et al., 2001).*

Pair	X_S	Y_S	Z_S	X_T	Y_T	Z_T
1	19.4100	28.4100	11.5766	19.5525	28.6400	10.5791
2	22.4800	31.6000	38.4800	22.5833	31.3700	36.7901
3	28.9950	29.5800	35.7500	28.7704	29.7400	35.6045
4	4.1400	8.5400	8.0300	4.4129	8.5100	8.6453
5	4.9600	3.7200	19.5900	4.6651	3.8100	17.7848
6	15.6000	9.2500	5.0200	15.9148	9.1500	4.3872
7	73.0000	78.0500	81.8000	73.9351	78.8200	84.5156
8	73.9950	78.3200	85.3060	69.1762	73.4000	79.7130
9	0.7040	0.7500	0.9720	0.6139	0.6500	0.8510
10	0.2200	0.2300	0.3250	0.0933	0.1000	0.1453

Table 5.2 *Test colour differences for the pairs of samples in Table 5.1.*

Pair	CIELAB	CMC(1:1)	CIE94	CIEDE2000
1	3.1819	1.4282	1.3910	1.2644
2	2.2134	1.2549	1.2481	1.2630
3	1.5390	1.7684	1.2980	1.8731
4	4.6063	2.0258	1.8204	1.8645
5	6.5847	3.0870	2.5561	2.0373
6	3.8864	1.7490	1.4249	1.4146
7	1.5051	1.9009	1.4194	1.4440
8	2.3238	1.7026	2.3226	1.5381
9	0.9441	1.8032	0.9385	0.6378
10	1.3191	2.4493	1.3065	0.9082

Table 5.3 *Test colour differences for the pairs of samples in Table 5.1 for CIEDE2000 (showing intermediate stages).*

Pair	G	T	S_L	S_C	S_H	R_T	DE_{00}
1	0.0017	1.3010	1.1427	3.2946	1.9951	0.0000	1.2644
2	0.0490	0.9402	1.1831	2.4549	1.4560	0.0000	1.2630
3	0.4966	0.6952	1.1586	1.3092	1.0717	−0.0032	1.8731
4	0.0063	1.0168	1.2148	2.9105	1.6476	00000	1.8645
5	0.0026	0.3636	1.4014	3.1597	1.2617	−1.2537	2.0373
6	0.0013	0.9239	1.1943	3.3888	1.7357	00000	1.4146
7	0.4999	1.1546	1.6110	1.1329	1.0511	00000	1.4440
8	0.5000	1.3916	1.5930	1.0620	1.0288	00000	1.5381
9	0.4999	0.9556	1.6517	1.1057	1.0337	−0.0004	0.6378
10	0.5000	0.7827	1.7246	1.0383	1.0100	00000	0.9082

Note that a more thorough set of test data for confirming CIEDE2000 implementations has been provided in the literature (Sharma *et al*., 2005). These data, comprising 34 colour pairs (and Sharma's MATLAB® and Excel implementations of CIEDE2000 are available online (Sharma, 2011).

6

Chromatic-Adaptation Transforms and Colour Appearance

6.1 Introduction

The distinction between colour specification and colour appearance was touched upon in the review of the CIE system presented in Section 1.3. Whereas the CIE system of colorimetry (based upon *XYZ* tristimulus values) is clearly a system for colour specification, some advanced colour specification models such as CIELAB could arguably be described as models of colour appearance. The polar coordinates of CIELAB allow the description of a colour stimulus in terms of three terms, *Lightness, Chroma* and *Hue*, and these correlate quite well with the perceptual attributes of *Brightness, Colourfulness* and *Hue*. Furthermore, the normalisation procedures inherent in the transform from *XYZ* to CIELAB result in a^* and b^* values close to zero for a perfect reflecting diffuser (or any surface whose spectral reflectance does not vary with wavelength) irrespective of the illuminant. This is consistent with the fact that surfaces in general tend to retain their daylight colour appearance when viewed under a wide range of light sources and contrasts with the properties of the *XYZ* system. In some sense then, CIELAB is a colour-appearance space whereas XYZ is not. CIELAB is a relatively poor colour-appearance model, however, and this chapter describes several advanced colour-appearance models (CAMs).

The human visual system in most cases has a remarkable ability to maintain the colour appearance of an object despite quite large changes in the quality and intensity of the illumination. A white piece of paper tends to look white whether it is viewed by daylight, tungsten light or candlelight. It is generally considered that the human visual system achieves colour constancy by some process that allows it to discount the effect of the illumination. The term *chromatic adaptation* is often used to

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.

describe this process and the chromatic adaptation is said to be complete if the effect of the illumination is completely discounted. Most CAMs therefore include a chromatic-adaptation transform (CAT). A CAT is a method for computing the corresponding colour under a reference illuminant for a stimulus defined under a test illuminant. Corresponding colours are colours that have the same appearance under different illumination (Bartleson, 1978).

Fairchild (1998) defines a CAM as any model that includes predictors of at least the relative colour-appearance attributes of lightness, chroma and hue. The attribute *brightness* is a visual perception according to which an area appears to exhibit more or less light. *Lightness* is the brightness of an area judged relative to the brightness of a similarly illuminated reference white. The lightness of a sample is usually in the range 0 to 100 and is influenced by the surrounding background[1]. *Colourfulness* is that attribute of a visual sensation according to which an area appears to exhibit more or less chromatic content. Hunt (1952) has shown that the colourfulness of an object increases as the luminance increases so that a typical outdoor scene appears much more colourful in bright sunlight than it does on an overcast day (the Hunt effect). *Chroma* is the colourfulness of an area judged as a proportion of the brightness of a similarly illuminated reference white. The colourfulness of an area judged in proportion to its brightness is called the *saturation*. Finally, *hue* is the attribute of a sensation according to which an area appears to be similar to one, or to a proportion of two, of the perceived colours; red, yellow, green and blue.

The basic principles that underlie CATs will be introduced and two models (CMCCAT97 and CMCCAT2000) will be described in detail. The CIECAM97s colour-appearance model will then be described and CIECAM02 will be introduced.

6.2 Chromatic-Adaptation Transforms (CATs)

In psychophysical studies and modelling of chromatic adaptation it is useful to define the concept of *corresponding colours*. If a colour stimulus seen in one set of viewing conditions has the same appearance as another stimulus seen in another set of conditions, then the two stimuli are known as corresponding colours (Hunt *et al*., 2005). In a typical colour-appearance experiment to determine the corresponding colour of a grey surface or chip under a test light source (for example, illuminant A) observers adapt to the chip viewed under that light source and are then asked to memorise the colour of the chip. They are then adapted to the reference light source (often illuminant D65) and requested to select a chip, from a large number of different coloured chips, which matches the memorised colour of the original chip that was viewed under the test illumination. Figure 6.1 shows the chromaticities of a perfectly neutral grey under D65 and illuminant A. If the observer is able to discount the change in illumination perfectly the colour appearance will be the same under both the test and reference illuminations. Therefore for the grey chip shown in Figure 6.1 the observer would select the same grey chip under the reference illumination (D65) as was viewed under the test illumination (A).

[1] CIELAB lightness is 100 for a perfect white but for some samples, for example those that are highly fluorescent, lightness can greatly exceed 100.

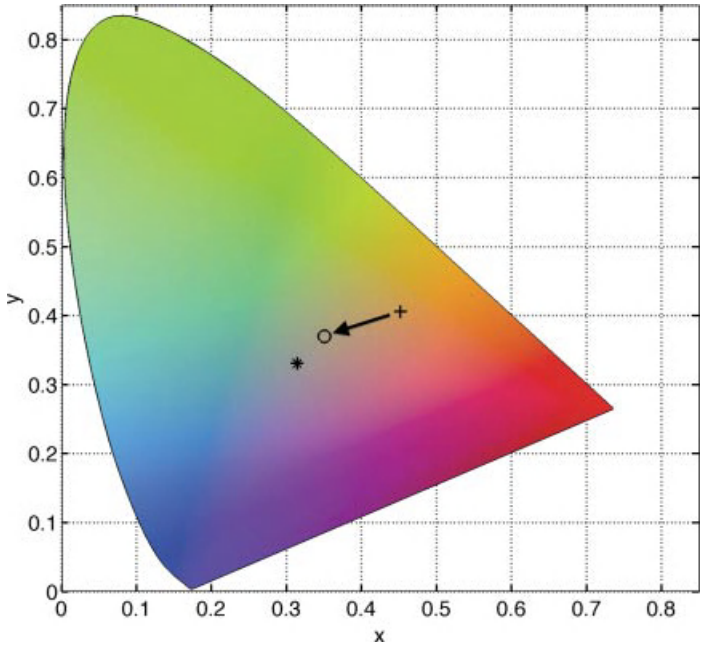

Figure 6.1 *Corresponding colour (o) under D65 (*) for a grey sample viewed under illuminant A (+).*

Usually, in such experiments the corresponding colours are not those predicted by a process discounts the illumination change perfectly. In Figure 6.1 the corresponding colour (denoted by o) determined under the reference illuminant D65 (*) to match a neutral chip viewed under illuminant A (+) is shown for a hypothetical experiment.

Fairchild (1998) has classified chromatic-adaptation mechanisms into two groups: sensory and cognitive. Sensory chromatic-adaptation mechanisms refer to those that respond automatically to the stimulus and are thought to relate to control mechanisms in the sensitivities of the long-, medium- and short-wavelength sensitive cone classes. Cognitive chromatic-adaptation mechanisms refer to higher-level cognitive processes that may relate to our understanding of scene content. Research has shown that chromatic-adaptation mechanisms are quite rapid, being 50% complete after 4 s, 90% complete after 70 s and 99% complete after 110 s (Fairchild and Lennie, 1992; Fairchild and Reniff, 1995).

A chromatic-adaptation transform (CAT) is a method for computing the corresponding colour under a reference illuminant for a stimulus defined under a test illuminant. Most modern CATs are at least loosely based on the von Kries model of adaptation. The likely mechanism underlying this process is that under a reddish light, for example, the long-wavelength-sensitive cones in particular will adapt and so become less sensitive. Under a bluish light, however, the sensitivity of the long-wavelength-sensitive cones will increase. In this way, the idea is that the cone responses for a given surface will stay almost the same even when the illumination is changed and that the visual system will be able to use the cone excitations to provide a constant appearance for a surface when

the illumination changes even though the spectral distribution of light entering the eye is changed.

The changes in sensitivity can be modelled for a static visual system by assuming that the cone responses for a surface under one illuminant can be predicted from those under another illuminant by simple scaling factors. Thus the long-wavelength response for a surface viewed under one light source can be obtained by multiplying the long-wavelength response for the surface viewed under a different light source by a scalar. The scalar values may be different for each cone class but critically do not depend upon the reflectance or chromaticity of the sample. In terms of linear algebra we can state that the cone responses for (a sample viewed under) one illuminant can be related to those for another illuminant by a linear transform. Since the linear transform's system matrix has non-zero entries only along the major diagonal, it is referred to as diagonal transform. Thus, for example, the cone responses under one illuminant (represented by the 3×1 column matrix $\mathbf{e_1}$) are the related to the cone responses under a second illuminant (represented by the 3×1 column matrix $\mathbf{e_2}$) by the diagonal matrix $\mathbf{D}$, thus:

$$\mathbf{e_2} = \mathbf{D}\mathbf{e_1} \tag{6.1}$$

where the coefficients of the diagonal matrix are given by the ratios of the long-, medium-, and short-wavelength sensitive cone responses (L_W, M_W and S_W) for a white viewed under each of the two illuminants, thus:

$$\mathbf{D} = \begin{bmatrix} L_{W,2}/L_{W,1} & 0 & 0 \\ 0 & M_{W,2}/M_{W,1} & 0 \\ 0 & 0 & S_{W,2}/S_{W,1} \end{bmatrix}. \tag{6.2}$$

The von Kries law is sometimes called the coefficient law or the scaling law since it assumes that the effect of an illumination change can be modelled by simply scaling the tristimulus values or cone responses by scalars or coefficients (the diagonal elements of $\mathbf{D}$). When the von Kries adaptation transform is performed using cone space then Terstiege (1972) has termed this a genuine von Kries transformation whereas practically it is often carried out in CIE *XYZ* space or in an *RGB* space when it is referred to as a wrong von Kries transformation[2].

Analyses of experimental data suggest that although Equation 6.1 cannot perfectly predict the performance of observers in psychophysical experiments psychophysical data can be modelled by such a system at least to a first-order approximation (Wandell, 1995). However, Finlayson and Süsstrunk has shown that a diagonal mapping is always possible between two real three-dimensional spaces if the spaces are first subject to a specific linear transformation. They argue that if the tristimulus values or cone responses are first transformed by a linear transform into a suitable *RGB* space, then a diagonal transform can effectively discount the illumination (Finlayson and Süsstrunk, 2000). The first linear transform is sometimes called a sharp transform since it can be shown to convert the cone responses into a set of channels whose spectral sensitivities are sharper than those that have been measured for humans. We can therefore consider a generalised CAT based

[2] Though, of course, such a 'wrong' transformation may be effective in certain cases.

upon Equation 6.3 where $\mathbf{c_1}$ and $\mathbf{c_2}$ refer to the tristimulus values of the sample under the two illuminants:

$$\mathbf{c_2} = \mathbf{M_{CAT}^{-1}} \mathbf{D} \mathbf{M_{CAT}} \mathbf{c_1} \tag{6.3}$$

and the diagonal matrix $\mathbf{D}$ is now composed from the white points of the two illuminants in the sharpened *RGB* space. In Equation 6.3 the tristimulus values $\mathbf{c_1}$ are first subject to a linear transform $\mathbf{M_{CAT}}$ which converts them into the *RGB* space, then to a diagonal transform $\mathbf{D}$ to apply the illuminant correction, and finally a linear transform $\mathbf{M_{CAT}^{-1}}$ to convert back to tristimulus space. Finlayson has derived the *RGB* or sharp transform as given by $\mathbf{M_{CAT}} = \mathbf{M_{SHARP}}$:

$$M_{SHARP} = \begin{bmatrix} 1.2694 & -0.0988 & -0.1706 \\ -0.8364 & 1.8006 & 0.0357 \\ 0.0297 & -0.0315 & 1.0018 \end{bmatrix} \tag{6.4}$$

The most popular CATs are consistent with Finlayson's idea and the procedure of subjecting the tristimulus values of a stimulus under one illuminant by a 3×3 linear transform, followed by a diagonal transform, and finally followed by the inverse linear transform to return to tristimulus space is ubiquitous in CAT research. Often researchers refer to the *RGB* space in which the diagonal transform takes place as cone space although the term is usually being used loosely in this sense.

A number of CATs are currently in use and the 3×3 linear transform $\mathbf{M_{CAT}}$ is different for each CAT. However, more significant differences between the transforms are found in the way in which the elements of the diagonal transform are computed and in which properties of the observing field are used to compute these elements.

Hunt (1998) classified the observing field into five areas: the colour element, the proximal field, the background, the surround, and the adapting field and these areas are shown schematically in Figure 6.2. The colour element is the central area of the observing field and this is typically a uniform patch of approximately 2° of visual angle. The proximal field is the immediate environment of the colour element extending for approximately 2° from the edge of the colour element. The colour element and its proximal field are considered to be viewed against the background, a region extending approximately 10° in every direction from the edge of the proximal field. The surround

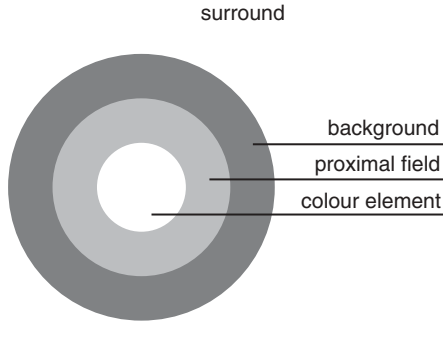

Figure 6.2 *Hunt's (1998) classification of colour appearance fields.*

is the field outside of the background. Finally, the adapting field is the total environment within which the colour element, the proximal field and the background are viewed.

It is common practice to follow Moroney's terminology (2000) so that that the term *adopted white* is used to describe the computational white point used in various model calculations whereas the term *adapted white* is used to define the white point to which a human observer is considered to be adapted to.

6.2.1 A Brief History of CATs

In 1994 the CIE recommended a CAT developed by Nayatani and colleagues (Nayatani *et al*., 1990; Nayatani *et al*., 1999) known as CIECAT94. Unlike the simple von Kries model, CIECAT94 takes into account the luminance level used and the degree of adaptation. This model therefore led the way for a plethora of modern CATs that currently dominate the colour literature. However, a number of studies have shown that the complexity of the CIECAT94 model is not justified by its performance (e.g. Sueeprasan, 2003).

Luo and Hunt (1998b) proposed a modified version of an earlier transform called the Bradford CAT (Lam, 1985; Hunt, 1997) that is known as CMCCAT97. Finlayson and Süsstrunk derived a CAT based upon the concept of sharpened sensors (Equation 6.4). There was some uncertainty over the reversibility of the CMCCAT97 transform and although this was solved by a small revision (Li, Luo and Hunt, 2000) a further weakness of the CMCCAT97 is that it was derived by fitting only a relatively small data set. Further work resulted in the development of a new CAT that was accepted by the Colour Measurement Committee of the Society of Dyers and Colourists and known as CMCCAT2000 (Li *et al*., 2002). Moroney *et al*. (2002) proposed a small modification of CMCCAT2000 to be used with the CIECAM02 model. In 2004 a CIE committee (CIE TC 1–52) was unable to agree that a single CAT (out of thirteen that were evaluated) should be recommended (Bianco and Schettini, 2010). Recently, two new CATs have been derived from numerical optimisation that outperform (or are statistically equivalent to) all existing CATs on the corresponding colour datasets that are available (Bianco and Schettini, 2010). A more detailed history and analysis of CATS is available in the literature (Fairchild, 2005; Nayatani, 2006). In the following sections CMCCAT97 and CMCCAT2000 are described and MATLAB® code that implements them is provided.

6.2.2 CMCCAT97

The CMCCAT97 CAT (Luo and Hunt, 1998b) is used in the CIECAM97s colour-appearance model. It is essentially based upon the form of Equation 6.3 where $\mathbf{M_{CAT}} = \mathbf{M_{BFD}}$ and:

$$\mathbf{M_{BFD}} = \begin{bmatrix} 0.8951 & 0.2664 & -0.1614 \\ -0.7502 & 1.7135 & 0.0367 \\ 0.0389 & -0.0685 & 1.0296 \end{bmatrix} \tag{6.5}$$

However, in CMCCAT97 the transformation to *RGB* space is performed upon the tristimulus values each normalised by the Y value of the sample.

The XYZ values of the adopted test and reference illuminants are also normalised by their respective Y values and transformed by $\mathbf{M_{BFD}}$ to yield R_{WT}, G_{WT}, B_{WT}, and R_{WR}, G_{WR}, B_{WR} respectively. CMCCAT97 incorporates the degree of adaptation D and this is computed by Equation 6.6:

$$D = F - \frac{F}{1 + 2L_A^{0.25} + L_A^2/300} \tag{6.6}$$

where $F = 1$ for stimuli seen under typical viewing conditions, $F = 0.9$ for stimuli seen under dim or dark conditions[3], and L_A is the luminance (in units cd/m^2) of the adapting test field (note, however, that D should be set as 1 if adaption is assumed to be complete). The degree of adaptation D is then used with the ratios of the white points of the illuminants to convert the RGB values of the sample to the RGB values of the corresponding colour thus:

$$\begin{aligned} R_C &= (D(R_{WR}/R_{WT}) + 1 - D)R \\ G_C &= (D(G_{WR}/G_{WT}) + 1 - D)G \\ B_C &= (D(B_{WR}/B_{WT}^P) + 1 - D)|B^P| \end{aligned} \tag{6.7}$$

or, where $B < 0, B_C = -(D(B_{WR}/B_{WT}^P) + 1 - D)|B^P|$, and where $P = (B_{WT}/B_{WR})^{0.0834}$.

Note that when $D = 1$, the transform (Equation 6.7) is quite close to a form of von Kries or diagonal transform except that a nonlinearity is applied to the B channel.

Finally, the corresponding RGB values are converted back to tristimulus values by multiplying them by the inverse of $\mathbf{M_{BFD}}$ to yield the normalised tristimulus values which can finally be converted by multiplying each by the Y tristimulus value of the sample under the test illuminant. The inverse of $\mathbf{M_{BFD}}$ is given by Luo and Hunt (1998b) as:

$$\mathbf{M_{BFD}^{-1}} = \begin{bmatrix} 0.98699 & -0.14705 & 0.15996 \\ 0.43231 & 0.51836 & 0.04929 \\ -0.00853 & 0.04004 & 0.96849 \end{bmatrix} \tag{6.8}$$

The function `cmccat97` implements the CMCCAT97 CAT. The function is called thus:

```
[xyzc] = cmccat97(xyz,xyzt,xyzr,la,f);
```

where **xyz** is an $n \times 3$ matrix of XYZ values for stimuli under the test illuminant, **xyzt** and **xyzt** are the tristimulus values of the test and reference illuminants respectively, **la** is the luminance of the adapting test field, and **f** is normally unity (see Equation 6.6). The function returns an $n \times 3$ matrix of corresponding tristimulus values.

[3] Dark surrounds occur when film is projected in rooms where there is little or no light except that which comes from the projector. Dim surrounds occur when self-luminous displays are viewed in environments having luminances substantially lower than that of the display (Luo and Hunt, 1998b).

```
% ================================================
% *** FUNCTION cmccat97
% ***
% *** function [xyzc] = cmccat97(xyz,xyzt,xyzr,la,f)
% *** implements the cmccat97s CAT
% *** xyz must be an n by 3 matrix of XYZ
% *** xyzt and xyzr contain white points
% *** f has default value of 1
% *** la is the luminance of the adapting field and
% *** has a default value of 100
% *** see also

function [xyzc] = cmccat97(xyz,xyzt,xyzr,la,f)

if (size(xyz,2)~=3)
   disp('first input must be n by 3'); return;
end
if (length(xyzt)~=3 | length(xyzr)~=3)
   disp('check white point inputs'); return;
end

if (nargin<5)
   disp('using default values of la and f')
   la = 100; f=1;
end

xyzc = xyz; % to allocate memory
rgb = xyz;

% define the matrix for the transform to 'cone' space
M(1,:) = [0.8951 0.2664 -0.1614];
M(2,:) = [-0.7502 1.7135 0.0367];
M(3,:) = [0.0389 -0.0685 1.0296];

% normalise xyz and transform to rgb
rgb = (M*(xyz./[xyz(:,2) xyz(:,2) xyz(:,2)])')';
rgbt = M*(xyzt/xyzt(2))';
rgbr = M*(xyzr/xyzr(2))';

%  compute d, the degree of adaptation
d = f - f/(1 + 2*(la^0.25) + la*la/300);
% clip d if it is outside the range [0,1]
if (d<0)
   d=0;
elseif (d>1)
   d=1;
end
p = (rgbt(3)/rgbr(3))^0.0834;
```

```
% compute corresponding rgb values
rgbc(:,1) = rgb(:,1)*(d*rgbr(1)/rgbt(1) + 1 - d);
rgbc(:,2) = rgb(:,2)*(d*rgbr(2)/rgbt(2) + 1 - d);
rgbc(:,3) = rgb(:,3).^p*(d*(rgbr(3)/(rgbt(3)^p)) + 1 - d);

index = (rgb(:,3)<0);
rgbc(:,3) = index*(-rgbc(:,3)) + (1-index)*rgbc(:,3);

% implement step 4: convert from rgb to xyz
xyzc = [xyz(:,2) xyz(:,2) xyz(:,2)].*(inv(M)*rgbc')';
```

Table 6.1 lists some XYZ values under illuminant A (1964 observer) and their corresponding values under illuminant D65 (1964 observer) which will allow readers to check that any code that they write is operating correctly. The luminance of the test adapting field is assumed to be 200 cd/m^2.

6.2.3 CMCCAT2000

In order to address some concerns about the CMCCAT97 transform further work by Luo's research group resulted in the development of a new CAT that was accepted by the Colour Measurement Committee and known as CMCCAT2000 (Li *et al*., 2002). In CMCCAT2000 the power function in the blue channel of CMCCAT97 was removed so that the transform is fully reversible and the model was fitted to all available data sets. Consequently the sharpening component $\mathbf{M}_{\mathbf{CMCCAT2000}}$, of CMCCAT2000 is slightly different from that of CMCCAT97 and is shown below:

$$\mathbf{M}_{\mathbf{CMCCAT2000}} = \begin{bmatrix} 0.7982 & 0.3389 & -0.1371 \\ -0.5918 & 1.5512 & 0.0406 \\ 0.0008 & 0.0239 & 0.9753 \end{bmatrix} \tag{6.9}$$

Table 6.1 *XYZ values for 10 test samples under illuminant A and their corresponding values under illuminant D65 using CMCCAT97. The luminance of the test adapting field is assumed to be 200 cd/m^2.*

Pair	X	Y	Z	X_C	Y_C	Z_c
1	19.41	28.41	11.57	17.30	30.35	34.02
2	22.48	31.60	38.48	27.90	36.27	106.19
3	28.99	29.58	35.75	32.69	32.77	99.15
4	4.14	8.54	8.03	5.22	9.85	22.47
5	4.96	3.72	19.59	9.07	5.24	47.94
6	15.60	9.25	5.02	13.82	8.67	15.17
7	73.00	78.05	81.80	79.66	85.74	229.24
8	73.99	78.32	85.30	81.50	86.25	238.38
9	0.70	0.75	0.97	0.82	0.84	2.67
10	0.22	0.23	0.32	0.26	0.26	0.88

Note that the matrix in Equation 6.9 is applied to XYZ values without normalisation (c.f. CMCCAT97). CMCCAT2000 incorporates the degree of adaptation D and this is calculated by Equation 6.10:

$$D = F(0.08\log_{10}((L_{AT} + L_{AR})/2) + 0.76 - 0.45(L_{AT} - L_{AR})/(L_{AT} + L_{AR})) \quad (6.10)$$

where, as for CMCCAT97, the parameter F is 1.0 for average viewing conditions and 0.8 for dim and dark surround conditions. However, L_{AT} and L_{AR} are the luminances of the test and reference adapting fields respectively (note that CMCCAT97 did not account for the luminance of the reference field). The degree of adaptation D is then used with the ratios of the white points of the illuminants to convert the RGB values of the sample to the RGB values of the corresponding colour thus:

$$\begin{aligned} R_C &= (\alpha(R_{WR}/R_{WT}) + 1 - D)R \\ G_C &= (\alpha(G_{WR}/G_{WT}) + 1 - D)G \\ B_C &= (\alpha(B_{WR}/B_{WT}) + 1 - D)B \end{aligned} \quad (6.11)$$

where $\alpha = DY_W/Y_{WR}$ (Y_W and Y_{WR} are the Y values of the test and reference illuminants respectively). Note that when $D = 1$, the transform (Equation 6.11) is simply a diagonal transform of the RGB values. Finally, the corresponding RGB values are converted back to tristimulus values by multiplying them by the inverse of $\mathbf{M}_{\mathbf{CMCCAT2000}}$ to yield the normalised tristimulus values which can finally be converted by multiplying each by the Y tristimulus value of the sample under the test illuminant. The inverse matrix is given (Li *et al*., 2002) as:

$$\mathbf{M}^{-1}_{\mathbf{CMCCAT2000}} = \begin{bmatrix} 1.076450 & -0.237662 & 0.161212 \\ 0.410964 & 0.554342 & 0.034694 \\ -0.010954 & -0.013389 & 1.024343 \end{bmatrix} \quad (6.12)$$

The function `cmccat00` implements the CMCCAT2000 CAT. The function is called thus:

```
[xyzc] = cmccat00(xyz,xyzt,xyzr,lt,lr,f);
```

where **xyz** is an $n \times 3$ matrix of XYZ values for stimuli under the test illuminant, **xyzt** and **xyzt** are the tristimulus values of the test and reference illuminants respectively, **lt** and **lr** are the luminances of the adapting test and reference fields respectively, and **f** is normally unity. The function returns an $n \times 3$ matrix of corresponding tristimulus values.

```
% ======================================================
% *** FUNCTION cmccat00
% ***
% *** function [xyzc] = cmccat97(xyz,xyzt,xyzr,lt,lr,f)
% *** implements the cmccat2000 CAT
% *** xyz must be an n by 3 matrix of XYZ
```

```
% *** xyzt and xyzr contain white points
% *** f has default value of 1
% *** lt and lr are the luminances of the adapting
% *** test and reference fields and default to 100
% *** see also cmccat97
function [xyzc] = cmccat00(xyz,xyzt,xyzr,lt,lr,f)

if (size(xyz,2)~=3)
   disp('first input must be n by 3'); return;
end
if (length(xyzt)~=3 | length(xyzr)~=3)
   disp('check white point inputs'); return;
end

if (nargin<6)
   disp('using default values of la, lt and f')
   la = 100; lt = 100; f = 1;
end

xyzc = xyz; % to allocate memory
rgb = xyz;

% define the matrix for the transform to 'cone' space
M(1,:) = [0.7982 0.3389 -0.1371];
M(2,:) = [-0.5918 1.5512 0.0406];
M(3,:) = [0.0008 0.0239 0.9753];

% normalise xyz and transform to rgb
rgb = (M*xyz')';
rgbt = M*xyzt';
rgbr = M*xyzr';
  d = f*(0.08*log10((lt+lr)/2)+0.76-0.45*(lt-lr)/(lt+lr))
% clip d if it is outside the range [0,1]
if (d<0)
   d=0;
elseif (d>1)
   d=1;
end

% compute corresponding rgb values
a = d*xyzt(2)/xyzr(2)
rgbc(:,1) = rgb(:,1)*(a*rgbr(1)/rgbt(1) + 1 - d);
rgbc(:,2) = rgb(:,2)*(a*rgbr(2)/rgbt(2) + 1 - d);
rgbc(:,3) = rgb(:,3)*(a*rgbr(3)/rgbt(3) + 1 - d);

% define the matrix for the inverse transform
IM(1,:) = [1.076450 -0.237662 0.161212];
IM(2,:) = [0.410964 0.554342 0.034694];
IM(3,:) = [-0.010954 -0.013389 1.024343];
```

```
% implement step 4: convert from rgb to xyz
xyzc = (IM*rgbc')';
```

Table 6.2 lists some XYZ values under illuminant A and their corresponding values under illuminant D65 according to CMCCAT2000. The MATLAB® toolbox also includes a function for the reverse CMCCAT2000 which is called `CMCCAT00r` and called thus:

```
[xyz] = cmccat00(xyzc,xyzt,xyzr,lt,lr,f);
```

6.3 Colour-Appearance Models (CAMs)

CIELAB can be considered to be a CAM but it makes relatively poor predictions of colour appearance in most cases. Fairchild (2005) notes that CIELAB has no luminance level dependency and therefore cannot predict the Hunt[4] (Hunt, 1952), Bezold-Bruck[5] (Purdy, 1931), or Stevens[6] (Stevens and Stevens, 1963) effects. CIELAB also fails to predict simple contrast effects (e.g. Bartleson and Breneman, 1967) such as those illustrated in Figure 6.3. The CIELAB colour coordinates for the gray patches shown in Figure 6.3 are identical but their colour appearances vary greatly depending upon the background. Although the normalisation of the tristimulus values by the white point in the computation of CIELAB values does attempt to deal with some issues of colour appearance (those caused by the ability of the visual system to compensate for changes in the illumination) it does not contain any spatial component. Yet in everyday viewing, the colours that we see are almost always related colours. That is, we see colours in relation to the surrounding colours in a scene. A CAM should, for example, be able to predict an increase in lightness when a grey paper is viewed against a dark background compared to when it is viewed against a light background. For many technologies, of course, colour appearance is not important and basic colorimetry is sufficient (Berns, 2000). So, for example, if we wish to compare a trial fabric sample to a standard we are

Table 6.2 *XYZ values for 10 test samples under illuminant A and their corresponding values under illuminant D65 using CMCCAT2000. The luminance of the test and reference fields are both assumed to be 200 cd/m^2.*

Pair	X	Y	Z	X_C	Y_C	Z_C
1	19.41	28.41	11.57	17.30	30.35	34.02
2	22.48	31.60	38.48	27.90	36.27	106.19
3	28.99	29.58	35.75	32.69	32.77	99.15
4	4.14	8.54	8.03	5.22	9.85	22.47
5	4.96	3.72	19.59	9.07	5.24	47.94
6	15.60	9.25	5.02	13.82	8.67	15.17
7	73.00	78.05	81.80	79.66	85.74	229.24
8	73.99	78.32	85.30	81.50	86.25	238.38
9	0.70	0.75	0.97	0.82	0.84	2.67
10	0.22	0.23	0.32	0.26	0.26	0.88

[4] Whereby objects with higher luminance appear more saturated.
[5] Whereby objects' hues shift with changes in luminance.
[6] Whereby perceived contrast increases with luminance.

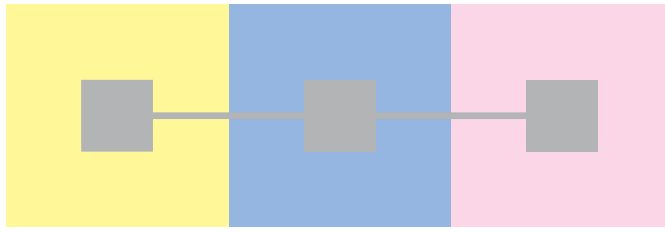

Figure 6.3 *The grey patches are all colorimetrically identical and yet have different colour appearances.*

often only concerned with whether the trial matches the standard rather than with what the two samples actually look like. The importance of this distinction is critical if the recent work on colour appearance is to be understood.

In 1997 the CIE Technical Committee TC1-34 agreed to adopt a colour-appearance model known as CIECAM97s (Luo and Hunt, 1998a). CIECAM97s comprises two parts: the first part is a CAT that computes the corresponding tristimulus values; the second part calculates a set of colour-appearance descriptors for the corresponding tristimulus values. The particular CAT that is used in CIECAM97s is known as CMCCAT97 (Luo and Hunt, 1998b) and this is described in Section 6.3. Although CIECAM97s was widely used in the colour-management industry a number of alternative models have been produced. There is still disagreement on the nature of the CAT that should be used (recall that CIECAM97s uses CMCCAT97). Nayatani *et al*. (1999) proposed an alternative CAT and a further transform $\mathbf{M}_{\mathbf{SHARP}}$ was developed based directly upon the principle of chromatic sharpening (Finlayson and Süsstrunk, 2000). Luo and his colleagues have developed CMCCAT2000 and it has been claimed that CMCCAT2000 gives a prediction to almost all of the available data sets that is more accurate than any of the other published transforms (Li *et al*., 2002). CMCCAT2000 is the CAT that forms the basis of a colour-appearance model known as CIECAM02.

CIECAM97s included a forward and reverse mode. The forward mode transforms the tristimulus values of a sample under a non-daylight illuminant, such as A, to those for the corresponding colour under a daylight illuminant (in fact, the equal-energy illuminant) and then computes some terms that describe the colour appearance of the sample under the daylight illuminant. The reverse mode is used for predicting the tristimulus values under a non-daylight illuminant based upon the colour-appearance descriptors for a sample viewed in daylight. The output of the forward mode is a set of attributes that predict colour appearance: brightness, lightness, colourfulness, chroma, saturation and hue (CIE, 1997). The predictions from the CIECAM97s model are in agreement with a number of colour-appearance phenomena such as chromatic adaptation (CMCCAT97 is included within CIECAM97s), Hunt's effect, Steven's effect (Stevens and Stevens, 1963), and the surround effect (Bartleson and Breneman, 1967). However, there are certain well-known colour-appearance phenomena that CIECAM97s cannot predict (Luo, 2002b).

In 2003 the CIE Technical Committee CIE TC 8-01 recommended a new colour appearance model to replace CIECAM97s known as CIECAM02. CIECAM02 has been shown to be simpler and more accurate then CIECAM97s (Li and Luo, 2005). It has also been shown to be able to provide a more uniform colour space (Luo *et al*., 2006). The robustness of CIECAM02 has been tested (Li and Luo, 2005) and some irregularities in CIECAM02 have been noted (Brill, 2006). In the following section CIECAM02 is

Table 6.3 *Parameters that depend upon viewing conditions in CIECAM02.*

Surround	F	c	N_c
Average	1.0	0.69	1.0
Dim	0.9	0.59	0.95
Dark	0.8	0.525	0.8

described and the MATLAB® code that implements it is introduced. However, it is important to understand that this is still a very fertile area of research; Nayatani and Sakai (2008) introduced a new type of CAM derived from CIELUV called In-CAM(CIELUV).

6.3.1 CIECAM02

The starting point for CIECAM02 is to determine the parameters that depend upon viewing conditions (Moroney *et al*., 2002). The surround is selected (average, dim or dark) and then the values of F, c and N_c are read from Table 6.3.

The value of F_L is then computed using Equation 6.12:

$$F_L = k^4 L_A + 0.1(1 - k^4)^2 (5L_A)^{1/3} \tag{6.13}$$

where $k = 1/(5L_A + 1)$ and L_A is the luminance of the adapting field. The luminance factors of the background Y_b and of the adopted white point Y_W are then used to calculate a ratio $n = Y_b/Y_W$ which in turn can be used to determine N_{bb}, N_{cb} and z.

$$N_{bb} = N_{cb} = 0.725(1/n)^{0.2} \tag{6.14}$$

$$z = 1.48 + \sqrt{n} \tag{6.15}$$

The following steps are then performed.

Step 1: Use a CAT to convert the *XYZ* values of the sample into sharpened *RGB* values. The CAT that is used is a modified form of CMCCAT2000 known as CAT02. The CAT02 matrix is given by Equation 6.16,

$$\mathbf{M_{CAT02}} = \begin{bmatrix} 0.7328 & 0.4296 & -0.1624 \\ -0.7036 & 1.6975 & 0.0061 \\ 0.0030 & 0.0136 & 0.9834 \end{bmatrix} \tag{6.16}$$

Step 2: Compute the degree of adaption D using the luminance of the adapting field L_A (in cd/m^2) and the surround condition F. Thus:

$$D = F\left[1 - (1/3.6)\left(\frac{-L_A - 42}{92}\right)\right] \tag{6.17}$$

Note that if full adaptation occurs, D should be set equal to 1; if no adaptation occurs D should be set equal to zero. A realistic minimum value is about 0.8 (Hunt and Pointer, 2011).

Step 3: The D factor is then used to convert the *RGB* values into corresponding values $R_C G_C B_C$:

$$R_C = [(DY_W/R_W) + 1 - D]R$$
$$G_C = [(DY_W/G_W) + 1 - D]G$$
$$B_C = [(DY_W/B_W) + 1 - D]B \tag{6.18}$$

where $R_W G_W B_W$ are for the adopted white under the test illuminant and obtained by multiplying $X_W Y_W Z_W$ by the matrix $\mathbf{M_{CAT02}}$.

Step 4: $R_C G_C B_C$ are next converted into Hunt-Pointer-Estevez space (Hunt and Pointer, 1985) according[7] to Equation 6.19:

$$\begin{bmatrix} \rho \\ \gamma \\ \beta \end{bmatrix} = \mathbf{M_H M_{CAT02}^{-1}} \begin{bmatrix} R_C \\ G_C \\ B_C \end{bmatrix} \tag{6.19}$$

The matrices $\mathbf{M_H}$ and $\mathbf{M_{CAT02}^{-1}}$ are defined as:

$$\mathbf{M_H} = \begin{bmatrix} 0.38971 & 0.68898 & -0.07868 \\ -0.22981 & 1.18340 & 0.04641 \\ 0.00000 & 0.00000 & 1.00000 \end{bmatrix} \tag{6.20}$$

$$\mathbf{M}_{CAT02}^{-1} = \begin{bmatrix} 1.096124 & -0.278869 & 0.182745 \\ 0.454369 & 0.473533 & 0.072098 \\ -0.009628 & -0.005698 & 1.015326 \end{bmatrix} \tag{6.21}$$

Step 5: A post-adaptation nonlinear response compression is then applied to $\rho\gamma\beta$ generating $\rho_a\gamma_a\beta_a$. The equation for ρ_a is shown below:

$$\rho_a = \left\{ \frac{400(F_L\rho/100)^{0.42}}{27.13 + (F_L\rho/100)^{0.42}} \right\} + 0.1 \tag{6.22}$$

Note that if ρ is negative then the positive equivalent must be used in the equation (Equation 6.22) and the expression inside the brackets {} must be made negative. Similar equations are used to obtain γ_a and β_a.. Values for $\rho_{aW}\gamma_{aW}\beta_{aW}$ are computed in a similar way.

Step 6: Preliminary Cartesian coordinates, a and b, are next computed which are then used for hue h:

$$a = \rho_a - 12\gamma_a/11 + \beta_a/11 \tag{6.23}$$
$$b = (\rho_a + \gamma_a - 2\beta_a)/9 \tag{6.24}$$
$$h = \tan^{-1}(b/a) \tag{6.25}$$

Step 7: Hue quadrature H is calculated based on the data in Table 6.4.
Set $h' = h + 360$ if $h < 20.14$; otherwise $h' = h$.
An appropriate value of i is selected so that $h_i \leq h' < h_{i+1}$ and then:

$$H = H_i + [100(h' - h_i)/e_i]/[(h' - h_i)/e_i + (h_{i+1} - h')/e_{i+1}] \tag{6.26}$$

Step 8: The eccentricity factor e_t is calculated:

$$e_t = (\cos(h\pi/180 + 2) + 3.8)/4 \tag{6.27}$$

[7] Note that $\mathbf{M_{CAT02}^{-1}}$ is used to convert to the corresponding colour to XYZ space and $\mathbf{M_H}$ is used to convert XYZ to cone space.

Table 6.4 *Unique hue data for hue quadrature.*

	Red	Yellow	Green	Blue	Red
i	1	2	3	4	5
h_i	20.14	90.00	164.25	237.53	380.14
e_i	0.8	0.7	1.0	1.2	0.8
H_i	0.0	100.0	200.0	300.0	400.0

Step 9: The achromatic responses for the stimulus A and white A_W are calculated:

$$A = (2\rho_a + \gamma_a + \beta_a/20 - 0.305)N_{bb} \tag{6.28}$$

$$A_a = (2\rho_{aw} + \gamma_{aw} + \beta_{aw}/20 - 0.305)N_{bb} \tag{6.29}$$

Step 10: Calculate the correlate of Lightness J:

$$J = 100(A/A_w)^{cz} \tag{6.30}$$

Step 11: Calculate the correlate of brightness Q:

$$Q = (4/c)(J/100)^{0.5}(A_w + 4)F_L^{0.25} \tag{6.31}$$

Step 12: Calculate the correlate of chroma C:

$$C = t^{0.9}(J/100)^{0.5}(1.64 - 0.29^n)^{0.73} \tag{6.32}$$

where:

$$t = [(50000/13)N_cN_{cb}][e_t(a^2 + b^2)^{0.5}]/[\rho_a + \gamma_a + 21\beta_a/20] \tag{6.33}$$

Step 13: Calculate the correlate of colourfulness, M:

$$M = CF_L^{0.25} \tag{6.34}$$

Step 14: Calculate the correlate of saturation s:

$$s = 100(M/Q)^{0.5}$$

The MATLAB® toolbox includes a function called `CIECAM02` and called thus:

```
[J,Q,C,M,s,h]=ciecam02(xyz,xyzw,la,yb,para)

function [J,Q,C,M,s,h]=ciecam02(xyz,xyzw,la,yb,para)

% function [j,c,hq,m,h,s,q]=ciecam02(xyz,xyzw,la,yb,para)
% implements the CIECAM02 colour appearance model
% operates on n by 3 matrix xyz containing tristimulus
```

```
% values of the stimulus under the test illuminant
% xyzw is a 1 by 3 matrix containing the
% white point for the test illuminant
% la and yb are the luminance and Y tristimulus values of
% the achromatic background against which the sample is viewed
% para is a 1 by 3 matrix containing f, c and Nc.
% J, Q, C, M, s and h are correlates of Lightness, Brightness,
% Chroma, colourfuness, saturation and hue

f = para(1); c = para(2); Nc = para(3);
MH = [0.38971 0.68898 -0.07868; -0.22981 1.18340 0.04641;
     0.0 0.0 1.0];
M02 = [0.7328 0.4296 -0.1624; -0.7036 1.6975 0.0061;
     0.0030 0.0136 0.9834];
Minv = [1.096124 -0.278869 0.182745; 0.454369 0.473533 0.072098;
     -0.009628 -0.005698 1.015326];

k = 1/(5*la+1);
fl = (k^4)*la + 0.1*((1-k^4)^2)*((5*la)^(1/3));
n = yb/xyzw(2);
ncb = 0.725*(1/n)^0.2;
nbb = ncb;
z = 1.48+sqrt(n);

% step 1
rgb = M02*xyz';
rgbw = M02*xyzw';

% step 2
D = f*(1-(1/3.6)*exp((-la-42)/(92)));

% step 3
rgbc(1,:) = (D*xyzw(2)/rgbw(1) + 1 - D)*rgb(1,:);
rgbc(2,:) = (D*(xyzw(2)/rgbw(2)) + 1 - D)*rgb(2,:);
rgbc(3,:) = (D*(xyzw(2)/rgbw(3)) + 1 - D)*rgb(3,:);

rgbwc(1) = (D*(xyzw(2)/rgbw(1)) + 1 - D)*rgbw(1);
rgbwc(2) = (D*(xyzw(2)/rgbw(2)) + 1 - D)*rgbw(2);
rgbwc(3) = (D*(xyzw(2)/rgbw(3)) + 1 - D)*rgbw(3);

% step 4
rgbp = MH*Minv*rgbc;
rgbpw = MH*Minv*rgbwc';

% step 5
rgbpa(1,:) = (400*(fl*rgbp(1,:)/100).^0.42)./
     (27.13+(fl*rgbp(1,:)/100).^0.42)+0.1;
rgbpa(2,:) = (400*(fl*rgbp(2,:)/100).^0.42)./
     (27.13+(fl*rgbp(2,:)/100).^0.42)+0.1;
```

```
rgbpa(3,:) = (400*(fl*rgbp(3,:)/100).^0.42)./
     (27.13+(fl*rgbp(3,:)/100).^0.42)+0.1;
rgbpwa(1) = (400*(fl*rgbpw(1,:)/100)^0.42)/
     (27.13+(fl*rgbpw(1)/100)^0.42)+0.1;
rgbpwa(2) = (400*(fl*rgbpw(2,:)/100)^0.42)/
     (27.13+(fl*rgbpw(2)/100)^0.42)+0.1;
rgbpwa(3) = (400*(fl*rgbpw(3,:)/100)^0.42)/
     (27.13+(fl*rgbpw(3)/100)^0.42)+0.1;

% step 6
a = rgbpa(1,:) - 12*rgbpa(2,:)/11 + rgbpa(3,:)/11;
b = (rgbpa(1,:) + rgbpa(2,:) - 2*rgbpa(3,:))/9;

% step 7
h = cart2pol(a, b);
h = h*180/pi;
h = (h<0).*(h+360) + (1-(h<0)).*h;

% step 8
ehH = [20.14 90.00 164.25 237.53 380.14; 0.8 0.7 1.0 1.2 0.8;
     0.0 100.0 200.0 300.0 400.0];
hh=h;
hh = (1-(hh<20.14)).*hh + (hh<20.14).*(360+20.14);
i=(hh>=20.14)+(hh>=90)+(hh>=164.25)+(hh>=237.53)+(h>=380.14);

H=ehH(3,i)+100.0*( (hh-ehH(1,i))/ehH(2,i) )/
     ( (hh-ehH(1,i))/ehH(2,i) + (ehH(1,i+1)-hh)/ehH(2,i+1));
et = (cos(2+h*pi/180)+3.8)/4;

% step 9
A = (2*rgbpa(1,:) + rgbpa(2,:) + rgbpa(3,:)/20 - 0.305)*nbb;
Aw = (2*rgbpwa(1) + rgbpwa(2) + rgbpwa(3)/20 - 0.305)*nbb;

% step 10
J = 100*(A/Aw).^(c*z);

% step 11
Q = (4/c)*((J/100).^0.5)*((Aw + 4)*fl^0.25);

% step 12
t = (Nc*ncb*50000/13)*((et.*(a.^2+b.^2).^0.5)./
     (rgbpa(1,:)+rgbpa(2,:)+21*rgbpa(3,:)/20));
C = (t.^0.9).*((J/100).^0.5)*(1.64-0.29^n)^0.73;

% step 13
M = C*fl^0.25;

% step 14
s = 100*(M./Q).^0.5;
```

7
Physiological Colour Spaces

7.1 Introduction

The system of colorimetry based on the CIE (1931) standard observer is still widely used in industry as it provides a convenient specification of the colour of a stimulus. For example, it is possible to reproduce a colour with acceptable approximation if its CIE (1931) *xy* chromaticity coordinates are known. Although this colorimetric information is based on the colour-matching performance of the ideal observer, it does not provide any information about the colour appearance of a stimulus. This is because the stimulus coordinates are not a direct representation of the physiological mechanisms that allow us to see that colour. The CIE (1931) chromaticity coordinates (and all the other CIE colour spaces which are based on it) provide numbers which have no physiological meaning. Several laboratories have attempted to describe the relationship between the colour matching functions (CMFs) and the physiological responses of the human visual system, also known as the cone fundamentals. These attempts have revealed important inaccuracies of the CIE (1931) CMFs. For these reasons, the widely accepted cone fundamentals estimated by Smith and Pokorny (1975)[1] and Stockman and Sharpe (2000) are based on the Judd-Vos modified CIE (1931) CMFs and the Stiles and Burch (1959) 10° CMFs, respectively. Perhaps, it would make more sense if colorimetry was actually based on cone fundamentals rather that CMFs, a thought that was also put forward by Boynton in his Frederic Ives Medal Paper address[2]. Of course, in 1931 this was not possible as cone fundamentals were not yet known. A positive step forward has certainly

[1] See also DeMarco, Pokorny and Smith (1992).
[2] The paper by Boynton (1996) offers a detailed review of the history of the CMFs.

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.

been the CIE adoption of the Stockman and Sharpe (2000) 2° cone sensitivities as the fundamental colour matching functions (CIE, 2007).

In this chapter we will describe how to use a colour representation that makes more explicit the contribution of the physiological mechanisms underlying the ideal observer's response and therefore adopt a physiologically meaningful colour space. For example, it is possible to describe a stimulus directly in terms of the excitations produced by the cones in response to that stimulus. In this case, the use of a cone-excitation space is recommended. It is also possible to specify the stimulus in terms of contrast which is based on a subsequent processing of the cone outputs. In this case, it is more appropriate to use an opponent colour space.

Our examples will show how to calculate physiological responses of stimuli presented on visual displays using MATLAB®. To obtain these physiological responses we need to take into account the spectral properties of the display device. In our calculations, we will be implementing the spectral properties of a CRT monitor, as this device is still widely used in vision laboratories. To calculate the physiological responses of a stimulus presented on a different device, the reader will simply need to replace the CRT's spectral power distributions with the requested device primaries.

7.2 Colour Vision

The absorption of photons by the photopigments located in the outer-segment of the photoreceptors can be considered as the first step of the visual processing. The probability of photon absorption varies with wavelength and with photoreceptor type. The photon absorption causes the photopigment molecule to isomerise. Such photopigment molecule is unable to discriminate which wavelength isomerised it, and thus it only signals that the event occurred. As a consequence, lights of different wavelengths with equal probability of been absorbed, cause an equal amount of isomerisation. This effect was first described by the visual physiologist William Rushton (1972) who named it *the principle of univariance*. Let us consider the implications of this principle. If we had only one type of photopigment which was exposed to two monochromatic lights with equal probability of being absorbed, we would be unable to distinguish them. If their probability was different, then we would be able to tell them apart based on the amount of isomerisation each one has caused. We would still be unable to distinguish them in terms of colour, as all the information about their wavelength is lost. This is precisely what happens during scotopic vision, when only rods can respond to light. Rods can only signal a difference of intensity between two lights that differ in absorption probabilities. This is why at scotopic levels, the world appears colourless. Interestingly, cones are as colour-blind as rods, as they are involved in the same isomerisation process. In fact, the ability to see colours is due to the fact that cones have different sensitivities to wavelengths. Thus, the responses to two monochromatic lights are likely to cause different triplets of cone responses. These cone responses go through a second stage of colour coding forming three distinct post-receptoral channels: the (L + M) luminance channel made of additive inputs from the L- and M-cones, the (L-M) chromatic channel made of opponent inputs from the L- and M-cones, and the second chromatic channel S-(L + M) made of opponent inputs from the S- and the (L+M)-cones.

Cone spectral sensitivities have been measured in terms of photon absorption curves by a variety of techniques. The curves obtained with psychophysical methods are very similar to those obtained with more direct techniques of retina densitometry, microspectrophotometry, and suction electrophysiology[3]. These techniques are considered to be more direct because they measure the actual spectral absorbance or sensitivity of each individual cone. In this chapter, we will consider the cone spectral sensitivities estimated using psychophysical methods, as they are believed to provide more accurate data and are also more consistent with the psychophysical methods considered here.

There are three types of cone spectral sensitivity, according to whether they absorb primarily in the short-, medium-, or long-wavelength range of the visible spectrum. We will refer to their corresponding cone types as the S-, M- and L-cone, respectively. The S-cone peak sensitivity lies in the violet (ca. 440 nm), the M-cone in the green (ca. 540 nm) and the L-cone in the yellow-green (ca. 570 nm) range.

There are several intriguing questions related to human cone spectral sensitivities. For example, why there are only three cone-types and not two or four, or even more? Why are their sensitivities broad and not narrow? Why do they overlap? Why do they peak at those specific wavelengths? And finally, why do different cone types combine their responses into post-receptoral channels? Despite the captivating discussion that we could initiate answering those questions, given the limited space we can dedicate in this book to theoretical issues, we can regrettably offer only brief answers. For the curious reader we provide a few references which have represented the most influential and provocative papers published on the subject.

In his seminal paper, Barlow (1982) conducted a careful analysis of the reasons for trichromacy and suggested that the useful number of different colour mechanisms is three, and that this dimensionality is highly constrained by the bandwidth of the three mechanisms themselves. In other words, we have three cone classes because of the broad width of their spectral curves. If their spectral curves were narrower, it would have been equally beneficial to have evolved more cone classes: but they did not. Then what is the advantage of having broad spectral curves? Although broad curves would cause a loss of discrimination between narrow-band stimuli, Barlow argued that this loss would be less for broad spectral band stimuli likely to be caused by most natural pigments. They would also absorb more light which would in turn improve their sensitivity. He also suggested that the positioning of their peak, in particular the close vicinity of the M-cone peak to the L-cone peak, would improve spatial discrimination as it would enable the two types of cone to be used without distinction.

It has been suggested that the reason why the M- and L-cone spectral sensitivities are so similar, relies on a random gene mutation that occurred during the early history of the catharrhine primates. As described by Mollon (1989)

> . . . either the two resulting genes then diverged so as to code for the present L and M pigments, or the original single gene was already polymorphic, as it is today in many New World species, and the two alternative forms of the gene became established on a single chromosome.

[3] For a review of these methods, see Chapter 5 in Kaiser and Boynton (1996).

According to some, this mutation revealed to be advantageous in finding edible fruit among leaves (Mollon, 1989; Osorio and Vorobyev, 1996; Dominy and Lucas, 2001).

Although the similarity between the M- and L-cone spectral sensitivities seems to be an advantage, it leads to a high degree of correlation in the activities of the M- and L-cones and therefore to a high degree of redundancy in the information carried by the cones. The redundancy problem seems to be solved by the opponent coding of the cone outputs which, according to Buchsbaum and Gottschalk (1983), has the ability to compress the information transmitted from the cones to the brain, and thus significantly reduce the amount of redundancy.

Since trichromacy was accepted as the principle behind human colour vision, several scientists have attempted to find out the relationship between the CMFs and the cone spectral sensitivities. Despite the difference in the cone spectral sensitivities measured by various laboratories, they all agree that a linear transformation must exist between CMFs and cone spectral sensitivities. In other words, given a colour match, there must be a linear transformation that relates that match to the sensitivities of the cones. The crucial differences between the various spectral sensitivity sets are largely caused by the different experimental techniques used to obtain those measurements, the assumptions made by the scientists (e.g. macular and optical densities) and the colour matching functions that are used to derive the corresponding linear transformations. There is an infinite number of possible mathematical transformations from colour matching functions to cone sensitivities, which is made even more complicated by the extensive overlap between the L- and M-cone sensitivities. A successful approach to the problem is to make use of König's hypothesis (König and Dieterici, 1886) according to which hereditary dichromacy, simply consists in a reduction of normal trichromacy. In practical terms, this means that to isolate a cone class it is sufficient to choose observers that lack one or more cone classes, as their spared cone class(es) is identical to normals. For example, to isolate S-cone sensitivities, colour matching data from blue – cone monochromats (observers missing the M- and L-cones) may be used. Fundamentals that use dichromatic data are known as König fundamentals.

Figure 7.1 illustrates a set of König fundamentals as measured by Stockman and Sharpe (2000). The Stockman and Sharpe 2° cone spectral sensitivities are defined as linear combinations of the Stiles and Burch (1959) 10° CMFs. These fundamentals will be used in the next paragraphs to generate colour representations that are physiologically meaningful. Obviously, it is possible to use different sets of cone fundamentals, such as DeMarco, Pokorny and Smith (1992), or Vos, Estévez and Walraven (1990), which are available in various formats at www.cvrl.org[4].

7.3 Cone-Excitation Space

In vision science, a light stimulus is often described in terms of quantum catches absorbed by each individual cone class. The advantage consists in providing information

[4] The CVRL database consists of a vast number of free downloadable standard data sets relevant to colour and vision research. For example, it provides tabulated values of cone fundamentals, colour matching functions, chromaticity coordinates, prereceptoral filter density spectra, photopigment spectra and CIE standards.

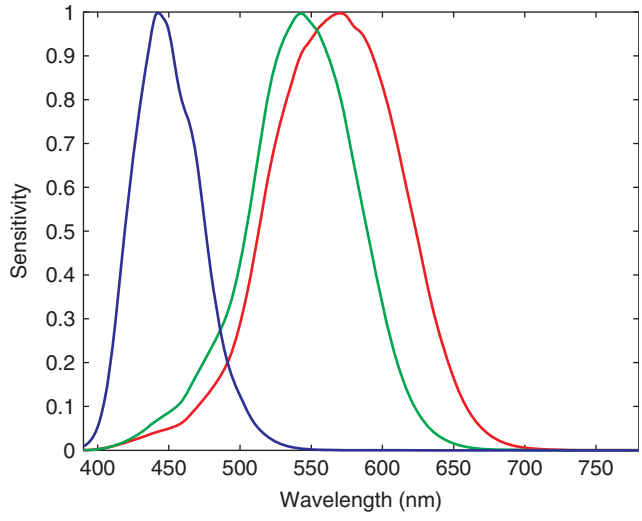

Figure 7.1 *Stockman and Sharpe 2° cone fundamentals of the short-, medium-, and long-wavelength sensitive cones in energy units normalised to 1 (Stockman and Sharpe, 2000).*

of the first-stage of vision without making assumptions on how these cone responses (or excitations) might be subsequently processed. For a given visual stimulus, the corresponding quantum catches will be equal to the mathematical integration over the visible spectrum of the product between the cone sensitivities and the spectral power distribution of the visible stimulus, such that:

$$
\begin{aligned}
l &= \sum_{\lambda=390}^{\lambda=780} L(\lambda)C(\lambda) \\
m &= \sum_{\lambda=390}^{\lambda=780} M(\lambda)C(\lambda) \\
s &= \sum_{\lambda=390}^{\lambda=780} S(\lambda)C(\lambda)
\end{aligned}
\tag{7.1}
$$

where l, m and s represent the quantum catches or cone excitations of the long-, medium- and short-wavelength-sensitive cones, whose sensitivities are indicated in the above formula as $L(\lambda)$, $M(\lambda)$ and $S(\lambda)$, respectively. $C(\lambda)$ is the spectral power distribution of the visual stimulus which can also be described in terms of colour mixture of the light emitted by the red, green, and blue phosphors of the CRT as follows:

$$
C(\lambda) = rR(\lambda) + gG(\lambda) + bB(\lambda) \tag{7.2}
$$

where $R(\lambda)$, $G(\lambda)$, and $B(\lambda)$ represent the spectral power distributions of the red, green and blue phosphors respectively, and the constants r, g and b represent the proportion of light emitted by each phosphor to generate the stimulus $C(\lambda)$. For example, to display a white stimulus the corresponding r, g and b values would all be equal to 1, and to

display a mid-grey stimulus the corresponding values would all be equal to 0.5. Equation 7.2 can be written in terms of linear algebra as:

$$\mathbf{e} = \mathbf{FC} \tag{7.3}$$

where $\mathbf{e}$ is a 3×1 column matrix representing the l, m and s cone-excitations, $\mathbf{F}$ is a $3 \times n$ matrix corresponding to the $L(\lambda)$, $M(\lambda)$ and $S(\lambda)$ cone fundamentals, and $\mathbf{C}$ is an $n \times 1$ column matrix representing $C(\lambda)$. The value of n will depend on the sampling interval. For example, if the spectral measurements are sampled from 390 nm to 780 nm, every 1 nm, then n will be equal to 391. $\mathbf{C}$ can also be written as:

$$\mathbf{C} = \mathbf{Pt} \tag{7.4}$$

where $\mathbf{P}$ is the $n \times 3$ matrix containing the $R(\lambda)$, $G(\lambda)$ and $B(\lambda)$ spectral power distributions of the phosphors and $\mathbf{t}$ is a 3×1 column vector containing the r, g and b values or display coordinates. Therefore, Equation 7.3 becomes:

$$\mathbf{e} = \mathbf{FPt} \tag{7.5}$$

which can be simplified further as:

$$\mathbf{e} = \mathbf{Mt} \tag{7.6}$$

where $\mathbf{M}$ is 3×3 matrix also known as the matrix transform. Equation 7.6 demonstrates that there is a linear transform between the cone excitations and the display coordinates. Given a set of cone excitations, it is possible to compute the corresponding display coordinates by considering the inverse of the matrix transform $\mathbf{M}$, such that:

$$\mathbf{t} = \mathbf{M}^{-1}\mathbf{e} \tag{7.7}$$

Note that the information about the spectral power distribution of the stimulus is lost. CRT monitors make use of the principle of additive colour mixing to display colorimetric metamers. This means that, two metameric colour signals that are characterised by different spectral power distributions, but that generate the same cone excitations, will be specified by the same display coordinates. It is important to note that our calculations assume that the output of the visual display is supposed to be linear, which in the case of a CRT means that we assume that the display has been gamma-corrected.

The function `rgb2lms` computes the cone excitations from a set of display coordinates. The inputs to the function consist of the $n \times 3$ matrix `phosphors`, which represents the spectral power distributions of the phosphors, the $n \times 3$ matrix `fundamentals`, which represents the cone fundamentals, and the 3×1 column matrix `rgb`, which represents the r, g and b values of the stimulus. The colour toolbox provides the above spectral data in two separate `.mat` files: `phosphors` and `fundamentals_ss`. The file `phosphors.mat`, contains the spectral power distribution in radiance units of the red, green and blue phosphors of a gamma-corrected 21″ Sony F520 Trinitron monitor sampled in the range $\{390, 780\}$ nm every 1 nm, and plotted in Figure 7.2. The file `fundamentals_ss.mat`, contains the Stockman and Sharpe 2° cone fundamentals in energy units normalised to 1 and sampled in the range $\{390, 780\}$ nm every 1 nm and

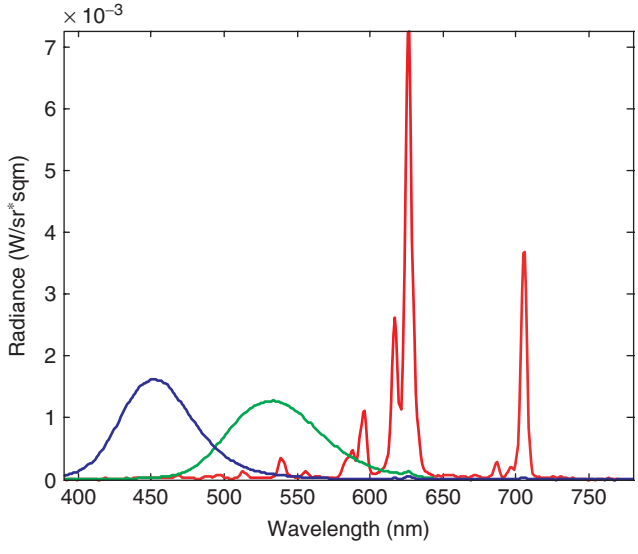

Figure 7.2 *Spectral power distribution in radiance units (W/sr*m^{-2}) of the red, green and blue phosphors of a 21″ Sony F520 Trinitron monitor.*

plotted in Figure 7.1. Saving these spectral data into separate `.mat` files gives more flexibility to use different spectral data.

Let us suppose we want to compute the cone excitations of a white stimulus whose r, g and b values are equal to 1. First, we need to load the inputs to the function `rgb2lms` in the MATLAB® workspace, and set the `rgb` stimulus values. Thus:

```
load phosphors
load fundamentals_ss
rgb=[1; 1; 1];
```

Then, we call `rgb2lms`:

```
[lms] = rgb2lms(phosphors,fundamentals,rgb)
```

where `rgb` represents the 3×1 column matrix $\mathbf{t}$ and `lms` represents the 3×1 column vector $\mathbf{e}$ described in Equation 7.6. The code for `rgb2lms` is shown below:

```
% ================================================================
% *** FUNCTION rgb2lms
% ***
```

```
% *** function [lms] = rgb2lms(phosphors,fundamentals,rgb)
% *** computes lms from rgb.
% *** phosphors is an n by 3 matrix containing
% *** the three spectral power distributions of the
% *** display device
% *** fundamentals is an n x 3 matrix containing
% *** the lms are the cone spectral sensitivities.
% *** the rgb are the rgb values of the display device.
% ================================================================
function [lms] = rgb2lms(phosphors,fundamentals,rgb)

% Compute lms from rgb.
rgbTOlms = fundamentals'*phosphors;
lms      = rgbTOlms * rgb;
% ================================================================
% *** END FUNCTION rgb2lms
% ================================================================
```

which will give in output:

```
lms =

  0.1318
  0.1098
  0.0750
```

The obtained cone-excitation values represent a proportion of the quantal absorption rates of the three types of cone. To compute the absolute quantal absorption rates, it is necessary to replace the cone fundamentals in energy units used in our example with the cone fundamentals in quantal units[5].

It is also possible to compute the display coordinates of a known set of cone excitations using the function `lms2rgb`.

```
% ================================================================
% *** FUNCTION lms2rgb
% ***
% *** function [rgb] = lms2rgb(phosphors,fundamentals,lms)
% *** computes rgb from lms.
% *** phosphors is an n by 3 matrix containing
% *** the three spectral power distributions of the
% *** display device
```

[5] Cone fundamentals in quantal units are available at www.cvrl.org.

```
% *** fundamentals is an n x 3 matrix containing
% *** the lms are the cone spectral sensitivities.
% *** the rgb are the rgb values of the display device.
% ========================================================================
function [rgb] = lms2rgb(phosphors,fundamentals,lms)

% Compute lms from rgb.
rgbTOlms = fundamentals'*phosphors;
lmsTOrgb = inv(rgbTOlms);
rgb      = lmsTOrgb * lms;
% ======================================================
% *** END FUNCTION lms2rgb
% ======================================================
```

which will return, for example:

```
rgb =

  1.0000
  1.0000
  1.0000
```

7.4 MacLeod and Boynton Chromaticity Diagram

In 1979 MacLeod and Boynton introduced a chromaticity diagram which was based on the relative responses of the L-, M- and S-cone excitations (MacLeod and Boynton, 1979). As the underlying concept was similar to that of Robert Luther, the authors called it the Luther diagram, although it has become known as the MacLeod and Boynton chromaticity diagram. The diagram has several advantages which make it a very desirable colour representation in visual experiments: first, it is based on physiologically-meaningful responses which are directly represented in the diagram; second it can be represented in only two dimensions, which are more easily visualised than three. The main drawback is that it relies on the assumption that S-cones do not contribute to luminance, which has been demonstrated not to be true under some conditions (Ripamonti *et al*., 2009). The two dimensions of the diagram are represented by the relative responses of the L- and M-cones along the abscissa, r_{MB}, and the relative responses of the S-cones along the ordinate, b_{MB}. The following equations define the quantities proposed in the chromaticity diagram:

$$\begin{aligned} r_{MB} &= \bar{l}/(\bar{l}+\bar{m}) \\ g_{MB} &= \bar{m}/(\bar{l}+\bar{m}) \\ b_{MB} &= \bar{s}/(\bar{l}+\bar{m}) \end{aligned} \tag{7.8}$$

where r_{MB}, g_{MB} and b_{MB} represent the MacLeod and Boynton chromaticities, $\bar{l}$, $\bar{m}$ and $\bar{s}$ are the cone excitations scaled so that luminance is equal to $\bar{l}+\bar{m}$ and b_{MB} is scaled

to unity. When using Stockman and Sharpe (2000) 2° fundamentals in energy units, the cone excitations are scaled as follows:

$$\begin{aligned} \bar{l} &= l/0.689903 \\ \bar{m} &= m/0.348322 \\ \bar{s} &= s/0.371597 \end{aligned} \tag{7.9}$$

where l, m and s correspond to the L-, M- and S-cone excitations.

Figure 7.3 shows the MacLeod and Boynton chromaticity diagram plotted as b_{MB} versus r_{MB}. The abscissa arbitrarily represents the L-cone excitations as a trade-off with respect to the M-cone excitations, such that:

$$r_{MB} + g_{MB} = 1 \tag{7.10}$$

Thus, it is sufficient to plot the value of r_{MB} as the value of g_{MB} can be derived easily from:

$$g_{MB} = 1 - r_{MB} \tag{7.11}$$

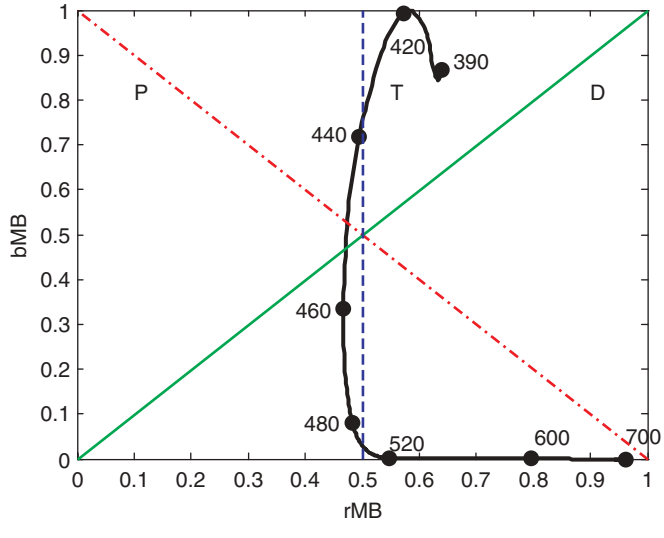

Figure 7.3 *MacLeod and Boynton chromaticity diagram (1979) based on Stockman and Sharpe (2000) 2° fundamentals. The red dotted line labelled P, which radiate from the right-hand side corner, represents a confusion line for protanopes. The green continuous line labelled D, which radiates from the left-hand side corner represents a confusion line for deuteranopes. The blue dashed line parallel to the ordinate labelled T, represents the confusion line for tritanopes.*

The origin of the diagram, where $r_{MB} = 0$ and $s_{MB} = 0$, represents a colour that would excite only the M-cones. The right-hand corner of the diagram, where $g_{MB} = 0$ and $s_{MB} = 0$, represents a colour that would excite only the L-cones. There is no analogous corner representing a colour that excites only the S-cones (MacLeod and Boynton, 1979). From the right and left corner of the abscissa, it is possible to draw confusion lines, which represent the loci of those colours that can be confused by dichromats. For example, deuteranopic observers missing M-cones will confuse those colours whose chromaticities lie along the line radiating from the left-hand side corner (D line in Figure 7.3). This is because those chromaticities correspond to stimuli that equally excite the L- and S-cones. Observers with normal trichromatic vision are able to distinguish those colours because they rely on the different excitations provided by the M-cones. Similarly, protanopic observers would be unable to distinguish the colours lying along the dotted line radiating from the right corner (P line in Figure 7.3), because they produce identical M- and S-cone excitations. Protanopes, who lack L-cones, would confuse those points as they would be unable to rely on the different excitations provided by the L-cones. Tritanopes, who lack S-cones, would confuse those colours that lie along vertical lines parallel to the ordinate, along which the L- and M-cone excitations are identical (T line in Figure 7.3). These confusion lines rely on the assumption that dichromatic vision is a reduced form of colour vision, and thus dichromatic observers accept trichromatic matches of normal observers (Kaiser and Boynton, 1996). The reader interested in learning more about the correspondence between the confusion lines of the MacLeod and Boynton diagram and the CIE (1931) chromaticity diagram can consult the paper by Capilla *et al*. (1998) which provides useful mathematical transformations that allow redrawing the confusion lines in the MacLeod and Boynton diagram into the CIE (1931) diagram.

Given a set of L-, M-, and S-cone excitations, the function `lms2rgbMB` computes the corresponding chromaticity coordinates in the MacLeod and Boynton diagram. The function takes two inputs: `lms`, which is a 3×1 matrix representing the L-, M- and S-cone excitations, and `s` which is a 1×3 matrix representing the scaling factors [0.689903 0.348322 0.0371597] described in Equation 7.9. These scaling factors depend on the cone fundamentals used to derive the cone excitations. The above scaling factors are specific for the Stockman and Sharpe (2000) 2° cone fundamentals and can be omitted when calling the function `lms2rgbMB` as they are defined as the default scaling functions in `lms2rgbMB`. If the cone excitations provided in input derive from different cone fundamentals, then the row matrix `s` needs to be defined.

Let us suppose we want to compute the MacLeod and Boynton chromaticity coordinates that correspond to the cone excitations `lms` of the white stimulus defined in the previous paragraph. In the MATLAB® command window we type:

```
lms = [0.1318; 0.1098; 0.0750];
[rgbMB] = lms2rgbMB(lms)
```

The code for `lms2rgbMB` is shown below:

```
% ============================================================
% *** FUNCTION lms2rgbMB
% ***
% *** function [rgbMB] = lms2rgbMB(lms,s)
% *** computes rgbMB from lms.
% *** lms is an n by 1 matrix containing
% *** the l-,m-, and s-cone excitations
% *** s is a 1 by n matrix containing the lms scaling factors.
% *** rgbMB are the chromaticity coordinates in the
% *** MacLeod and Boynton (1979) chromaticity diagram.
% ============================================================
function [rgbMB] = lms2rgbMB(lms,s)
% check number of input arguments.
% if s is not provided, use default scaling factors valid
% for the Stockman and Sharpe (2000) 2-deg fundamentals.
% else use s.
if nargin==1
    lms_scaling=[0.689903 0.348322 0.0371597];
else
    lms_scaling = s;
end
% scale cone excitations.
lms = diag(lms_scaling')*lms;
% compute luminance
luminance = lms(1)+ lms(2);
% compute rgbMB chromaticities.
rMB = lms(1)/luminance;
gMB = lms(2)/luminance;
bMB = lms(3)/luminance;
rgbMB = [rMB gMB bMB];
% ====================================================
% *** END FUNCTION lms2rgbMB
% ====================================================
```

The function will return:

```
rgbMB =
    0.7039    0.2961    0.0216
```

where the row matrix contains the r_{MB}, g_{MB} and b_{MB} coordinates of the MacLeod and Boynton chromaticity diagram. It is possible to compute the L-, M- and S-cone excitations from a given set of MacLeod and Boynton chromaticity coordinates, providing that the luminance quantity $\bar{l} + \bar{m}$ used to scale the cone excitations is known. In this case, we can call the function `rgbMB2lms` which takes three inputs: `rgbMB`, which is a $1 \times n$ matrix representing the r_{MB}, g_{MB} and b_{MB} chromaticity coordinates of the MacLeod and

Boynton diagram, `luminance`, which is the scalar $\bar{l} + \bar{m}$ as described in Equation 7.8, and `s`, which is a $1 \times n$ matrix containing the scaling factors for the L- M- and S- cone excitations. A typical call to the function would be:

```
[lms] = rgbMB2lms(rgbMB,luminance,s)
```

The code for `rgbMB2lms` is shown below:

```
% ================================================================
% *** FUNCTION rgbMB2lms
% ***
% *** function [lms] = rgbMB2lms(rgbMB,luminance,s)
% *** computes lms from rgbMB.
% *** lms is a 1 by n matrix containing
% *** the l-,m-, and s-cone excitations
% *** rgbMB is a 1 by n matrix which represents
% *** the chromaticity coordinates in the
% *** MacLeod and Boynton (1979) chromaticity diagram.
% *** luminance is the sum of the scaled l_bar and m_bar
% *** excitations.
% *** s is a 1 by n matrix containing the lms scaling factors.
% ================================================================
function [lms] = rgbMB2lms(rgbMB,luminance,s)
% check number of input arguments.
% if s is not provided, use default scaling factors valid
% for the Stockman and Sharpe (2000) 2-deg fundamentals.
% else use s.
if nargin==2
    lms_scaling=[0.689903 0.348322 0.0371597];
else
    lms_scaling = s;
end
% compute scaled LMS from rgbMB and luminance.
lms_scaled = rgbMB.* luminance
% to obtain LMS excitations the previous LMS
% *** values need to be unscaled.
% define LMS scaling according to which fundamentals are used.
lms = lms_scaled./(lms_scaling);
% ================================================================
% *** END FUNCTION lms2rgbMB
% ================================================================
```

Thus, if:

```
rgbMB = [0.7039 0.2961 0.0216];
```

the function `rgbMB2lms` will return:

```
lms =

    0.1318   0.1098   0.0750
```

which is identical to the L-, M- and S- cone excitations previously used as input to the function `lms2rgbMB`.

7.5 DKL Colour Space

In 1984, Derrington, Krauskopf and Lennie measured the ganglion cell responses in the lateral geniculate nucleus of the macaque and proposed a new method for distinguishing between responses originating from chromatically opponent cone outputs and responses originating from achromatic additive cone outputs. This led to the development of a new colour space, which is partly based on ideas developed by MacLeod and Boynton (1979) and Krauskopf, Williams and Heeley (1982). This space became known as the DKL colour space after Derrington *et al*. (1984). The DKL colour space is recommended particularly when the test stimulus is seen against a background. The underlying idea is that the background provides the adaptation level according to which the relative excitations caused by the test stimulus can be calculated. In other words, the test stimulus is represented in terms of contrast against the background. There are several ways of calculating the contrast between two lights, and the choice depends on the assumptions relative to the underlying mechanism regulating such contrast. The DKL colour space is based on a classic formula for contrast, according to which the contrast C is equal to:

$$C = \frac{(T - B)}{B} \tag{7.12}$$

or simply:

$$C = \frac{\Delta T}{B} \tag{7.13}$$

where T represents the test stimulus and B the background. As suggested by von Kries (1905), a given adapting background can scale the cone excitations independently and by different amounts. This means that the contrast needs to be calculated within each cone-class as follows:

$$\begin{bmatrix} C_L \\ C_M \\ C_s \end{bmatrix} = \begin{bmatrix} \frac{\Delta T_L}{B_L} \\ \frac{\Delta T_M}{B_M} \\ \frac{\Delta T_S}{B_S} \end{bmatrix} \tag{7.14}$$

where C_L, C_M and C_S represent the normalised cone contrasts between the cone excitations T_L, T_M and T_S of the target and the cone excitations B_L, B_M and B_S of the background. The quantities ΔT_L, ΔT_M and ΔT_S represent the individual cone-excitation differences between T and B.

Sometimes it is convenient to define a single index that pools together the individual cone contrasts, as described in Equation 7.15 (Chaparro *et al*., 1993):

$$C = \sqrt{C_L^2 + C_M^2 + C_S^2} \tag{7.15}$$

where C represents the pooled cone contrast of the normalised contrasts for the three cone classes. As pointed out by Brainard (1996):

> using pooled cone contrast as a measure of signal strength has the attractive feature that is independent of apparatus, observer, and stimulus configuration details ... although it carries the unusual feature that the maximum achievable pooled contrast is equal to $C_{\max} = \sqrt{3}$ instead of the conventional $C_{\max} = 1$.

As the DKL colour space is based on post-receptoral responses, the contrast will be calculated within each post-receptoral mechanism and not within each cone class, and the overall contrast will correspond to the pooled contrast of all mechanisms.

There are several advantages for using the DKL colour space, among which: (i) it is based on physiological responses represented by the post-receptoral pathways; (ii) these responses, are assumed to be independent and thus can be easily isolated and used to test in which proportion each signal contributes to a given phenomenon. On the other hand, the space makes strong assumptions on how the cone responses combine their outputs to form the post-receptoral pathways and assumes that those combinations are linear. Also, when the background is nonuniform, it is sometimes difficult to decide how to compute the level of adaptation of the background.

In the DKL colour space a given stimulus target is specified by a set of three-dimensional coordinates which identify the relative spatial location of the target with respect to the adapting background. It is important to remember that the DKL coordinates are relative coordinates with respect to the arbitrary chosen adaptation level. In fact, if the adaptation level changes the DKL coordinates of the target might change too.

Here, we show how to compute the DKL coordinates of a stimulus target T seen against an adapting background B[6]. Our ultimate goal is to be able to represent the target T in terms of relative contrast against the adaptation level B. According to the assumptions mentioned above, the space is three-dimensional and defined by a vertical axis along which only luminance varies, without change in chromaticity (isochromatic L + M axis); a chromatic axis orthogonal to the previous, along which only the L- and M-cone excitations vary and the S-cone excitations are constant (isoluminant L-M axis); and a second chromatic axis, orthogonal to the previous two, along which only the S-cone excitations vary, and the L- and M-cone excitations are constant (isoluminant S-(L + M) axis). The three intercepting axes pass through the centre of the space, which corresponds to an arbitrarily designated white point represented by the background. This point does not necessarily appear white, or achromatic. It can appear of any colour as specified by the L-, M-, and S-cone excitations of the background. In fact, if the reader

[6] We are deeply indebted to David H. Brainard's Appendix (Brainard, 1996) which offers both the theoretical background and the practical implementation of the DKL colour space upon which our MATLAB® functions are based. We also recommend reading the paper by Capilla *et al*. (1998), which is a nice complement to Brainard's appendix.

is interested in using the chromatic adaptation technique to desensitise a specific cone-class, than the background will be set to a bright and saturated colour that maximally adapt that particular cone class. In the DKL colour space, stimuli are usually specified in terms of polar coordinates, whereby the L-M axis corresponds to $0-180^{\circ}$ azimuth, the S axis to $90-270^{\circ}$ azimuth, and the luminance axis to elevation from -90° to $+90^{\circ}$ respect to the isoluminant planes (Figure 7.4). The radius of each axis represents the relative excitation of each mechanism.

Thus, given a target T and a background B, their difference in terms of cone-excitations will correspond to ΔT_L, ΔT_M and Δ_S^2 as described in Equations 7.13 and 7.14. The corresponding DKL coordinates will be equal to these differential cone excitations normalised by the pooled contrast such that:

$$\begin{bmatrix} \Delta(L+M) \\ \Delta(L-M) \\ \Delta S-(L+M) \end{bmatrix} = \begin{bmatrix} k_{L+M} \\ k_{L-M} \\ k_{S-(L+M)} \end{bmatrix} \times \begin{bmatrix} w_{L+M,L} & w_{L+M,M} & W_{L+M,S} \\ w_{L-M,L} & w_{L-M,M} & W_{L-M,S} \\ w_{S-(L+M),L} & w_{S-(L+M),M} & W_{S-(L+M),S} \end{bmatrix} \begin{bmatrix} \Delta T_L \\ \Delta T_M \\ \Delta T_S \end{bmatrix} \tag{7.16}$$

where $\Delta(L+M)$ corresponds to the difference in luminance between the target and the background, $\Delta(L-M)$ corresponds to the difference in chromaticity along the isoluminant $L-M$ axis, and $\Delta S-(L+M)$ corresponds to the difference in chromaticity

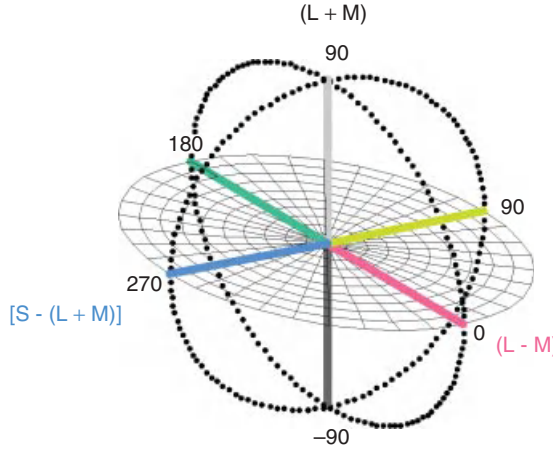

Figure 7.4 *The DKL colour space is defined by a vertical axis (L + M) along which only luminance varies; a chromatic axis (L-M) along which only the L- and M-cone excitations vary; and a second chromatic axis defined as S-(L + M), along which only the S-cone excitations vary. The three intercepting axes pass through the centre of the space, which corresponds to an arbitrarily designated white point with DKL coordinates [0 0 0]. The coloured solid illustrated in the figure represents the colour gamut of a 21″ Sony F520 Trinitron monitor. Only the DKL coordinates that lie inside this solid can be accurately reproduced by this monitor.*

along the isoluminant $S-(L+M)$. The constants k_{L+M}, k_{L-M}, $k_{S-(L+M)}$ are the normalisation factors for each mechanism and the weights of the 3×3 matrix are the proportions in which each cone class contributes to each mechanism. Equation 7.16 can be simplified when implementing the following assumptions (Derrington *et al*., 1984). The luminance mechanism consists of additive inputs only from the L- and M-cones and no contribution from the S-cones, which results in $w_{(L+m),S}$ equal to zero. The isoluminant $L-M$ mechanism is a chromatic mechanism, therefore when the differential cone excitations are identical to the background, the mechanism response will be zero. Furthermore, this mechanism is only based on opposed signals from the L- and the M-cones, with no contribution of the S-cones, which means that $w_{(L-M),S}$ is equal to zero. Similarly, the isoluminant $S-(L+M)$ mechanism will be zero when the differential cone excitations are identical to the background, and when ΔT_s and $\Delta(L+M)$ are zero. In this case, it follows that: $w_{S-(L+M),L}=w_{(L+M),L}$, $w_{S-(L+M),M}=w_{(L+M),M}$, and $w_{S-(L+M),S}=(w_{S-(L+M),L}B_L+w_{S-(L+M),M}B_M)/B_S$.

Given the above assumptions, Equation 7.16 can be simplified as:

$$\begin{bmatrix}\Delta(L+M)\\ \Delta(L-M)\\ \Delta S-(L+M)\end{bmatrix}=\begin{bmatrix}k_{L+M}\\ k_{L-M}\\ k_{S-(L+M)}\end{bmatrix}\begin{bmatrix}1 & 1 & 0\\ 1 & -\dfrac{B_L}{B_M} & 0\\ -1 & -1 & \dfrac{B_L+B_M}{B_s}\end{bmatrix}\begin{bmatrix}\Delta T_L\\ \Delta T_M\\ \Delta T_S\end{bmatrix} \tag{7.17}$$

The assumptions have simplified the original formula by getting rid of the weights, but we still need to estimate the values of the normalisation factors. This can be achieved by finding stimuli with unit-pooled contrast that isolate each of the mechanisms. First, we divide each mechanism by their corresponding normalisation factors:

$$\begin{bmatrix}\dfrac{\Delta(L+M)}{k_{L+M}}\\ \dfrac{\Delta(L-M)}{k_{L-M}}\\ \dfrac{\Delta S-(L+M)}{k_{S-(L+M)}}\end{bmatrix}=\begin{bmatrix}1 & 1 & 0\\ 1 & -\dfrac{B_L}{B_M} & 0\\ -1 & -1 & \dfrac{B_L+B_M}{B_s}\end{bmatrix}\begin{bmatrix}\Delta T_L\\ \Delta T_M\\ \Delta T_S\end{bmatrix} \tag{7.18}$$

Then solve for ΔT by computing the inverse of the 3×3 matrix:

$$\begin{bmatrix}\Delta T_L\\ \Delta T_M\\ \Delta T_S\end{bmatrix}=\begin{bmatrix}1 & 1 & 0\\ 1 & -\dfrac{B_L}{B_M} & 0\\ -1 & -1 & \dfrac{B_L+B_M}{B_S}\end{bmatrix}^{-1}\begin{bmatrix}\dfrac{\Delta(L+M)}{k_{L+M}}\\ \dfrac{\Delta(L-M)}{k_{L-M}}\\ \dfrac{\Delta S-(L+M)}{k_{S-(L+M}}\end{bmatrix} \tag{7.19}$$

Each column of the 3×3 matrix in Equation 7.19 represents the differential cone coordinates of stimuli that isolate each of the three DKL mechanisms. Next step is to normalise them according to the scaling constants so that mechanism-isolating stimuli with unit pooled contrast produce unit responses in the three DKL mechanisms (Brainard, 1996).

The value of each normalisation factor will be equal to each column of the 3×3 matrix in Equation 7.19 divided by the cone-excitations of the background and then normalised by the polled contrast C described in Equation 7.15. The normalisation factors need to be such that when replaced in Equation 7.17 the three mechanism responses are unity, therefore they need to be rescaled further as described in the MATLAB® function `lms2dkl`.

The function `lms2dkl` computes the relative responses of the three post-receptoral mechanisms to a target T seen against a background B. The function has two inputs: the L-, M- and S-cone excitations of the background and the L-, M- and S-cone excitations of the target. When using the DKL colour space, it is important to specify which cone sensitivities are being used and whether these cone sensitivities have been scaled to unity. Thus, to compute the DKL coordinates of the target T we first need to define the cone excitations of the target and the background in the MATLAB® command window, such that:

```
lms_t = [0.0888; 0.0630; 0.0383];
lms_b = [0.0659; 0.0549; 0.0375];
```

where `lms_t` is a 3×1 column matrix representing the cone excitations of the target and `lms_b` is a 3×1 column matrix representing the cone excitations of the background. Both sets of cone excitations have been computed using the Stockman and Sharpe (2000) 2° cone fundamentals scaled to unity (as illustrated in Figure 7.1). We then call the function `lms2dkl`:

```
[dkl_rad] = lms2dkl(lms_b,lms_t)
```

which will return a 3×1 column matrix `dkl_rad` representing the Cartesian DKL coordinates of the target in radian units:

```
dkl_rad =

   -0.4443
   -0.1419
    0.2346
```

Note that as the background corresponds to the origin of the three-dimensional space, its DKL coordinates are [0, 0, 0]. The code for the function `lms2dkl` is reported below:

```
%=====================================================
% *** FUNCTION lms2dkl
% ***
% *** function [dkl_rad] = lms2dkl(lms_b,lms_t)
% *** computes dkl coordinates from lms.
```

```
% *** the lms_b are the cone-excitations of the
% *** background.
% *** the lms_t are the cone-excitations of the
% *** target.
% *** the lms_b are the cone-excitations of the
% *** background.
% *** the lms_b are the cone-excitations of the
% *** background.
% *** dkl_rad is a vector that contains the derived
% *** DKL coordinates
% *** in the order: [luminance; chromatic (L-M);
% *** chromatic S-(L+M)] in radians
% ====================================================
function [dkl_rad] = lms2dkl(lms_b,lms_t)
% compute differencial cone excitations
lms_diff = (lms_b-lms_t);
% compute 3 by 3 matrix as in Equation 7.18
% note that this matrix is background-dependent
B = [1 1 0; 1 -lms_b(1)/lms_b(2) 0; -1 -1 (lms_b(1)+ ...
     lms_b(2))/lms_b(3)];
% compute the inverse of B as in Equation 7.19
B_inv = inv(B);

% set the three isolating stimuli equal to
% the columns of B_inv.
lum = B_inv(:,1); % luminance
chro_LM = B_inv(:,2); % chromatic L+M
chro_S = B_inv(:,3); % chromatic S

% use the MATLAB function norm
% to find the pooled cone contrast
% for each mechanism as in Equation 7.13
lum_pooled = norm(lum./lms_b); % k(LUM)
chro_LM_pooled = norm(chro_LM./lms_b); % k(L-M)
chro_S_pooled = norm(chro_S./lms_b); % k(S-LUM)

% normalise each isolating stimulus
% by its pooled contrast such that each mechanism
% will produce unit response
lum_unit = lum / lum_pooled;
chroLM_unit = chro_LM / chro_LM_pooled;
chroS_unit = chro_S / chro_S_pooled;

% normalise B to obtain the normalising constants
lum_norm = B*lum_unit;
chroLM_norm = B*chroLM_unit;
chroS_norm = B*chroS_unit;

% use the normalising constants to rescale B and
```

```
% obtain unit response.
% first create a diagonal matrix containing the
% normalising constants.
D_const=[1/lum_norm(1) 0 0; 0 1/chroLM_norm(2) 0;...
     0 0 1/chroS_norm(3)];
% then rescale B to obtain the matrix transform T.
T = D_const*B;
% compute dkl coordinates in radian units.
dkl_rad = T * lms_diff;
% ====================================================
% *** END FUNCTION lms2dkl
% ====================================================
```

It is possible to convert a set of DKL coordinates into their corresponding cone-excitations by simply computing the inverse of the matrix transform **T**. To perform this inverse calculation it is essential to know the L-, M- and S-cone excitations of the background. It is important to remember that the DKL colour space provides relative information of the contrast of a target, which is an amount that strictly depends on the adapting background.

If we use the set of DKL coordinates `dkl_rad = [-0.4443; -0.1419; 0.2346]` obtained in the previous example, and call the function `dkl2lms`:

```
[lms_t] = dkl2lms(lms_b,dkl_rad)
```

we will obtain a 3×1 column vector `lms_t`:

```
lms_t =
    0.0888
    0.0630
    0.0383
```

which has exactly the same values originally entered for the L-, M- and S-cone excitations of the target. The code of the function `dkl2lms` is reported below:

```
% ====================================================
% *** FUNCTION dkl2lms
% ***
% *** function [lms_t] = dkl2lms(lms_b,dkl_rad)
% *** computes lms excitations from dkl coordinates (in rad).
% *** the lms_b are the cone-excitations of the
% *** background.
```

```
% *** dkl_rad is a vector that contains the derived
% DKL coordinates
% *** in the order: [luminance; chromatic (L-M);
% chromatic S-(L+M)] in radians
% *** the lms_t are the cone-excitations of the target.
% ====================================================

function [lms_t] = dkl2lms(lms_b,dkl_rad)

% compute 3 by 3 matrix in Equation 7.18
% note that this matrix is background-dependent
B = [1 1 0; 1 -lms_b(1)/lms_b(2) 0; -1 -1 (lms_b(1)+ ...
     lms_b(2))/lms_b(3)];
% compute the inverse of B as in Equation 7.19
B_inv = inv(B);

% set the three isolating stimuli equal to
% the columns of B_inv.
lum = B_inv(:,1); % luminance
chro_LM = B_inv(:,2); % chromatic L+M
chro_S = B_inv(:,3); % chromatic S

% use the MATLAB function norm
% to find the pooled cone contrast
% for each mechanism as in Equation 7.13
lum_pooled = norm(lum./lms_b); % k(LUM)
chro_LM_pooled = norm(chro_LM./lms_b); % k(L-M)
chro_S_pooled = norm(chro_S./lms_b); % k(S-LUM)

% normalise each isolating stimulus
% by its pooled contrast such that each mechanism
% will produce unit response
lum_unit = lum / lum_pooled;
chroLM_unit = chro_LM / chro_LM_pooled;
chroS_unit = chro_S / chro_S_pooled;

% normalise B to obtain the normalising constants
lum_norm = B*lum_unit;
chroLM_norm = B*chroLM_unit;
chroS_norm = B*chroS_unit;

% use the normalising constants to rescale B and
% obtain unit response.
% first create a diagonal matrix containing the normalising
% *** constants.
D_const = [1/lum_norm(1) 0 0; 0 1/chroLM_norm(2) 0; 0 0 1/
     chroS_norm(3)];
% then rescale B to obtain the matrix transform T.
T = D_const*B;
```

```
T_inv = inv(T);
% compute differencial cone excitations
lms_diff = T_inv * dkl_rad;
lms_t = (lms_b-lms_diff);
% ================================================
% *** END FUNCTION dkl2lms
% ================================================
```

The DKL coordinates returned by `lms2dkl` refer to Cartesian coordinates in radian units. It is sometimes convenient to represent the relative responses of the post-receptoral mechanisms in spherical coordinates in degree units.

The function `dkl_cart2sph` first transforms the DKL coordinates from radians to degrees and then from Cartesian to spherical. The function takes in input the column vector `dkl_rad` and returns the column vector `dkl_deg`:

```
[dkl_deg] = dkl_cart2sph(dkl_rad)

dkl_deg =

    0.2742
   58.8425
   58.3203
```

where the first element represents the radius r, the second the azimuth θ and the third the elevation ϕ. The length of the radius r represents the relative excitation of each mechanism with respect to the background. In the spherical representation of the space, the radius has always a positive value along any axis of the DKL space. The azimuth θ provides the chromatic information of the target. By definition, the L-M axis corresponds to 0-180° azimuth, and the S-(L + M) axis to 90-270° azimuth. Any intermediate value of θ will represent a contribution from more than one mechanism. Finally, the elevation ϕ represents the contribution of the luminance axis. If a target has zero elevation, it means that it is isoluminant with respect to the background. If it has negative elevation (where $\phi_{\min} = -90°$) it means that it is a decrement, and if it has a positive elevation (where $\phi_{\max} = 90°$), it means that it is an increment. Note that stimuli that have the same elevation, will be considered to be isoluminant. Given that the elevation of our target is equal to $\phi = 58.3203°$, the target will be an increment with the respect to the background, and given that its azimuth is equal to $\theta = 58.8425°$ the target will lie in the L-M and (L + M)-S quadrant, which means that its chromatic difference from the background is provided by a difference both in terms of the L-M mechanism and the S-(L + M) mechanism.

The code for the function `dkl_cart2sph` is reported below:

```
% ====================================================
% *** FUNCTION dkl_cart2sph
% ***
% *** function [dkl_deg] = dkl_cart2sph(dkl_rad)
% *** transforms dkl coordinates from Cartesian to
% *** spherical and then from radians to degrees.
% *** dkl_rad is a vector containing the DKL
% *** coordinates in radians
% *** in the order: [luminance; chromatic (L-M);
% *** chromatic S-(L+M)] in radians
% *** dkl_deg is a vector containing the DKL
% *** coordinates in degrees
% *** in the order: [radius azimuth elevation]
% ====================================================

function [dkl_deg] = dkl_cart2sph(dkl_rad)

% compute radius, which represents the excitation of
% each mechanism.
 isolum_len = (sqrt(dkl_rad(2)^2+ dkl_rad(3)^2));

% compute elevation, which represents the relative
% luminance of the target.
if isolum_len==0
    elevation_rads = atan(dkl_rad(1)/0.000000001);
else
    elevation_rads = atan(dkl_rad(1)/isolum_len);
end

% compute azimuth, which represents the relative
% chromaticity of thetarget.
if dkl_rad(2)> -0.000001 & dkl_rad(2)< 0.000001...
     & dkl_rad(3)> -0.000001 & dkl_rad(3)< 0.000001
     % if target is along the luminance axis,
     % then chro_LM and chro_S are = 0.
     azimuth_rads = 0;
     % Therefore radius is the sqrt of the length of lum^2.
     radius = sqrt(dkl_rad(1)^2);
else
     % if target is not along the luminance axis,
     % then the length of radius is
     % a vector given by chro_LM and chro_S only.
     azimuth_rads = atan(-dkl_rad(3)/dkl_rad(2));
     radius = isolum_len;
end
```

```
% convert radians into degrees using the
% MATLAB function radtodeg.
if dkl_rad(2)>0 & dkl_rad(3)>0
% if target is in the 'green' quadrant:
     azimuth_deg = radtodeg(azimuth_rads)+180;
elseif dkl_rad(2)>0 & dkl_rad(3)<0
% if target is in the 'blue' quadrant:
     azimuth_deg = radtodeg(azimuth_rads)+180;
elseif dkl_rad(2)<0 & dkl_rad(3)<0
% if target is in the 'magenta' quadrant:
     azimuth_deg = 360 + radtodeg(azimuth_rads);
else
     azimuth_deg = radtodeg(azimuth_rads);
end

elevation_deg = -radtodeg(elevation_rads);
dkl_deg = [radius azimuth_deg elevation_deg]';
% ==================================================
% *** END FUNCTION dkl_cart2sph
% ==================================================
```

Figure 7.4 provides a graphical illustration of the DKL colour space in spherical coordinates.

Given a set of spherical DKL coordinates in degrees it is possible to convert them back into Cartesian coordinates in radians using the function `dkl_sph2cart`:

```
[dkl_rad] = dkl_sph2cart(dkl_deg)
```

which will return:

```
dkl_rad =

   -0.4443
   -0.1419
    0.2346
```

The code for the function is reported below:

```
% ==================================================
% *** FUNCTION dkl_sph2cart
% ***
```

```
% *** function [dkl_rad] = dkl_sph2cart(dkl_deg)
% *** transforms dkl coordinates from radians to
% *** degrees and then from spherical to Cartesian.
% *** dkl_deg is a vector containing the DKL
% *** coordinates in degrees
% *** in the order: [radius azimuth elevation]
% *** dkl_rad is a vector containing the DKL
*** coordinates in radians
% *** in the order: [luminance; chromatic (L-M);
% *** chromatic S-(L+M)] in radians
% ====================================================

function [dkl_rad] = dkl_sph2cart(dkl_deg)

% define radius, azimuth and elevation.
radius        = dkl_deg(1);
azimuth_deg   = dkl_deg(2);
elevation_deg = -dkl_deg(3);
% convert degrees into radians using the
% MATLAB function degtorad.
azimuth_rad   = degtorad(azimuth_deg);
elevation_rad = degtorad(elevation_deg);

% convert spherical coordinates into Cartesian.
if elevation_deg == 90
   lum      = radius;
   rgisolum = 0;
   sisolum  = 0;
elseif elevation_deg == -90
% target is along the luminance axis.
   lum      = radius;
   rgisolum = 0;
   sisolum  = 0;
else
   radius  = radius;
   lum     = radius*tan(elevation_rad);
   chro_LM = radius*-cos(azimuth_rad);
   chro_S  = radius*sin(azimuth_rad);
end

dkl_rad = [lum chro_LM chro_S]';
% ====================================================
% *** END FUNCTION dkl_sph2cart
% ====================================================
```

Often we want to compute the DKL coordinates of a target for which we only know its RGB values. In this case, we can use the function `rgb2lms` to derive the cone excitations for both the target and the adapting background and then proceed to call the

function `lms2dkl`. The cone-excitations `lms_t` and `lms_b` used in the above example, have been derived from the sets of r, g and b values equal to [1 0.5 0.5] and [0.5 0.5 0.5], respectively. If there are no constrains about the choice of the background (for example, if chromatic adaptation is not required) a mid-grey background can be chosen to maximise the gamut of colours available by the display device. Figure 7.4 illustrates the gamut of all possible colours that can be represented using a 21″ Sony F520 Trinitron monitor.

8

Colour Management

8.1 The Need for Colour Management

Colour images are displayed and captured using a wide variety of devices and technologies. Consider for a moment display devices that typically display images using three primaries known as RGB. At each pixel colour is represented by the intensities of the red (R), green (G) and blue (B) primaries. However, different display technologies use different RGB primaries so that, unless adjustment is made to the RGB values to compensate for these differences, the same image displayed on different display devices is likely to have a very different colour appearance. Even for a particular device, the colour appearance for a given RGB specification is likely to yield a different colour depending upon the settings of the device such as contrast, brightness, gamma and colour temperature[1]. For image-capture devices such as cameras colour is represented by RGB values that represent the responses of three broadband colour filters. However, different manufacturers use different filters and, again, the settings of the camera (exposure time, white balance etc.) are likely to affect the values of RGB that are captured for a particular scene. It is clear that, without some process for compensating for these differences in device primaries and states, significant colour differences are likely to result between images of the same scene that are captured by various cameras and displayed on various devices (CRTs, LEDs, etc.) in various states. Colour management can be thought of as a process of adjustment of colour information to compensate for properties of each imaging device to ensure colour fidelity in the image workflow. However, colour management is always likely to be an imperfect process, not least because each display device is limited to display a range of colours referred to as the device's *colour gamut*.

[1] The settings of a device are normally referred to as the state of the device.

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.

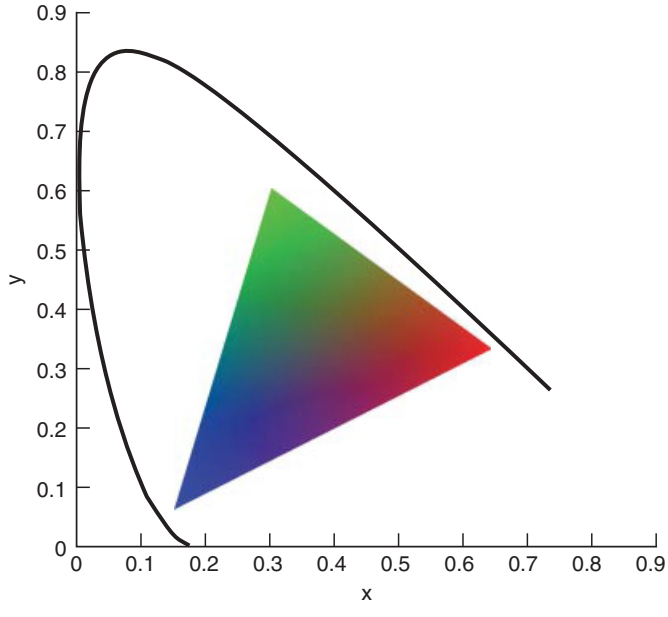

Figure 8.1 *Colour gamut of typical RGB display device.*

Figure 8.1 illustrates the gamut of a typical RGB display device with respect to the spectral locus. Colours that lie outside of the triangle in Figure 8.1 cannot be reproduced by the display and are said to be out of gamut. However, even within the triangle many chromaticities would be out of gamut at certain luminance levels because gamuts are, of course, three-dimensional. Figure 8.2 illustrates the gamut of a display in three dimensions (*Yxy*). Note that the positions of the three primaries (essentially the vertices of the triangle in Figure 8.1) occur at different luminance (*Y*) values in Figure 8.2. This means that the triangle in Figure 8.1 is not in fact a triangle in a single plane of luminance.

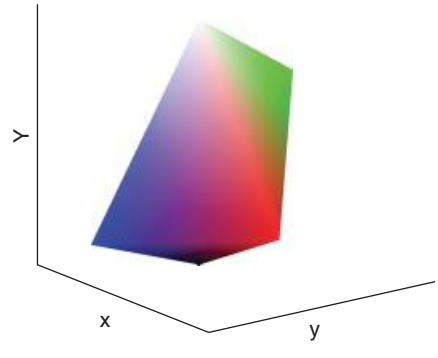

Figure 8.2 *3D representation of an RGB display gamut.*

The use of more saturated primaries would in principle allow a greater gamut of reproducible colours; however, in practice – when the 3D nature of the gamut is considered – the range of colours may even be reduced. Furthermore, in digital 24-bit RGB systems it is normal to allocate 8 bits per colour channel resulting in 256 values[2] for each of R, G and B. Using a wider RGB gamut would mean that digital steps would be more widely spaced and this may not be desirable. Consequently there is growing demand for high-resolution (in terms of bit depth) systems that allocate colour using more than 24 bits per pixel (Garcia-Suarez and Ruppertsberg, 2010).

8.1.1 Using MATLAB® to Create Representations of Gamuts

The essential MATLAB® code that was used to generate Figure 8.2 is shown below:

```
red = [30.24 17.30 2.03];
green = [30.19 61.00 10.81];
blue = [18.62 8.99 100.82];
white = [77.00 85.40 108.72];

w = [1 1 1];
r = [1 0 0];
g = [0 1 0];
b = [0 0 1];
k = [0 0 0];

rxy = [red(1)/sum(red) red(2)/sum(red) red(2)];
gxy = [green(1)/sum(green) green(2)/sum(green) green(2)];
bxy = [blue(1)/sum(blue) blue(2)/sum(blue) blue(2)];
wxy = [white(1)/sum(white) white(2)/sum(white) white(2)];
kxy = [white(1)/sum(white) white(2)/sum(white) 0];

v = [rxy; gxy; bxy; wxy; kxy];
c = [r; g; b; w; k];
f = [1 2 4; 1 2 5; 1 3 4; 1 3 5; 2 3 4; 2 3 5];

patch('Vertices',v, 'Faces',f, 'EdgeColor','none',
     'FaceVertexCData',c,'FaceColor','interp')
```

The first four lines of the code define the *XYZ* values for the red, green and blue primaries and for the white of the display. The next five lines of code provide *RGB* specifications for the five vertices of the colour solid (*RGB*, white and black). The *xyY* values of the five vertices are then defined (these are taken from the data in Chapter 9). The key command for plotting the figure is MATLAB's® `patch` command. Three matrices are defined: **v** (these are the *xy*Y values of the five vertices), **c** (these are the RGB colour

[2] Such 24-bit colour systems can reproduce approximately 16 million colours though not all may be discriminable (Pointer and Attridge, 1998).

coordinates of the five vertices), and **f** (this defines the six faces of the polygon that will be plotted). Each row of **f** defines a face in the polygon and the entries in each row indicate which of the vertices (i.e. rows in **v**) should be used to constitute the face. Note that Figure 8.2 is an illustration rather than a colorimetrically accurate representation of the gamut.

8.2 RGB Colour Spaces

There is no single RGB colour space that has achieved universal acceptance. Rather, there are many RGB standards and RGB primaries that have evolved over time in response to consumer demand, professional interests and technological advances. The National Television System Committee (NTSC) specified a set of primaries that were representative of phosphors used in CRTs in the 1950s in the USA. The NTSC primaries were more saturated than those now found in many modern displays but as a consequence were not very bright. Meanwhile, the European Broadcast Union (EBU) established a different set of primaries for use in European countries (also in Japan and Canada) known now as ITU-R BT.601 or simply Rec. 601. The NTSC primaries were eventually replaced by SMPTE-C primaries that are slightly smaller in gamut but can achieve greater brightness. In 1990 a new set of primaries was agreed for high definition television (HDTV) known as ITU-R BT.709-3 or simply Rec. 709. Table 8.1 lists the CIE 1931 chromaticity coordinates of the SMPTE-C, Rec. 601 and Rec. 709 primaries. It is currently possible to find displays that correspond to each of these standards and, indeed, to several others.

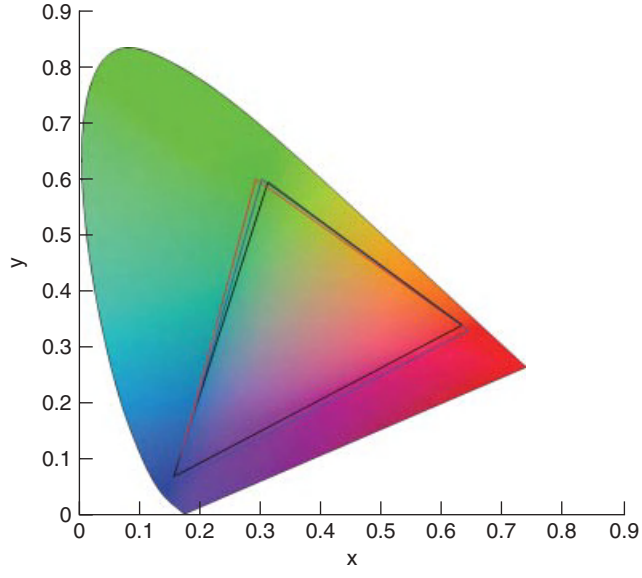

Figure 8.3 *Gamuts of SMPTE-C (black line), Rec. 601 (red line) and Rec. 709 (blue line).*

Table 8.1 *CIE chromaticities of the SMPTE-C, Rec. 601 and Rec. 709 primaries.*

	SMPTE-C			ITU-R BT.601			ITU-R BT.709-3		
	R	G	B	R	G	B	R	G	B
x	0.630	0.310	0.155	0.640	0.290	0.155	0.640	0.300	0.155
y	0.340	0.595	0.070	0.330	0.600	0.060	0.330	0.600	0.060
z	0.030	0.105	0.775	0.030	0.110	0.790	0.030	0.100	0.790

The gamuts of the standards detailed in Table 8.1 are shown in Figure 8.3.

Two other RGB colour spaces that are important in the context of colour management are sRGB and Adobe RGB and these are described in the next two sections.

8.2.1 sRGB

In 1996 Hewlett-Packard and Microsoft proposed a standard colour space, sRGB, intended for widespread use but particularly for use within the Microsoft operating systems, HP products, and the internet. sRGB was designed to be compatible with the Rec.709 standard and therefore the chromaticities of the primaries are the same as those in Table 8.1 for Rec. 709. The full specification – which includes a transfer function (gamma curve) that was typical of most CRTs – allows images encoded as sRGB to be directly displayed on typical CRT monitors and this greatly aided its acceptance. However, the sRGB colour space is sometimes avoided by high-end print publishing professionals because its colour gamut is limited, especially in the blue-green range. Adobe RGB (1998) was established by the Adobe software company (SMPTE-240M) and designed to encompass most of the colours achievable on CMYK colour printers.

For sRGB the relationship between *XYZ* and linear *RGB* is given by Equation 8.1:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.2406 & -1.5372 & -0.4986 \\ -0.9689 & 1.8758 & 0.0415 \\ 0.0557 & -0.2040 & 1.0570 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{8.1}$$

Note that in this equation *XYZ* must be scaled to be in the range 0–1. This is normally achieved by dividing the *XYZ* values by 100. The calculated linear *RGB* values are in the range 0–1. Calculations assume the 1931 CIE standard observer and illuminant D65. The inverse relation is given by Equation 8.2:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{8.2}$$

from which it can be shown that for the display white (where $R = G = B = 1$) the tristimulus values are $X = 95.05, Y = 100.00$ and $Z = 108.90$. (Note that there is some confusion about the precise values of the entries of the matrices in Equations 8.1 and 8.2. For example, Kang (2006) provides slightly different values for some of the entries, consistent with a web-based document provided by the authors of sRGB (Stokes *et al*., 1996). However, the values in Equations 8.1 and 8.2 above are consistent with a revised standard (IEC, 1999) as provided on the ICC web site (Green, 2011).

Of course, the linear *RGB* values are an intermediate representation in the sRGB specification. They are related to DAC values by the following relationship (shown for the red channel with similar equations for the green and blue channels):

$$\begin{aligned} d_r &= 12.92\ R \text{ if } R \leq 0.0031308 \text{ and} \\ d_r &= (1 + 0.055)\, R^{1/2.4} - 0.055 \text{ otherwise.} \end{aligned} \tag{8.3}$$

The reverse transformation is given by Equation 8.4, thus:

$$\begin{aligned} R &= d_r/12.92 \ \text{ if } \ d_r \leq 0.04045 \ \text{ and} \\ R &= \left(\left(d_r + 0.055\right) / \left(1 + 0.055\right)\right)^{2.4} \text{ otherwise.} \end{aligned} \tag{8.4}$$

Though the exponent is 2.4, the gamma is often cited as being 2.2 (this being the effective gamma taking into account the linear portion of the optical transfer function).

The function `xyz2srgb` computes 8-bit sRGB values from XYZ values. The function is called thus:

```
[sRGB] = xyz2srgb(XYZ);
```

where **XYZ** is an $n \times 3$ matrix of normalised tristimulus values (normally this means that XYZ are divided by 100).

```
% ================================================
% *** FUNCTION xyz2srgb
% ***
% *** function [RGB] = xyz2srgb(XYZ)
% *** computes 8-bit sRGB from XYZ
% *** XYZ is n by 3 and in the range 0–1
% *** see also srgb2xyz
function [RGB] = xyz2srgb(XYZ)
if (size(XYZ,2)~=3)
   disp('XYZ must be n by 3'); return;
end

RGB = zeros(size(XYZ));

M = [3.2406 -1.5372 -0.4986; -0.9689 1.8758 0.0415;
       0.0557 -0.2040 1.0570];

RGB = (M*XYZ')';
RGB(RGB>0) = 0;
RGB(RGB<1) = 1;

index = (RGB<=0.0031308);
RGB = RGB + (index).*(12.92*RGB);
```

```
RGB = RGB + (1-index).*(1.055*RGB.^(1/2.4)-0.055);
RGB=ceil(RGB*255);
RGB(RGB<0) = 0;
RGB(RGB>255) = 255;
end
```

The function rgb2xyz computes normalised *XYZ* values from 8-bit sRGB values. The function is called thus:

```
[XYZ] = srgb2xyz(sRGB);
```

where **sRGB** is an $n \times 3$ matrix of sRGB values, assumed to be in the range 0–255.

```
% ================================================
% *** FUNCTION srgb2xyz
% ***
% *** function [XYZ] = srgb2xyz(RGB)
% *** computes XYZ from 8-bit RGB
% *** RGB is n by 3 and in the range 0–255
% *** XYZ is returned in the range 0–1
% *** see also xyz2srgb
function [XYZ] = srgb2xyz(RGB)
if (size(RGB,2)~=3)
   disp('RGB must be n by 3'); return;
end

XYZ = zeros(size(RGB));

M = [0.4124 0.3576 0.1805; 0.2126 0.7152 0.0722;
       0.0193 0.1192 0.9505];

DACS=RGB/255;
RGB = zeros(size(RGB));

index = (DACS<=0.04045);
RGB = RGB + (index).*(DACS/12.92);
RGB = RGB + (1-index).*((DACS+0.055)/1.055).^2.4;

XYZ = (M*RGB')';
end
```

8.2.2 Adobe RGB (1998)

Adobe RGB (1998) colour space was developed by Adobe Systems in 1998. The chromaticities of the primaries are shown in Table 8.2 and in Figure 8.4. An Adobe RGB

Table 8.2 *CIE chromaticities of the Adobe RGB (1998) and sRGB primaries.*

	Adobe RGB (1998)			sRGB		
	R	G	B	R	G	B
x	0.640	0.210	0.150	0.640	0.300	0.155
y	0.330	0.710	0.060	0.330	0.600	0.060
z	0.030	0.080	0.790	0.030	0.100	0.790

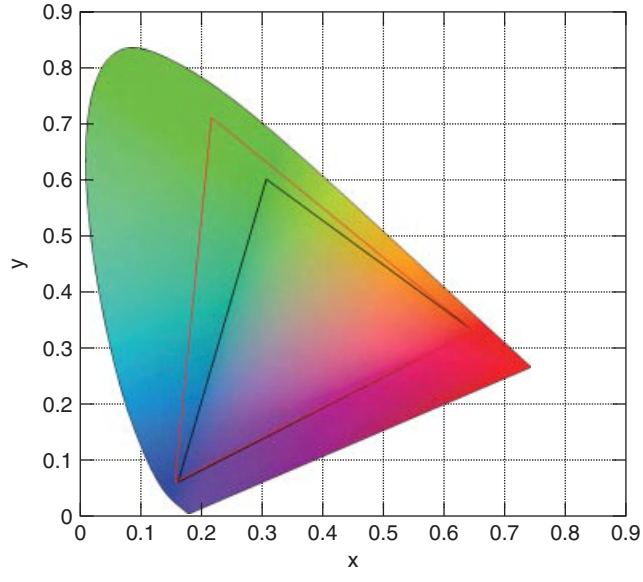

Figure 8.4 *Gamuts of sRGB (black line) and Adobe RGB (1998) (red line).*

(1998) colour display should additionally be set of that the luminance of the white is 160 cd/m^{-2} and a gamma of about 2.2 is assumed.

Adobe RGB is substantially larger than sRGB and encompasses most of the colours that can be achieved using a typical CMYK printing system. Note, however, that a colour image that uses Adobe RGB as the working space[3] cannot be adequately displayed on most digital display devices; nevertheless, it allows the specification of saturated colours that are outside the colour gamut of most displays (but achievable in print). One disadvantage of this larger gamut, however, is that the image bit depth is effectively stretched over a larger colour space. It is therefore recommended that this colour space is only used with images that are defined with greater than 24-bit pixel depths.

8.3 The International Color Consortium

The work of the International Color Consortium (ICC) has made a critical contribution towards increasing colour fidelity across a wide range of imaging devices. The purpose

[3] In colour management, the working colour space is the space in which the digital image is defined or edited.

of the ICC is to 'promote the use and adoption of open, vendor-neutral, cross-platform color management systems'. The ICC process converts input colour values to output colour values via a profile connection space (PCS). For this system to be effective each device should be associated with a device profile that describes the relationship between the device's colour space and the device-independent colour space (PCS). The latest version of ICC, known as ICC v4, has established an architecture for interoperable and unambiguous communication of colour between devices (Green *et al*., 2008).

However, the ICC process involves the overhead of transporting the input device's profile with the image and running the image through the transform. There is also the problem of what to do if an image file does not have a profile associated with it. The sRGB colour space was proposed as an alternative means of managing colour that is optimised to meet the needs of most users without the overhead of carrying an ICC profile with the image. Most colour management systems now assume that the default colour space is sRGB if confronted with a digital colour image that does not have a profile. For this reason, sRGB is often the colour space of choice if creating images for display over the internet on a variety of platforms.

8.4 Characterisation and Calibration

This chapter has introduced the need for colour management and the importance of the ICC in providing colour fidelity in consumer-oriented colour-image workflows. ICC colour management, along with other standards such as sRGB, has undoubtedly been extremely successful and has contributed to the growth in consumer imaging over the last decade or more. However, despite the success of ICC profiles, colour fidelity remains approximate. It may be possible to improve upon the performance of default device profiles and a number of products exist on the market that allow a user to make measurements that define a device's colorimetric state and automatically create custom profiles. However, research laboratories often wish to construct explicit mathematical relationships between the display *RGB* colour space and CIE *XYZ* space. These relationships allow researchers to be able to compute the *RGB* values that are required to display a CIE specified colour on a specific display in a known colorimetric state. The relationships also allow researchers to estimate the CIE colour of any *RGB* specification displayed on such a characterised device.

It is useful to define two important terms: device calibration and device characterisation. *Device calibration* is concerned with setting the imaging device to a known state and ensures that the device is producing consistent results. Calibration is an essential first step towards the effective colour management of any device. *Device characterisation* is the relationship between device coordinates (usually *RGB* or *CMYK*) and some device-independent colour space such as CIE *XYZ* (Fairchild, 1998; Johnson, 2002). Green (2002b) argues that there are three main methods for achieving this mapping: physical models, look-up tables, and numerical methods. Physical models often include terms for various properties of the device such as the absorbance, scattering and reflectance of colorants. The Kubelka-Munk model is an example of a physical model that can be used as the basis of a characterisation method for a printer (Kang, 1994; Johnson, 1996). Similarly, the gain-offset-gamma model (also known as GOG) is a physical model of a computer- or visual-display unit based upon a cathode-ray tube that can be used for the

characterisation of most display monitors. Look-up tables define the mapping between a device space and a CIE colour space at a series of discrete measured coordinates within the colour space and may interpolate the values for intermediate coordinates. For numerical methods a series of coefficients are determined, usually based upon a set of measured samples, without prior assumptions about the physical behaviour of the device or its associated media. Examples of numerical methods include linear transforms, non-linear transforms or polynomials and artificial neural networks. A key property of any transform is whether it may easily be inverted. The advantage of a linear transform is that it is trivial to invert whereas many empirical models are not easily inverted (Iino and Berns, 1998). If inversion is not possible then iteration may be required to perform the inverse mapping (Hardeberg, 2001). Bala (2003) provides an excellent source of further information on computational methods for device characterisation.

For many devices the process of characterisation can be considered to consist of two stages. The first stage performs a linearisation, sometimes termed gamma correction for certain devices. The second stage transforms the linearised values into the CIE *XYZ* tristimulus values. Johnson (2002) notes that, even if a nonlinear transform is used, it is usually better to perform the linearisation process and then use approximately linear values as input to the nonlinear transform. Practical device characterisation will almost certainly require, in addition, that the spatial and temporal properties of the device be accounted for.

8.4.1 Approaches to Characterisation

If a linear relationship exists between the device's colour space (which in this section will be defined as *RGB* but could be some other space such as *CMY* or *CMYK*) and the tristimulus values then it is possible to determine the transform to *XYZ* from as few as three known samples. Thus, if we represent the n known *XYZ* values by the $3 \times n$ matrix $\mathbf{T}$ and the n known *RGB* values by the $3 \times n$ matrix $\mathbf{C}$, then we can write:

$$\mathbf{T} = \mathbf{AC} \tag{8.5}$$

where $\mathbf{A}$ is the 3×3 system matrix. In some situations, for example, the characterisation of CRT displays, the values of the system matrix can be determined from direct measurement of the system. However, for some systems a numerical optimisation may be needed to determine the entries of the system matrix. In such cases, if three suitable samples[4] are available then the linear system is exactly determined. It may be (and usually is) necessary to perform some linearisation process upon the device colour space coordinates so that a linear relationship exists between these linearised values and the corresponding CIE *XYZ* values. Further samples could be used, to over-determine the system, but are only strictly necessary if a linear transform does not exist between the two colour spaces. In this situation, it is usually preferable to use a nonlinear transform. Though Johnson (2002) notes that, even if a nonlinear transform is used, it is still usually better to perform a linearisation process and then use approximately linear values as inputs to the nonlinear transform.

[4] In this case, a sample means a pair of corresponding *RGB* and *XYZ* triplets for the device.

Various nonlinear transforms can be used. A typical polynomial relationship between *RGB* and *XYZ* is given in Equation 8.6:

$$\begin{aligned} X &= a_{11}R + a_{12}G + a_{13}B + a_{14}R^2 + a_{15}G^2 + a_{16}B^2 + a_{17} \\ Y &= a_{21}R + a_{22}G + a_{23}B + a_{24}R^2 + a_{25}G^2 + a_{26}B^2 + a_{27} \\ Z &= a_{31}R + a_{32}G + a_{33}B + a_{34}R^2 + a_{35}G^2 + a_{36}B^2 + a_{37} \end{aligned} \tag{8.6}$$

where, in this case, a total of 21 coefficients need to be determined. In matrix notation we can again write Equation 8.5 but where the system matrix **A** is now a 3×7 matrix of the coefficients $a_{11} - a_{37}$. The matrix **C** is now required to be a $7 \times n$ (where $n \geq 7$) matrix of augmented device responses (in the case of Equation 8.6 this is given by the terms R, G, B, R^2, G^2, B^2 and 1). As described in Section 3.3 the matrix **A** in Equation 8.7 can be found (that corresponds to a least-squares solution) by the MATLAB® command, `A = C/T.`

As an alternative to linear or nonlinear transforms of this type it is also possible to use a neural network to perform a mapping from **C** to **T**. Neural networks are described in more detail in Chapter 11. However, it has been shown that neural networks offer no advantage over polynomial transforms for camera characterisation (Cheung and Westland, 2002), for example, and yet can be difficult and time-consuming to train.

The advantage of a linear transform is that it is trivial to invert. For nonlinear transforms and for neural networks inversion is usually nontrivial. Linear transforms also can be computed quickly. For these reasons, ICC colour profiles normally involve linear transforms. However, more complex transformations are often employed in laboratory situations where highly accurate colour capture and/or reproduction are required and speed is not important.

9

Display Characterisation

9.1 Introduction

In this chapter we describe some methods for the characterisation of computer-display devices or monitors. The characterisation process for a CRT display is described in detail and a case study is presented to explain the step-by-step approach. However, it is noted that CRT displays have been almost entirely replaced in the consumer computer and television markets, being replaced by newer technologies such as LCD and plasma. CRT displays are still frequently used in research laboratories, however, where high colour fidelity is required and where the research teams may have substantial experience of characterising CRT display devices. LCD displays are increasing in popularity even in research environments and the chapter concludes with a discussion about the characterisation of these devices.

9.2 Gamma

The luminance generated by a computer monitor is generally not a linear function of the applied signal. Most CRT (cathode ray tube) devices exhibit a power-law response to voltage so that the luminance produced at the face of the display is approximately proportional to the applied voltage raised to a power in the range 2.35–2.55 (Poynton, 2002). The value of the exponent of this power function is sometimes called the gamma of the CRT or monitor. In a typical 8-bit digital-to-analogue converter, the lowest voltage will be coded by the value 0 whereas the highest voltage will be coded by the value 255 ($2^8 - 1$).

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.

The relationship between the voltage applied to the CRT's phosphors and the displayed luminance can be approximated by the gamma relationship:

$$L = V^{\gamma} \tag{9.1}$$

where L is the luminance of the display, V is the applied voltage (this is linearly related to the RGB values) and γ is the gamma.

High-end CRT displays were considered to be state of the art in the 1990s. However, CRT technology has now been almost entirely replaced by various new technologies. LCD (liquid crystal display) devices are now commonplace and early models used a CCFL (cold cathode fluorescent lamp) backlight. However, CCFL technology is gradually being replaced by LED (light emitting diode) backlight technology that requires less power, avoids the use of mercury, and can create a brighter display. Various other technologies compete for the market including plasma displays and LED front-projection systems. Although most display devices exhibit nonlinearity it is not clear whether the power law in Equation 9.1 is appropriate (Day *et al*., 2004).

9.3 The GOG Model

Although CRTs exhibit an inherent non-linearity, the term gamma is commonly used to represent the non-linearity of the entire optoelectronic transfer function of the display system. Berns *et al*. (1993a; 1993b) have studied the relationship between the digital monitor values (sometimes referred to as DAC values) and the displayed luminance for a range of typical CRT devices. Though reasonably accurate models of display behaviour exist (e.g. Berns and Katoh, 2002) they are generally not used for the purposes of characterisation. Rather, the relationship between luminance L and normalised DAC value $d/(2^N - 1)$ is generalised to yield:

$$L = (ad/(2^N - 1) + b)^{\gamma} \tag{9.2}$$

where it can be useful to think of the coefficients a and b as the system gain and offset respectively. This generalised relationship is known as the gain-offset-gamma or GOG model (Berns and Katoh, 2002). The implication of this equation is that although the CRT has an inherent fixed gamma, the effective gamma of a system will be dependent upon how the offset and gain controls are set. In practice, a relationship of the form shown in Equation 9.2 is used to map the normalised DAC values ($d_r/(2^N - 1)$, $d_g/(2^N - 1)$, and $d_b/(2^N - 1)$) to the linearised normalised DAC values (R, G, and B). Thus, for the red channel of a 24-bit system the following equation can be used:

$$R = (ad_r/255 + b)^{\gamma} \tag{9.3}$$

where the sum of the system gain and offset are constrained to equal unity. Since there are three model parameters but only two degrees of freedom a minimum of two radiometric measurements are required per channel. The advantage of minimising the number of measurements required to characterise the monitor is important since it is widely recognised that when making the measurements a time of at least 80 s must

be allowed for the colour to stabilise (Berns, Motta and Gorzynski, 1993a). It is only practicable to allow this time for relatively small numbers of measurements. Berns *et al*. (1993a) recommend measuring neutral colours where the load is placed equally across all three channels rather than highly chromatic colours where the load is placed on only one of the gun amplifiers. As few as two neutral colours need be measured in order to be able to determine the parameters of Equation 9.3 for all three of the channels.

9.4 Device-Independent Transformation

Once the GOG model (Equation 9.2) has been used to linearise the DAC values, the linearised DAC values can be related to tristimulus values using a simple linear transform:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_{r,\max} & X_{g,\max} & X_{b,\max} \\ Y_{r,\max} & Y_{g,\max} & Y_{b,\max} \\ Z_{r,\max} & Z_{g,\max} & Z_{b,\max} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{9.4}$$

where *RGB* are the linearised and normalised (in the range 0 to 1) DAC values. Three measurements are required in order to specify the system matrix for Equation 9.4. The tristimulus values *XYZ* must be measured for each of the guns at the maximum DAC value (255 for a 24-bit system). The *XYZ* values of the red gun at maximum intensity form the first column of the system matrix (Equation 9.4) and the *XYZ* values for the green and blue guns form the second and third columns respectively.

Before making radiometric measurements for characterisation, the monitor should be placed into the position where it will be used and then turned on (for most monitors a deguassing process takes place whenever the monitor is initially powered up). Sufficient time should be allowed for the monitor to warm up. The warm-up time required for a monitor to stabilise after initial power up varies for different devices but can range from 15 minutes to three or more hours (Berns, Gorzynski and Motta, 1993b). For accurate characterisation it is important that the monitor exhibits good spatial independence and channel independence. Spatial independence can be assessed by measuring the colour difference between a white patch displayed in the centre of the screen with a black surround and a identical white patch with a white surround. Berns *et al*. (1993b) measured spatial independence using this technique for five different monitors and found CIELAB colour differences between 2.6 (for the best monitor) and 17.4 (for the worst monitor). Channel independence can be assessed by computing the colour difference between full-field white and the prediction of the full-field white obtained by adding the tristimulus values of the full-field pure red, green and blue conditions. Berns *et al*. found that the channel-independence error can be minimised by reducing the maximum value of luminance that can be displayed. Experience suggests that for many monitors a maximum display luminance of about 80 cd/m^2 provides a suitable compromise between being able to achieve good characterisation and being able to display reasonable brightness levels.

For characterisation purposes it is recommended that patches be displayed at the centre of the monitor against a neutral field set at about one fifth of the luminance of the maximum brightness in order that the measurements are taken in typical conditions.

9.5 Characterisation Example of CRT Display

A LaCie Electron Blue IV 19'' CRT monitor was selected for the study. A Minolta CS-100A portable tristimulus colorimeter was used to obtain the colour measurements. The display was turned on to allow it to reach operating temperature at least 30 minutes before the measurements were taken. Measurements of *Yxy* were obtained for colour patches displayed in the centre of the display against a neutral background. Three sets of measurements were taken; the pure primaries at full strength (Table 9.1), colour ramps (or neutral samples) for determining the nonlinearity (Table 9.2), and some test samples to test the performance of the characterisation (Table 9.3).

The matrix can be directly constructed to convert linear *RGB* to *XYZ*. The colorimetric data for the three primaries is given in Table 9.1.

The equation that relates linear *RGB* to *XYZ* is given as Equation 9.5 and its inverse is Equation 9.6.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 30.24 & 30.19 & 18.62 \\ 17.30 & 61.00 & 8.99 \\ 2.03 & 10.82 & 100.82 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{9.5}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 0.04582 & -0.02152 & -0.00655 \\ -0.01307 & 0.02279 & 0.00038 \\ 0.00048 & -0.00201 & 0.01001 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{9.6}$$

The measurements of nine neutral samples are displayed in Table 9.2.

Table 9.1 *Colorimetric measurements for CRT primaries.*

	R	G	B	X	Y	Z
Red	255	0	0	30.24	17.30	2.03
Green	0	255	0	30.19	61.00	10.81
Blue	0	0	255	18.62	8.99	100.82

Table 9.2 *Colorimetric measurements for CRT neutral samples.*

	d_r	d_g	d_b	X	Y	Z	R	G	B
1	0	0	0	0.37	0.34	0.32	1.9	0.8	0.7
2	25	25	25	3.05	3.36	3.83	10.8	9.7	8.4
3	50	50	50	7.95	8.91	10.55	26.4	26.3	23.3
4	80	80	80	14.78	16.60	20.49	47.5	49.2	45.6
5	120	120	120	26.25	29.30	36.88	84.4	86.4	82.3
6	160	160	160	41.02	45.80	58.12	130.9	135.2	129.9
7	200	200	200	57.48	63.80	82.55	183.8	187.3	185.0
8	220	220	220	66.61	73.70	95.90	213.9	215.7	215.1
9	255	255	255	77.00	85.40	108.72	249.6	250.4	243.1

Table 9.3 *Colorimetric measurements for CRT test samples (the rightmost three columns are predicted data).*

	Measured data						Predicted data		
	d_r	d_g	d_b	X	Y	Z	X	Y	Z
1	110	40	170	21.36	14.70	56.40	21.79	14.57	58.14
2	250	187	160	60.27	63.40	60.64	58.87	61.57	61.13
3	10	180	230	35.76	46.20	95.06	35.21	46.00	93.97
4	5	20	240	18.21	10.20	93.01	17.74	9.45	92.79
5	240	20	5	28.93	17.90	2.62	28.34	17.04	2.07
6	5	240	20	28.95	57.40	12.79	28.16	56.39	11.39
7	34	40	32	5.20	6.40	6.03	4.53	5.56	4.85
8	220	210	195	62.42	69.00	80.89	60.55	67.15	79.01

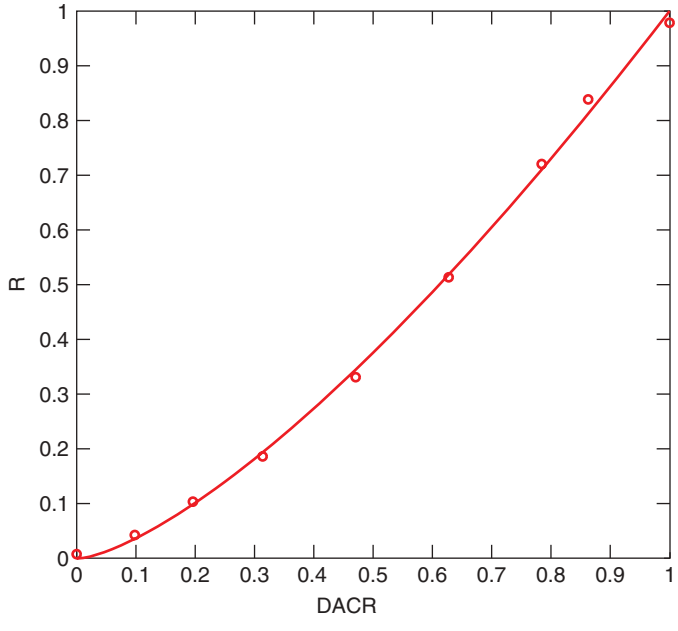

Figure 9.1 *Fitting the nonlinearity of the red channel.*

In Table 9.2 the DAC values were the value used to generate the patches; their equivalent linear RGB values (the rightmost three columns) are obtained using Equation 9.6. Measurements of as few as two neutral patches would allow the parameters of the GOG model to be computed but in practice at least five neutral patches are normally used (Luo, 2003). Figure 9.1 shows the normalised DAC values for the red channel d_r and the corresponding linear red channel values R. The GOG parameters may be determined using a multidimensional optimisation technique such as the simplex algorithm. MATLAB® provides a function called `fminsearch` that works well in this regard.

The characterisation procedure is now complete. The *XYZ* values of the test samples can now be predicted and compared with the measured values to assess the accuracy of the characterisation model. Prediction of *XYZ* values is simple; take the nonlinear DAC values, use the GOG model to generate linear *RGB* values and then use Equation 9.5 to estimate *XYZ*. The average CMC(2;1) colour difference between the measured and predicted *XYZ* values in Table 9.3 is 1.53.

The MATLAB® function `crtdemo` demonstrates how this case study can be implemented in MATLAB®.

```
% ================================================
% *** FUNCTION crtdemo
% ***
% *** function = crtdemo
% *** demonstrates use of GOG model to
% *** characterize a CRT monitor
function crtdemo
% RGB Yxy measured data
rawdata = [255 0 0 17.30 0.610 0.349;...
0 225 0 61.00 0.296 0.598;...
0 0 255 8.99 0.145 0.070;...
0 0 0 0.34 0.360 0.331;...
25 25 25 3.36 0.298 0.328;...
50 50 50 8.91 0.290 0.325;...
80 80 80 16.60 0.285 0.320;...
120 120 120 29.30 0.284 0.317;...
160 160 160 45.80 0.283 0.316;...
200 200 200 63.80 0.282 0.313;...
220 220 220 73.70 0.282 0.312;...
255 255 255 85.40 0.284 0.315;...
110 40 170 14.70 0.231 0.159;...
0 176 240 45.00 0.197 0.248;...
10 180 230 46.20 0.202 0.261;...
5 20 240 10.20 0.150 0.084;...
240 20 5 17.90 0.585 0.362;...
5 240 20 57.40 0.292 0.579;...
34 40 32 6.40 0.295 0.363;...
220 210 195 69.00 0.294 0.325];

for i=1:20
    Y = rawdata(i,4);
    x = rawdata(i,5);
    y = rawdata(i,6);
    z = 1 - x - y;
    X = x*Y/y;
    Z = z*Y/y;
    XYZ(i,:) = [X Y Z];
end
```

```
% use the white of the display for CIELAB
white = XYZ(12,:);
LAB = xyz2lab(XYZ,'user',white);

M = [XYZ(1,1) XYZ(2,1) XYZ(3,1); XYZ(1,2) XYZ(2,2)...
      XYZ(3,2); XYZ(1,3) XYZ(2,3) XYZ(3,3)];
disp(M)

% get the neutral samples
nXYZ = XYZ(4:12,:);
% get their linear RGB values
nRGB = (inv(M)*nXYZ')';
% and the corresponding non-linear RGBs
dacs=rawdata(4:12,1:3)/255;

% compute the GOG values for each channel
x1=linspace(0,1,101);
% red channel
x = [1, 1];
options = optimset;
x = fminsearch('gogtest',x,options,dacs(:,1), nRGB(:,1));
gogvals(1,:) = x;
figure
plot(dacs(:,1),nRGB(:,1),'r*')
y1 = compgog(gogvals(1,:),x1);
hold on
plot(x1,y1,'r-')
% green channel
x = [1, 1];
options = optimset;
x = fminsearch('gogtest',x,options,dacs(:,2), nRGB(:,2));
gogvals(2,:) = x;
hold on
plot(dacs(:,2),nRGB(:,2),'g*')
y2 = compgog(gogvals(2,:),x1);
hold on
plot(x1,y2,'g-')
% blue channel
x = [1, 1];
options = optimset;
x = fminsearch('gogtest',x,options,dacs(:,3), nRGB(:,3));
gogvals(3,:) = x;
hold on
plot(dacs(:,3),nRGB(:,3),'b*')
y3 = compgog(gogvals(3,:),x1);
hold on
plot(x1,y3,'b-')
disp(gogvals)
```

```
% now take the dacs of the test samples
% and linearize them using the gogvals
red = rawdata(13:20,1)/255;
RGB(:,1) = compgog(gogvals(1,:), red);
green = rawdata(13:20,2)/255;
RGB(:,2) = compgog(gogvals(2,:), green);
blue = rawdata(13:20,3)/255;
RGB(:,3) = compgog(gogvals(3,:), blue);

linXYZ = (M*RGB')';
linLAB = xyz2lab(linXYZ,'user',white);
DE = cmcde(LAB(13:20,:),linLAB,2,1);

disp([mean(DE) max(DE)])
end

% function [err] = gogtest(gogs,dacs,rgbs)
% computes the error between measured and predicted
% linearized dac values for a given set of GOG values
% gogs is a 2 by 1 matrix containing gamma and gain
% dacs is an n by 1 matrix containing dac values
% rgbs is an n by 1 matrix that is obtained from a
% linear transform of measured XYZ values
function [err] = gogtest(gogs,dacs,rgbs)

gamma = gogs(1);
gain = gogs(2);

% force to be row matrices
dacs = dacs(:)';
rgbs = rgbs(:)';

if (length(dacs) ~= length(rgbs))
   disp('dacs and rgbs vectors must be the same length');
   err = 0;
   return
end

% compute gog model predictions
for i=1:length(dacs)
   if (gain*dacs(i) + (1-gain)) <= 0
      pred(i)=0;
   else
      pred(i)=(gain*dacs(i) + (1-gain))^gamma;
   end
end

% force to be a row matrix
pred = pred(:)';
```

```
% compute rms error
err = sqrt((sum((rgbs-pred).*(rgbs-pred)))/length(dacs));
end

% function [rgb] = compgog(gogs,dacs)
% computes the linearized RGB values
% from the normalized RGB values
% for a given set of gog values
% gog is a 2 by 1 matrix containing gamma and gain
% dacs is an n by 1 matrix containing dacs
% rgb is an n by 1 matrix of linearized RGB values
function [rgb] = compgog(gogs,dacs)

gamma = gogs(1);
gain = gogs(2);
for i=1:length(dacs)
   if (gain*dacs(i) + (1-gain)) < = 0
      rgb(i)=0;
   else
      rgb(i)=(gain*dacs(i) + (1-gain))^gamma;
   end
end
% force output to be a column vector
rgb=rgb(:);
end
```

The MATLAB® code `crtdemo` shows an example of how the MATLAB® function `fminsearch` can be used for an optimisation task. The function `fminsearch` performs unconstrained nonlinear optimisation using the simplex method of Lagarias *et al*. (1998). A typical call of `fminsearch` is:

```
fminsearch('gogtest',x,options,dacs(:,3), nRGB(:,3));
```

where `x` is a vector containing starting values for the two parameters of the GOG model, and `dacs(:,3)` and `nRGB(:,3)` are vectors containing the linear and nonlinear channel values (in this case, the B channel). The parameter `options` contains default parameters for the optimisation process. The first parameter is a reference to the function which `fminsearch` will use to define the minimisation. In this case, the function `gogtest` is included which computes the rms error between the actual and predicted linear channel values for a particular set of GOG parameters.

```
function [err] = gogtest(gogs,dacs,rgbs)
```

This case study implements the GOG model; it uses neutral samples to estimate the parameters of the nonlinearity and a small number of other samples to test the performance of the characterisation model. Other approaches exist to estimate the parameters of the nonlinearity. For example, rather than neutral samples it is possible to use individual colour ramps for each of the primaries. It would also be possible to avoid the GOG model entirely and use a look-up table (LUT) to relate nonlinear DACS to linear RGB values. In this study only eight samples were used to validate the performance of the characterisation model. However, in a full characterisation study it would be normal to use more than eight test samples. Particularly it would be useful to ensure that the test samples contain both neutral and saturated samples since there is some evidence that the conventional GOG model approach performs particularly badly for very saturated and very dark samples (Oulton, 2011). Imperfections in technology are likely to mean that the performance of display characterisation will always be limited. For example, consideration of Tables 9.1 and 9.2 show that the sum of the luminances of the three primaries at full strength is 87.29 whereas the luminance of the white is only 85.40 suggesting that the device used in the case study suffered from some lack of channel independence. In addition, note that the luminance of the black (where $R = G = B = 0$) is not zero but is 0.34 (Table 9.2); this results in a small error but, in some situations, for example where the display is characterised in a room with substantial ambient illumination, this 'flare' could be much greater and should be taken into account. A modified version of the GOG model known as GOGO[1] (gain-offset-gamma-offset) exists to address this (Berns and Kato, 2002). Other techniques for CRT characterisation use iterative methods. For example, one approach is to use iteration to optimise the linearisation process to ensure correct display of the white point at all intensity levels (Oulton and Porat, 1992; Oulton 2010). A further method (Yoonessi and Kingdom, 2007) uses a calibrated camera to capture an image displayed on a CRT and then (following various corrections) redisplays the corrected captured image as an 'output' image on the CRT (iteration is then used to find the optimal linear transform). Jiménez *et al*. (1998) studied the effect of brightness and contrast settings on the colour-reproduction properties of CRTs and concluded that low to medium brightness and maximum contrast were optimal.

9.6 Beyond CRT Displays

Whereas CRT devices exhibit a power-law relationship between the DAC values and the output luminance, LCD devices often exhibit an electro-optic response that is better modelled as a sigmoidal or S-shaped function (Sharma, 2002). Many LCD manufacturers, however, build correction tables into the video card that result in the LCD response mimicking the response of a CRT with a power law of about 2.0 (Bala, 2003). Bastani *et al*. (2005) studied several different technologies (including CRT and LCD displays and LCD projectors) and reported that linear models worked well on the devices that they tested. Nevertheless, the GOG model has alternatively been found to be inadequate

[1] For the case study data the GOGO model was also implemented but the results were statistically no better than for the GOG model. Of course, in a brighter viewing environment the flare term might be expected to be much greater and this may require implementation of the GOGO model.

for characterising LCDs (Day *et al*., 2004) mainly due to channel interaction and lack of constancy in channel chromaticity (Fu *et al*., 2008).

Day *et al*. (2004) have suggested that the use of 1D LUTs based on sub-samples and linear (or nonlinear) interpolation. They have also noted that there is more than one source of flare and that for LCDs there may be significant radiant output at the black level ($R = G = B = 0$) caused by the liquid crystal having a minimum transmittance well above zero. Following work by Fairchild and Wyble (1998) they suggest the following model to relate *XYZ* to linear *RGB*:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_{r,\max} - X_{k,\min} & X_{g,\max} - X_{k,\min} & X_{b,\max} - X_{k,\min} & X_{k,\min} \\ Y_{r,\max} - Y_{k,\min} & Y_{g,\max} - Y_{k,\min} & Y_{b,\max} - Y_{k,\min} & Y_{k,\min} \\ Z_{r,\max} - Z_{k,\min} & Z_{g,\max} - Z_{k,\min} & Z_{b,\max} - Z_{k,\min} & Z_{k,\min} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \\ 1 \end{bmatrix} \tag{9.7}$$

where the subscript $_{k,\min}$ refers to the black-level radiant output. Of course, this model assumes that the channels are independent and scalable. However, for some LCDs, there can be significant lack of independence (Day *et al*., 2004) and it is also known that the spectral transmittance of liquid crystals can vary with voltage (Ye and Gu, 1999).

Equation 9.7 can be inverted thus:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} X_{r,\max} - X_{k,\min} & X_{g,\max} - X_{k,\min} & X_{g,\max} - X_{k,\min} \\ Y_{r,\max} - Y_{k,\min} & Y_{g,\max} - Y_{k,\min} & Y_{g,\max} - Y_{k,\min} \\ Z_{r,\max} - Z_{k,\min} & Z_{g,\max} - Z_{k,\min} & Z_{g,\max} - Z_{k,\min} \end{bmatrix} \begin{bmatrix} X - X_{k,\min} \\ Y - Y_{k,\min} \\ Z - Z_{k,\min} \end{bmatrix} \tag{9.8}$$

Day *et al*. (2004) applied this inverted equation and measurements of the tristimulus terms to each of three colour ramps (each sampled at 11 points). The measured values were then adjusted by nonlinear optimisation until the average CIEDE2000 colour difference between measured and estimated tristimulus values for all three colour ramps was minimised.

A recent study (Thomas *et al*., 2008) suggests that generalisation of a method known as PLVC (Piecewise Linear assuming Variation in Chromaticity) originally proposed by Post and Calhoun (1989) may have applicability to the characterisation of LCD and LED displays.

10
Characterisation of Cameras

10.1 Introduction

Characterisation is only worthwhile assuming that the device has already been calibrated. In the case of a camera system, this would normally mean that the imaging system, in a wider sense, is controlled. For example, the system could be configured so that the subject is a fixed distance from the camera lens and is illuminated uniformly. The camera itself should have all settings (focus, exposure time, white balance) optimised and fixed. The purpose of calibration is so that the camera is capable of capturing consistent *RGB* values over time. White balance deserves special attention and is a process of adjusting the colour balance in an image to remove unwanted colour castes. Most digital SLR cameras have three settings for white balance; automatic, preset and custom. Automatic white balance is where the camera on-board software uses a 'best-guess' algorithm to estimate the colour of the illumination and correct for it. Preset white balance is where the camera uses a fixed (preset) estimation of the illumination (such as daylight or tungsten). Custom white balance is where the user can take an image of, say, a grey reference card under lighting that will subsequently be used for images. It is very important that automatic white balance (which is the default for many digital cameras) is not used. Custom white balance is obviously appropriate if images are being captured in, say, a viewing cabinet under stable illumination. Many researchers prefer to capture images using the so-called raw file format so that no colour balance is applied by the camera[1]. (The raw file format has the additional advantage in that no gamma is applied to the sensor responses.)

[1] Indeed, many professional photographers also prefer to record in raw file format; this allows full control of white balance and retains all information to give maximum flexibility in post processing.

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.

For camera characterisation it is usually useful to image a chart containing a set of colours of known tristimulus values (Johnson, 2002). Such charts commonly include neutral patches that may be used to linearise the camera *RGB* outputs and coloured patches that may be used to characterise a transform from linearised *RGB* values to CIE *XYZ* values. In the late 1980s a working group of the ANSI IT8 (Image Technology Committee #8) was created to define standard targets to be used in the characterisation of scanners and printers (McDowell, 2002). The IT8 committee chose to colorimetrically define the colours that should appear in the target, but then allow individual manufacturers to produce targets to meet these requirements. Two standards were developed, ANSI IT8.7/1 and ANSI IT8.7/2, for transmission and reflectance modes respectively and they were combined into a single ISO standard (ISO 12641:1997). Two further charts that are sometimes used for device characterisation are the Macbeth ColorChecker chart (which contains 24 patches) and the Macbeth ColorChecker DC chart (which contains over 200 patches). The Macbeth ColorChecker was developed in 1976 and its 24 patches included six neutral colours, red-green-blue and cyan-magenta-yellow primaries, and other important colours such as light and dark skin, sky blue, foliage and so on (McCamy *et al*., 1976). The pigments were selected to be optimally colour constant.

Though the Macbeth ColorChecker DC chart is widely used it is not clear whether the colours selected for use in the chart are optimal and some work has been carried out to determine an optimum set of colours. For example, Hardeberg (1999) proposed a method to select a set of samples whose reflectance factors would be optimal for estimating the spectral sensitivity of camera systems. Cheung and Westland (2006) used similar techniques to propose an algorithm to select n known stimuli out of a larger set of m available stimuli so that the n stimuli would be optimal for colour characterisation of a camera. However, it has been suggested that the outcome of colorimetric characterisation is dependent on the properties of the cross-product matrix encompassing the colour signals (Alsam and Finlayson, 2008); this led to an optimisation technique that showed that the 24 surfaces of the Macbeth ColorChecker could be accounted by 13 appropriately chosen spectra.

10.2 Correction for Nonlinearity

Although the response of CCD material is approximately linearly related to the intensity of the light falling on it, it is unlikely that the *RGB* outputs of a scanner or digital camera will be linearly related to the *XYZ* tristimulus values of the surfaces in the scene. The raw channel responses are invariably processed by on-board software in an attempt to generate *RGB* responses that are closer to a linear transform of the colour-matching functions than are allowed by current methods for producing filters. Typically, the raw *RGB* values may be transformed by a 3×3 linear transform to give the output *RGB* values (many cameras produce *RGB* values close to sRGB.). Furthermore, many manufacturers impose a nonlinearity during this 'matrix-mixing' stage to approximately match the inverse of the nonlinearity of display systems or as part of the solution to provide high signal-to-noise ratios. It is therefore common to consider a correction for nonlinearity as the first-stage of a camera- or scanner-characterisation process. For a

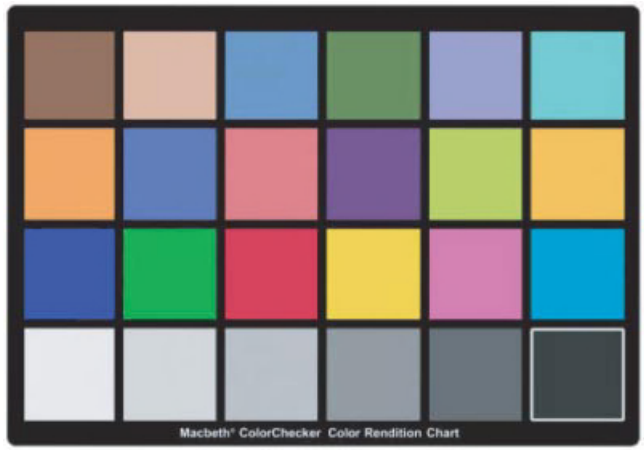

Figure 10.1 *Image of a Macbeth ColorChecker chart.*

digital camera we may, for example, consider a relation of the form:

$$C_i = C_i'^{p} \tag{10.1}$$

where C_i' is the response of the camera channel i, p is an exponent for the channel, and C_i is a transformed camera response for that channel that is linearly related to the channel input. A set of grey-scale samples (see Figure 10.1) is often used to empirically determine the exponent p. Thus, the camera responses are determined for a range of grey samples under a constant and known light source. The XYZ values can then easily be computed for the grey samples and linearisation is then achieved by finding the value of the exponent p such that there is a linear relationship between C_i and Y for the set of grey samples.

However, strictly, in order to correctly determine p (according to Equation 10.1) we need to know the values of C_i and C_i' for each of the grey patches. The value of C_i' is the input to the channel[2] and the value of C_i is the channel response possibly subject to nonlinear processing. The input to the channel may easily be computed if the spectral reflectance of the sample, the spectral power of the illumination, and the channel sensitivity are all known (Thomson and Westland, 2002). Unfortunately, it is usually the case that the channel sensitivity is not known. If p is determined so that there is a linear relationship between C_i and Y, then it is clear that the Y colour-matching function is being used as a crude estimate of the channel sensitivity. For grey samples imaged under an equal-energy illuminant Y is a reasonable estimate of C_i' since they would be approximately linearly related. However, this proportionality decreases for increasingly chromatic samples and light sources. It is interesting to note that if the channel sensitivity was known, and the true values of C_i' could be computed, then it would not be necessary to use grey samples to linearise the channel responses; any

[2] By input to the channel we mean the response of the channel before any processing.

samples could be used. Since the spectral sensitivities are generally not known, however, linearisation is typically established using achromatic samples (Sun and Fairchild, 2001). The grey samples of the Munsell ColorChecker provide a convenient grey scale for this purpose. Some workers choose to linearise the camera responses with the appropriate tristimulus values (X for the red channel, Y for the green channel and Z for the blue channel) or with the average reflectance of the samples.

10.3 Correction for Lack of Spatial Uniformity

Obtaining accurate CIE colour information from camera data obtained from images captured in typical consumer environments is extremely difficult. Researchers normally control the illumination and viewing conditions of the image-capture process if accurate camera characterisation is required. Even under such laboratory-controlled conditions, however, it is normally extremely difficult to ensure that the illumination of the imaging target is even and some spatial correction of the captured image is usually required.

The following equation has been used, for example, by Hardeberg (1999) to correct linearised channel responses C_i at each pixel:

$$C_{Si} = \frac{\left(\overline{C}_{Wi} - \overline{C}_{Bi}\right)\left(C_i - C_{Bi}\right)}{\left(C_{Wi} - C_{Bi}\right)} \tag{10.2}$$

where C_{Wi} and C_{Bi} are the linearised responses for channel i to a white and black uniform field at each pixel value. $\overline{C}_{Wi}$ and $\overline{C}_{Bi}$ are mean values pooled over all pixels.

10.4 Characterisation

General linear and nonlinear relationships are typically used to convert camera RGB values into CIE XYZ values and can be derived from linear models (Brainard and Wandell, 1990).

If a colour camera satisfies the Luther condition[3], the camera spectral sensitivities are linearly related to the CIE colour-matching functions. Such a camera would effectively be an imaging colorimeter. However, this is extremely difficult to accomplish (Holm, 2006). For colorimetric cameras the relationship between linearised and spatially corrected camera RGB responses and corresponding CIE XYZ values is a 3×3 linear transform. However, in practice the vast majority of digital cameras do not satisfy the Luther condition and therefore nonlinear modelling is frequently employed.

Hong *et al*. (2000) found that a 11×3 polynomial matrix (see Table 10.1), essentially a second-order matrix with a single third-order term, was effective. They claimed that the third-order term (RGB) and a 'black term' (1) were particular important. Hong *et al*.

[3] The Luther condition is also known as the Maxwell-Ives criterion; the spectral sensitivity of the channels of a camera that satisfies this condition is a linear combination of the CIE CMFs.

Table 10.1 *Selected matrices from the study by Cheung et al. (2004).*

$m \times 3$	Augmented matrix
3×3	[*RGB*]
5×3	[*R G B RGB* 1]
8×3	[*R G B RG RB GB RGB* 1]
11×3	[*R G B RG RG GB* R^2 G^2 B^2 *RGB* 1]

also suggested that because of the existence of eye-camera metamerism, the transfer matrices derived by colour samples from one set of targets or dyes do not produce the same accuracy when they are used for predicting colour samples of another material.

It is also possible to use artificial neural networks (Kang and Anderson, 1992; Cheung and Westland, 2002). Cheung *et al*. (2004) conducted a study of camera characterisation and specifically compared the ability of polynomials and neural networks to carry out this task. In their study, using an Agfa StudioCam camera, 192 samples from the centre of the Macbeth DC Colorchecker were used as training samples and the 24 samples of the Macbeth Colorchecker were used as test samples. The training samples were used to train neural networks and to determine the coefficients of various polynomial transforms. The camera *RGB* values were linearised and corrected for spatial nonuniformity (of lighting and camera CCD response) and used to predict CIE *XYZ* values using either the neural network or the polynomial. The test samples were used to assess the characterisation performance for the various models that were used.

Various polynomial transforms were attempted as detailed in Table 10.1. These polynomials always attempted to map camera *RGB* values to CIE tristimulus values. A $192 \times m$ matrix was constructed from the training set where each row contained the *m RGB* terms (see Table 10.1 for some example matrices from the study) for one of the samples. A linear system is then assumed where the $192 \times m$ matrix is multiplied by an $m \times 3$ matrix of coefficients to yield the 192×3 matrix of tristimulus values. The values of the coefficients were determined using the MATLAB® backslash operator.

Figure 10.2 shows the median CIELAB colour differences of the $m \times 3$ polynomials whereas Figure 10.3 show the training and testing error for the neural networks with n hidden layers.

It is evident that the performance of the best neural network and polynomial models produces a test error of about 2 CIELAB units. It is not surprising that the two systems should provide equivalent performance. Training the neural networks can be quite time-consuming, however, and there are many parameters to determine such as the number of hidden units, the transfer functions for the units in the network, the learning rule, the parameters of the learning rule, and so on and so forth. Furthermore, each time the network is trained from different random initial weights a different transform is achieved. The results shown in Figure 10.3 are in fact the average results from training each network five times. By contrast, it is relatively straight forward to develop the polynomial systems and relatively fewer decisions regarding parameters need to be

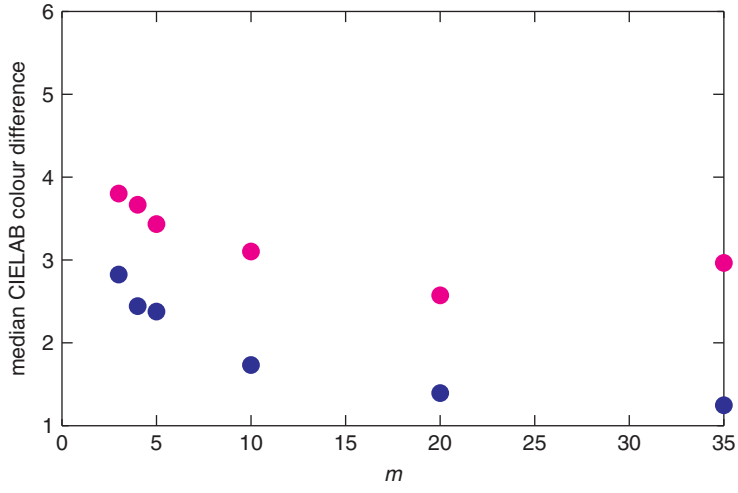

Figure 10.2 *Median colour differences for training (blue symbols) and test (magenta symbols) data for polynomials of various size (with m-terms).*

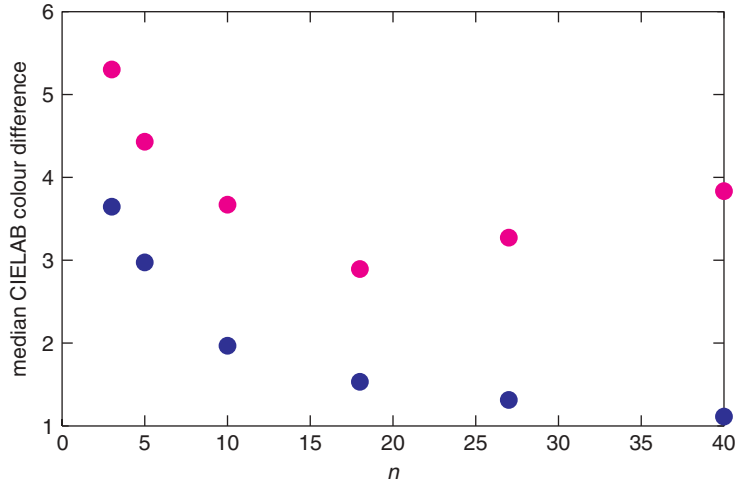

Figure 10.3 *Median colour differences for training (blue symbols) and test (magenta symbols) data for neural networks of various size (with n hidden units).*

made. There therefore seems little reason therefore to use neural networks for camera characterisation (Cheung *et al*., 2004). Cheung *et al*. also looked at the impact of the size of the training set and found that both neural networks and polynomials showed a surprising degree of robustness with test performance[4] only degrading substantially

[4] Note, for the polynomial system in particular, that the training error reduces for small training set sizes. If a proper independent test set was not used to evaluate the performance of the modes then the use of a small set of training sample could lead to an optimistic view of the performance of the characterisation models.

for training set sizes less than 50. This study also showed that a third-order polynomial model or less was adequate for camera characterisation.

The methods described so far can be described as data-driven methods for camera characterisation since they use data sets to optimise arbitrary linear and nonlinear functions. These methods are widely used in the literature. However, model-driven methods are also used by some researchers and these typically attempt to determine the spectral sensitivity functions of the camera channels. Barnard and Funt (2002) have argued that it is best to jointly fit the linearisation and sensor response functions.

10.5 Example Characterisation of a Digital Camera

Obtaining a configuration where there is little systematic spatial variation in camera response over the imaging field is extremely difficult. Therefore, for this example, images were captured used a commercially available system, DigiEye[5], that is known to provide extremely stable and uniform illumination (daylight simulator). The camera used in the system was a Nikon D70 SLR and three images[6] were captured: image of the Macbeth ColorChecker DC (Figure 10.4), a white card (Figure 10.5) and a black (Figure 10.6). The black image was obtained by taking an image with the lens cap in place.

Note (Figure 10.4) that a grey-scale is provided in the central portion of the Macbeth ColorChecker DC. The *RGB* values of these grey patches were captured and are listed in Table 10.2 along with corresponding normalised CIE Y values.

Figure 10.7 shows the nonlinear nature of the Nikon D70 response. The solid lines are fits using Equation 10.1 (obtained using the `fminsearch` function) and the optimised

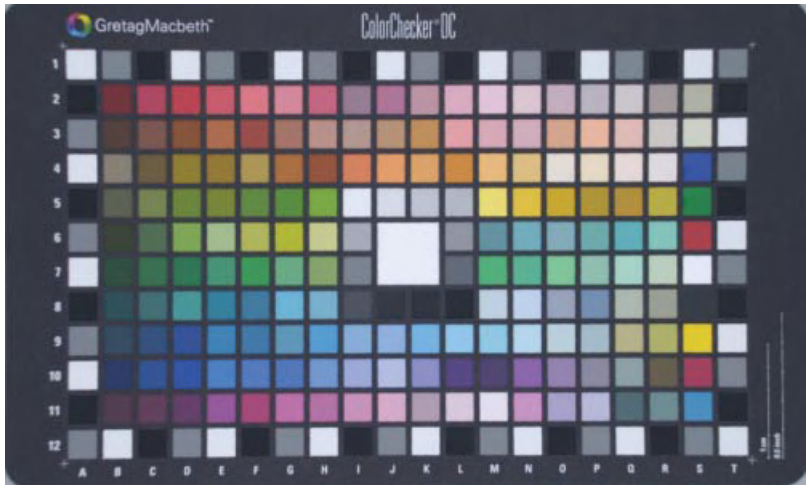

Figure 10.4 *Image of ColorChecker DC chart.*

[5] www.digieye.co.uk/

[6] Note that the images shown in the figures are slightly cropped versions of the original captured images.

Figure 10.5 *Image of white card to be used for spatial correction.*

Figure 10.6 *Black image to be used in spatial correction.*

exponents for the fits were 1.97 (red channel), 2.19 (green channel) and 2.89 (blue channel). Figure 10.8 shows the effect of the linearisation (Equation 10.1) applied to Figure 10.4.

Two MATLAB® functions have been provided to demonstrate camera characterisation. The first, `camera_demo.m`, takes the original captured image and applies linearisation and/or spatial correction. The data in Table 10.2 are provided in `grey.mat` to enable the linearisation process. Spatial correction is applied according to Equation 10.2. The function is called thus:

```
camera_demo(linflag, spflag);
```

where **linflag** and **spflag** are flags set to zero or one (and specify whether linearisation and/or spatial correction should be applied respectively. The function processes the image of the ColorChecker and then extracts the *RGB* values of the individual colour patches (averaging over about 100 pixels in each case). A total of 240 colour patches are extracted

Table 10.2 *RGB and corresponding Y values for grey patches from captured ColorChecker DC image.*

R	G	B	Y
225.53	231.53	245.48	0.9443
165.24	171.28	185.24	0.3556
124.41	132.28	145.88	0.1894
75.70	82.41	96.62	0.0807
211.69	217.60	231.60	0.7367
231.73	235.98	247.98	1.0000
48.31	53.99	66.66	0.0492
190.55	195.98	210.15	0.5948
43.94	49.74	62.16	0.0396
177.97	183.97	197.33	0.4685
143.98	150.07	164.06	0.2700
98.89	106.11	121.79	0.1275
37.73	43.61	54.49	0.0347

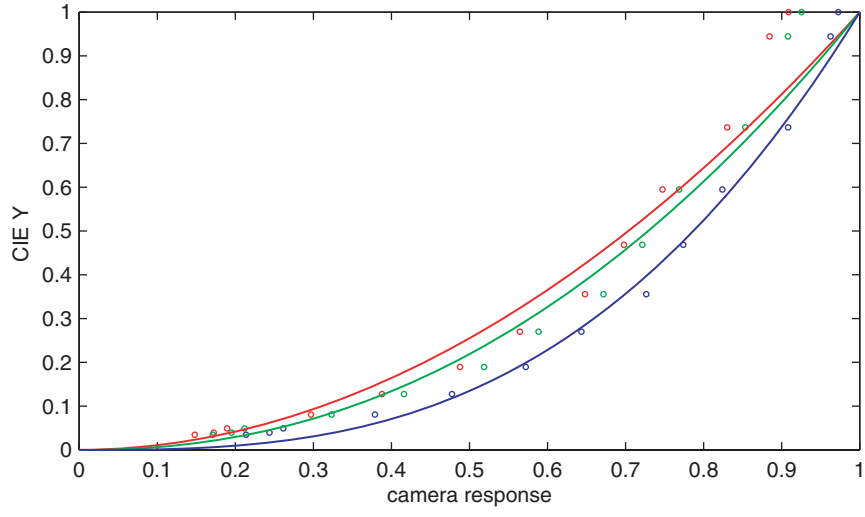

Figure 10.7 *The nonlinear response of the Nikon D70 camera shown for the red (red symbols), blue (blue symbols) and green (green symbols) channels. The solid lines are fitted using a power law function.*

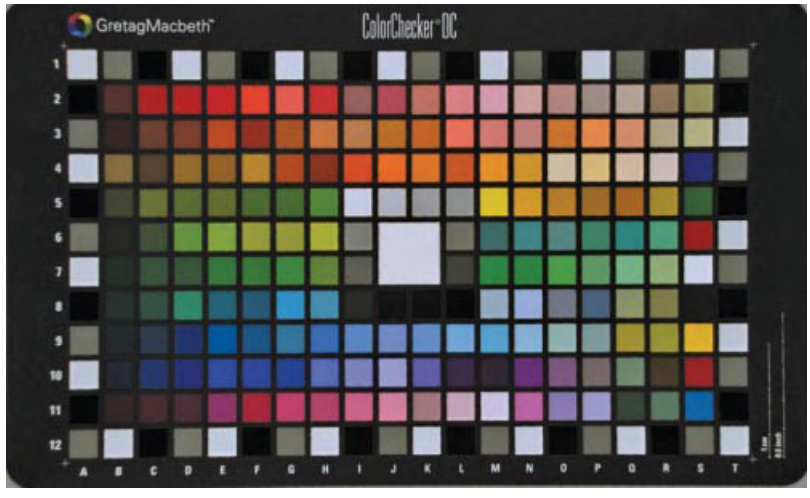

Figure 10.8 *Image of ColorChecker DC chart after linearisation.*

and the *RGB* values of these are saved in a file `rgb_cc.mat`. A corresponding file, `xyz_cc.mat` is provided containing CIE *XYZ* values of the 240 patches. These file will be used by the second MATLAB® function to perform characterisation. The `camera_demo` function is listed in the following.

```
% ====================================================
% *** FUNCTION camera_demo
% ***
% *** function camera_demo(linflag, spflag)
% *** demonstrates camera characterisation
% *** performs linearisation and spatial correction
% *** linflag and spflag are set respectively
% *** requires cc.tif and grey.mat are in the path
% *** see also camera_demo1
% ====================================================
function camera_demo(linflag, spflag)
data = double(imread('cc.tif','tif'))/255;
imshow(uint8(255*data))

if (linflag) % perform linearisation
    load grey.mat
    % greyy 13x1
    % greyrgb 13x3

    figure
    greyrgb=greyrgb/255;
    plot(greyrgb(:,1),greyy,'ro')
    hold on
```

```
plot(greyrgb(:,2),greyy,'go')
hold on
plot(greyrgb(:,3),greyy,'bo')

% compute nonlinearity of each channel
x1=linspace(0,1,101);
% red channel
x = 1;
options = optimset;
x = fminsearch(@gtest,x,options,greyrgb(:,1), greyy(:,1));
gamma(1) = x;
y1 = x1.^gamma(1);
hold on
plot(x1,y1,'r-');
% green channel
x = 1;
options = optimset;
x = fminsearch(@gtest,x,options,greyrgb(:,2), greyy(:,1));
gamma(2) = x;
y1 = x1.^gamma(2);
hold on
plot(x1,y1,'g-');
% blue channel
x = 1;
options = optimset;
x = fminsearch(@gtest,x,options,greyrgb(:,3), greyy(:,1));
gamma(3) = x;
y1 = x1.^gamma(3);
hold on
plot(x1,y1,'b-');

greyrgb(:,1) = greyrgb(:,1).^gamma(1);
greyrgb(:,2) = greyrgb(:,2).^gamma(2);
greyrgb(:,3) = greyrgb(:,3).^gamma(3);

figure
plot(greyrgb(:,1),greyy,'ro')
hold on
plot(greyrgb(:,2),greyy,'go')
hold on
plot(greyrgb(:,3),greyy,'bo')
data(:,:,1)=data(:,:,1).^gamma(1);
data(:,:,2)=data(:,:,2).^gamma(2);
data(:,:,3)=data(:,:,3).^gamma(3);

disp(gamma)

imwrite(uint8(255*data),'cc_lin.tif','tif');
figure
```

```
    imshow(uint8(255*data))
else
    gamma = [1 1 1]; % needed in case spatial correction is called
end

if (spflag)
    % spatial correction
    grey = (double(imread('ccgrey.tif','tif')))/255;
    grey(:,:,1)=grey(:,:,1).^gamma(1);
    grey(:,:,2)=grey(:,:,2).^gamma(2);
    grey(:,:,3)=grey(:,:,3).^gamma(3);
    black = (double(imread('ccblack.tif','tif')))/255;
    black(:,:,1)=black(:,:,1).^gamma(1);
    black(:,:,2)=black(:,:,2).^gamma(2);
    black(:,:,3)=black(:,:,3).^gamma(3);

    meangrey = mean(mean(grey));
    meanblack = mean(mean(black));

    data(:,:,1) = (meangrey(1)-meanblack(1))*(data(:,:,1)-
       black(:,:,1))./(grey(:,:,1)-black(:,:,1));
    data(:,:,2) = (meangrey(2)-meanblack(2))*(data(:,:,2)-
       black(:,:,2))./(grey(:,:,3)-black(:,:,2));
    data(:,:,3) = (meangrey(3)-meanblack(3))*(data(:,:,3)-
       black(:,:,3))./(grey(:,:,3)-black(:,:,3));

    imwrite(uint8(255*data),'cc_lin_sp.tif','tif');
    figure
    imshow(uint8(255*data))
end

index = 0;
for col = 80:35:745
    for row=65:35:450
        index = index+1;
        rgb(index,:)=mean(mean(data(row-5:row+5,col-5:col+5,:)));
    end
end
save 'rgb_cc.mat' rgb

end

function [err] = gtest(x,rgb,y)
% force to be row matrices
rgb = rgb(:)';
y = y(:)';
gamma=x;
if (length(rgb)~=length(y))
    disp('vectors must be the same length');
```

```
    err = 0;
    return
end

% compute predictions with gamma
for i=1:length(rgb)
   if (rgb(i)) < = 0
      pred(i)=0;
    else
      pred(i)=rgb(i)^gamma;
   end
end
pred = pred(:)';
% compute rms error
err = sqrt((sum((y-pred).*(y-pred)))/length(y));
end
```

The second function that has been provided is `camera_demo1.m` which is called without arguments to provide linear, second-order and third-order characterisation modelling to predict *XYZ* from *RGB*. The *RGB* values are obtained from the `rgb_cc.mat` file that is generated by `camera_demo`. Note, however that the full 240 sets of RGB values are not all used in this characterisation. This is because some of the patches (in the second column from the right of Figure 10.4 contain glossy patches of quite a different nature to the majority of patches which are matte) are constructed from a glossy material and the neutral patches that surround the ColorChecker are also removed. This results in 170 patches being available for the characterisation. The `camera_demo1` function is shown below:

```
% ================================================
% *** FUNCTION camera_demo
% ***
% *** function camera_demo1()
% *** demonstrates camera characterisation
% *** performs linear, second-order and third-order
% *** modelling to predict XYZ from RGB
% *** requires rgb_cc.mat and xyz_cc.mat are in the path
% *** see also camera_demo
function camera_demo1

load rgb_cc.mat
load xyz_cc.mat
% rgb 240x3
% xyz 240x3

% only use a sub-set of the 240 samples for the analysis
trgb = rgb([14:23 26:35 38:47 50:59 62:71 74:83 86:95 98:107
     110:119 122:131 134:143 146:155 158:167 170:179 182:191
     194:203 206:215],:)/255;
```

```
txyz = xyz([14:23 26:35 38:47 50:59 62:71 74:83 86:95 98:107
     110:119 122:131 134:143 146:155 158:167 170:179 182:191
     194:203 206:215],:);

% find the linear transform between trgb and txyz
M1=txyz'/trgb';
% use this transform pxyz1 = (M1*trgb')';
% calculate performance
pxyz1(pxyz1<0)=0;
lab = xyz2lab(txyz,'d65_64');
lab1 = xyz2lab(pxyz1,'d65_64');
de1 = cielabde(lab, lab1);
disp('mean and max colour differences for linear transform');
disp([mean(de1) max(de1)])

% now perform a second-order transform
trgb2 = [trgb trgb(:,1).^2 trgb(:,2).^2 trgb(:,3).^2 trgb(:,1).
       *trgb(:,2) trgb(:,1).*trgb(:,3) trgb(:,2).*trgb(:,3)
       ones(170,1)];
M2=txyz'/trgb2';
pxyz2 = (M2*trgb2')';
pxyz2(pxyz2< 0)=0;
lab = xyz2lab(txyz,'d65_64');
lab2 = xyz2lab(pxyz2,'d65_64');
de2 = cielabde(lab, lab2);
disp('mean and max colour differences for 2nd-order transform');
disp([mean(de2) max(de2)])

% now perform a third-order transform
trgb3 = [trgb trgb(:,1).^2 trgb(:,2).^2
       trgb(:,3).^2 trgb(:,1).*trgb(:,2) trgb(:,1).*trgb(:,3)
       trgb(:,2).*trgb(:,3)...
    trgb(:,1).^3 trgb(:,2).^3 trgb(:,3).^3...
    trgb(:,1).*trgb(:,1).*trgb(:,2) trgb(:,1).*trgb(:,1).
       *trgb(:,3)...
    trgb(:,2).*trgb(:,2).*trgb(:,1) trgb(:,2).*trgb(:,2).
       *trgb(:,3)...
    trgb(:,3).*trgb(:,3).*trgb(:,1) trgb(:,3).*trgb(:,3).
       *trgb(:,2)...
    ones(170,1)];
M3=txyz'/trgb3';
pxyz3 = (M3*trgb3')';
pxyz3(pxyz3<0)=0;
lab = xyz2lab(txyz,'d65_64');
lab3 = xyz2lab(pxyz3,'d65_64');
de3 = cielabde(lab, lab3);
disp('mean and max colour differences for 3rd-order transform');
disp([mean(de3) max(de3)])

end
```

Table 10.3 *Mean CIEDE2000 colour differences for the 170 colour patches used in the camera characterisation demo.*

	ΔE linear	ΔE 2nd-order	ΔE 3rd-order
No linearisation or spatial correction	14.05	3.65	1.09
With linearisation but no spatial correction	4.76	1.68	1.41
No linearisation but with spatial correction	10.04	3.63	1.09
With linearisation and spatial correction	4.74	1.65	1.37

Now that we have the `camera_demo` and `camera_demo1` functions it is possible to evaluate the characterisation performance under various conditions and for various mathematical transforms. For example:

```
camera_demo(0,0); camera_demo1;
```

will provide mean colour differences for the three mathematical models without any linearisation or spatial correction being applied. Table 10.3 summarises the results obtained from the four possible configurations.

The results in Table 10.3 indicate that better performance is achieved with more complex polynomial transforms (that is, as we move from left to right in the table). However, this indicates only that memorisation performance improves; an additional set of data would be needed to evaluate generalisation performance in any full study. The results provided here are only an example to understand the process of camera characterisation. Note that the effect of spatial correction is small; this is most likely because the images that were captured were already relatively uniformly illuminated because of the DigiEye system that was used for their capture. The effect of the linearisation is dramatic however. The performance of the linear transform, in particular, is greatly improved if the data are linearised before being used. The performance of the third-order transform is slightly worsened by the application of the linear transform.

11
Characterisation of Printers

11.1 Introduction

Physical models tend to play a more important role in the characterisation of printers than they do with other imaging devices. One reason for this is that the relationship between printer inputs and CIE tristimulus values is usually extremely nonlinear. In addition, however, there is a great deal of theory that has been developed for the prediction of the colour of printing inks from colourant concentration values in a wider context. The Kubelka-Munk theory, for example, has been used for more than half a century to predict spectral reflectance from colourant concentration values. Artificial neural networks have also been used quite widely to find mappings between vectors of colourant concentration values and spectral reflectance values. Numerous technologies are used in printers and this is another reason why different and specific models are used to characterise the devices. Most printers use three or four primaries: cyan, magenta, yellow and black. Note that the primaries of a subtractive colour mixing process are quite different from those (typically red, green and blue) for an additive colour mixing process. For both additive and subtractive devices the primaries are normally selected to enable the greatest gamut of colours to be reproduced. In a subtractive process, the intensities of the red, green and blue light in the print are indirectly controlled by the amount of the cyan, magenta and yellow ink deposited respectively. Some printers – typically dye-sublimation printers – operate by depositing a layer of ink where the thickness of the ink is varied to control the colour of the print. Other printers, however, such as most laser printers, use a half-tone process. For half-tone printers a fixed thickness of ink is deposited in a pattern of dots and tonal and colour variation is achieved by varying either size or frequency of the dots. It is not unreasonable therefore that different

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.

physical models are used for different printers depending upon the technology that the printer uses. Nevertheless, the aim of printer characterisation is the same as in camera or monitor characterisation. Device coordinates (cyan, magenta, yellow and black) are converted into device-independent CIE *XYZ* values. In this chapter, the use of models for the characterisation of printers is described. There is particular emphasis on the Kubelka-Munk and Neugebauer models for device characterisation of half-tone printers and of more general techniques for the characterisation of continuous-tone printers. In this chapter, two examples of printer characterisation are detailed; one for a half-tone printer and one for a continuous-tone printer.

11.1.1 Physical Models

Characterisation of input and display devices is predominantly achieved through linear and nonlinear transforms but although these techniques are also often used for the characterisation of printers, physical models are also important for these devices. Physical printer models can be categorised into two types (Green, 2002b): (i) those that aim to predict the relationship between reflectance and dot area or colorant strength; (ii) those that predict the colour of different colorant combinations, in terms of either colorimetry or spectral reflectance. It may be useful to consider these two models as processes of colorant and colour prediction respectively and to recognise that they are inversely related. Thus many models can be used to predict reflectance or tristimulus values from colorant information but can then be inverted to predict colorant information.

Many printing systems print solid-colour ink in a dot pattern. Such halftone systems provide tonal variation by varying either the size of the dots or their frequency. The measured reflectance of a half-tone system may be predicted by spatially averaging the reflectance of the dots and the substrate on which the dots are printed. A weighted average for each pixel in the image is usually computed based upon the proportional areas of the dots and the substrate. Models such as Neugebauer and Murray-Davies are used for this purpose and such models can also take into account mechanical and optical dot gain. Mechanical dot gain is the phenomenon where the printed dot is physically larger than it should be because of ink spreading during the printing process. Optical dot gain is where there is an apparent gain in the size of the dot caused by scattering the substrate. Substrate scattering is responsible for light being absorbed by the ink dot even when it strikes the substrate directly on an unprinted area. When more than one colour is printed, the second colour can overprint the first and the Neugebauer model must include the colour of the substrate, the primary colours and the overprint colours. For a typical printing system the number of possible overprint colours is usually quite small and therefore it may not be unreasonable to measure them directly. In certain circumstances, however, it may be necessary to predict the overprint colour and the Kubelka-Munk theory may be used for this purpose (Bala, 2003). The Kubelka-Munk theory characterises each colorant by absorption K and scattering S coefficients at each wavelength. The theory can be difficult to apply since specific calibration samples are required to allow the estimation of the K and S values (Nobbs, 1985). Methods are available to allow the use of characterisation data derived from one substrate to be used on a differently coloured substrate (Shaw *et al*., 2003).

11.1.2 Neural Networks

The field of artificial neural networks (ANNs) defines a set of computational methods that were inspired from studies of how humans process information to solve problems. There are a great many different types of ANNs and readers are recommended to study the extensive literature that is now available to explain the principles and algorithms of neural computing (e.g. Rumelhart and McClelland, 1986; Kohonen, 1988; Aleksander and Morton, 1991; Haykin, 1994). This chapter includes only a cursory analysis of just one class of neural network known as a multilayer perceptron (MLP). Despite the large variety of network structures that have been developed, the majority of practical applications of neural computing are in fact based upon MLPs.

An MLP consists of layers of processing units known as neurones or simply as units. Each unit receives input and performs some function upon this input to produce an output. The function between input and output for any unit is known as the activation function or the transfer function and is normally nonlinear. A typical nonlinear transfer function is the sigmoid (S-shaped) function but linear transfer functions are sometimes used for the units in the output layer. The input for each unit is the weighted sum of the outputs from all of the units in the previous layer. The units in the first layer (known as the input layer) receive their input from an input vector and those in the last layer (known as the output layer) generate an output vector. Each unit in the hidden and output units also receives weighted input from a bias unit whose output is fixed at unity. The network as a whole can be thought of as a universal function approximater that attempts to find a mapping between input vectors and output vectors (Figure 11.1). Such networks are interesting because, in principle, they can perform any valid mapping to any arbitrary degree of accuracy. A valid mapping is one that is computable.

The number of units in the input and output layers are determined from the nature of the problem being solved. If for example the network is being used to perform a mapping between a four-dimensional vector and a one-dimensional vector then the number of units in the input and output layers would be four and one respectively. However, the number of hidden layers and the number of units in each hidden layer must be determined empirically.

For ANNs it is important to distinguish between the training mode and the testing mode. During the training mode, example of input-output pairs are presented to the

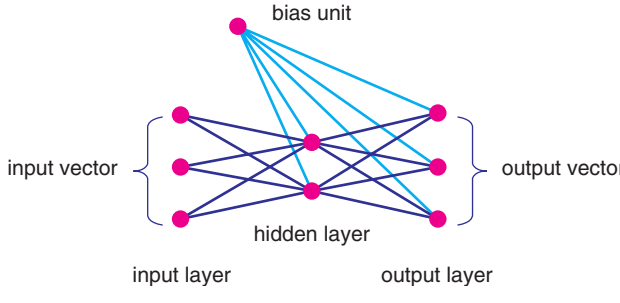

Figure 11.1 *Schematic diagram showing an artificial neural network with one hidden layer of processing units.*

network, the error between the desired output and the actual output (using the current set of weights) is computed, and the values of the weights are modified to reduce this error. This process is repeated for each input-output pair in the training set and the presentation of the whole training set in this way is known as a training epoch. Training may require thousands or even hundreds of thousands of epochs and typically the training procedure is very computationally intensive. However, at the end of the training period the values of the weights are fixed. During the testing mode, input vectors are presented to the network and output vectors are computed. The performance of the network in testing mode using the data from the training set is known as the training error. A common problem with MLPs is that they are prone to over-fitting the training data. As the number of hidden layers or units in the network increases then the training error should decrease. In the limit a sufficiently complex MLP can produce a training error of zero; such a network, however, may exhibit poor generalisation performance. Generalisation is the ability of the network to perform using data that was not used during the training period. A second data set, known as a testing data set, is therefore used to determine the testing error. Of course, the training and testing data sets should be drawn from the same population so that they both represent, in a statistical sense, the problem being addressed by the network.

11.2 Characterisation of Half-Tone Printers

11.2.1 Correction for Nonlinearity

If we consider an ink printed on a substrate in a half-tone pattern and denote the reflectance of the unprinted substrate by P_w and the reflectance of the solid ink by P_s, then the Murray-Davies relationship (Yule, 1967) predicts the measured reflectance P of the print. The value of P is related to the sum of the reflectance factors of the two components weighted by their fractional area coverage thus:

$$P = aP_s + (1 - a)P_w \tag{11.1}$$

where a is the fractional area of coverage of the ink.

Equation 11.1 can be inverted to predict the proportional dot area by:

$$a = (P_w - P)/(P_w - P_s). \tag{11.2}$$

The simple Murray-Davies equation does not take dot gain into account. Yule and Nielsen (1951) proposed a correction to the Murray-Davies equation thus:

$$P = (a\sqrt{P_s} + (1 - a)\sqrt{P_w})^2 \tag{11.3}$$

Equation 11.3 results in a nonlinear relationship between the area coverage a and the resulting reflectance P. The generalised Yule-Nielsen equation allows an exponent n so that:

$$P = (aP_S^{1/n} + (1 - a)P_W^{1/n})^n \tag{11.4}$$

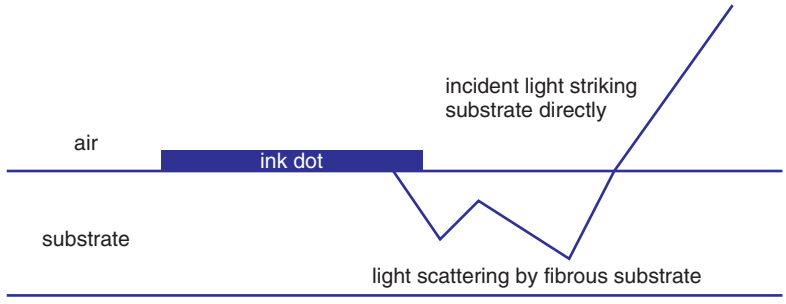

Figure 11.2 *Optical gain results when light strikes unprinted substrate but is then absorbed as it travels upwards out of the substrate in a printed area because of scattering.*

where n is usually given a value between 1.0 (for a glossy substrate) and 2.0 (for a matt substrate). This nonlinear relationship is required to account for the phenomenon of optical dot gain. Optical dot gain is the phenomenon that half-tone prints usually appear darker than expected (based on Equation 11.1) because some light that strikes the unprinted substrate is absorbed by the ink dots. This occurs because of scattering of light in the substrate (Figure 11.2).

In addition to optical dot gain it is also necessary to consider mechanical dot gain which is the phenomenon where the printed dots are usually physically larger than their target sizes because of flow of the wet ink when it is applied to the substrate. The effect of mechanical dot gain is that a nonlinear relationship exists between the digital input count d and the dot coverage a. For given values of d and P, the optimum area coverage a may be computed using:

$$a = \frac{\sum_{\lambda}(P_s^{1/n}(\lambda) - P^{1/n}(\lambda))(P_s^{1/n}(\lambda) - P_w^{1/n}(\lambda))}{\sum_{\lambda}(P_s^{1/n}(\lambda) - P_w^{1/n}(\lambda))} \tag{11.5}$$

where it is assumed that there are no inter-colourant interactions (Bala, 2003). Thus, P_j is measured for a number of levels d_j and then Equation 11.5 is used to determine a_j. This procedure yields pairs of $[d_j \; a_j]$ from which a continuous function can be derived that maps digital count d to dot area coverage a. Some alternative methods for determining dot areas that minimise the error in CIELAB colour-difference units are also available (Bala, 1999).

11.2.2 Neugebauer Models

For half-tone printers, device-independent representation is normally obtained by finding a mapping between the proportional dot coverages for the inks and the spectral reflectance of the print, from which it is then trivial to compute XYZ values. Alternative methods may use neural networks to find a mapping either from dot coverages to reflectance or even directly to CIE XYZ values. The most common method for predicting reflectance involves the Neugebauer model which takes into account the various overlapping binary mixtures. For example, if cyan, magenta and yellow inks are considered then the resulting reflectance will be a function of the reflectances of the unprinted substrate P_w, the three solid colours (P_c, P_m, P_y) and the four overlap colour combinations of cyan + magenta

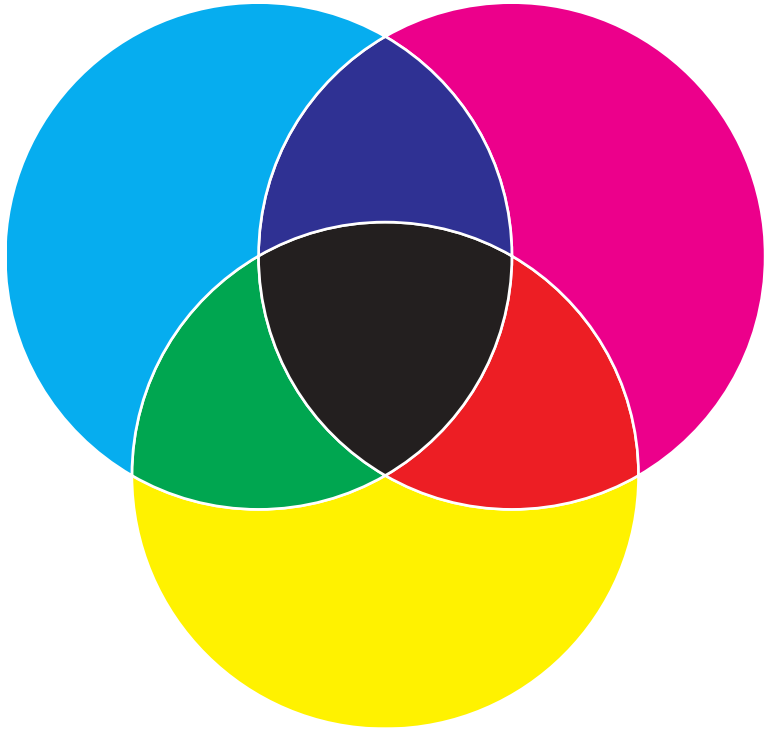

Figure 11.3 *Over-printing with CMY primaries results in eight differently coloured areas (cyan, magenta, yellow, red, green, blue, white and black).*

(blue P_b), cyan + yellow (green P_g), yellow + magenta (red P_r), and black P_k (see Figure 11.3).

If the fractional areas of these eight areas are represented by a_c, a_m and so on, then we can write:

$$P = a_w P_w + a_c P_c + a_y P_y + a_b P_b + a_g P_g + a_y P_y + a_k P_k \tag{11.6}$$

It is evident that the Neugebauer model is a simple extension of the Murray-Davies equation (Equation 11.1) that assumes that the reflectance of a spatial area is the additive combination of the reflectances of the primary colours and their overlapping areas. In the original Neugebauer equations the approach was used to predict the broadband reflectance in the short-, medium, and long-wavelength portions of spectrum and, indeed, modern versions of Neugebauer sometimes operate using *XYZ* tristimulus values. However, the n-modified spectral Neugebauer approach (illustrated for a CMY system by Equation 11.6) has been shown to be most accurate (Bala, 2003). In the n-modified Neugebauer model all the reflectance factors are raised to the power $1/n$ as in Equation 11.5.

In order to implement the Neugebauer approach the digital counts are first converted into the dot coverage areas using a tone-reproduction curve as described in the previous section. A method to compute the actual areas of the primary and secondary colours is

then required. For the three-colour example, the proportional areas of the eight colour regions can be computed using Demichel's equation (Green, 2002c):

$$\begin{aligned} a_w &= (1-c)(1-m)(1-y) \\ a_c &= c(1-m)(1-y) \\ a_m &= m(1-c)(1-y) \\ a_y &= y(1-c)(1-m) \\ a_b &= cm(1-y) \\ a_g &= cy(1-m) \\ a_r &= my(1-c) \\ a_k &= cmy \end{aligned} \tag{11.7}$$

where c, m and y are the proportional dot areas of the three primary colours obtained from the tone-reproduction curves. Demichel's equation has been shown to work reasonably well for rotated halftone screen configurations where the screens for cyan, magenta and yellow are placed at different angles that are carefully selected to avoid moiré artefacts (Viggiano, 1990).

It is important to note, however, that Equations 11.7 make certain assumptions concerning the amount of overlap between the primary colours. If we consider the case where $c = 0.4, y = 0.4$ and $m = 0$, then Demichel's equation will predict $A_w = 0.36, A_c = 0.24, A_y = 0.24$, and $A_g = 0.16$. However, it would be possible for the cyan and magenta dots to be printed without overlap ($A_w = 0.20, A_c = 0.40, A_y = 0.40$, and $A_g = 0.00$), with total overlap ($A_w = 0.60, A_c = 0.00, A_y = 0.00$, and $A_g = 0.40$) or with any intermediate amount of overlap. The primaries are normally printed at different screen angles and the relationship between these two angles is one of several factors that could affect the degree of overlap. The dot-on-dot half-tone configuration, for example, places the primary dots at the same screen angle and phase so that they maximally overlap. In practice it has been shown that a weighted combination of the Demichel model and the dot-on-dot model can give good performance (Bala, 2003).

The Neugebauer models assume that the reflectance (for spectral Neugebauer approaches) or tristimulus values (for tristimulus Neugebauer approaches) are known for the over-printed area of the secondary colours. So, for example, if yellow dots are over-printed (with partial overlap) with cyan dots we need to know the colour of the over-print area where cyan ink falls on yellow ink. For three- and four-colour printing it is usually possible to obtain the colours of the over-printed secondary and tertiary colours by direct colour measurement since they are relatively few in number.

Of course, the Neugebauer equations predict colour (either trichromatic or spectral) from colorant concentrations of values. In practice the inverse relationship is required and this is normally achieved by an iterative approach (though see Mahy and Delabastita, 1996).

11.2.3 Example Characterisation of a Half-Tone Printer

Figures 11.4 and 11.5 show the reflectance spectra for two inks (a magenta and a yellow respectively) printed at full coverage (solid colour) and at various proportions of cover over white using a half-tone process.

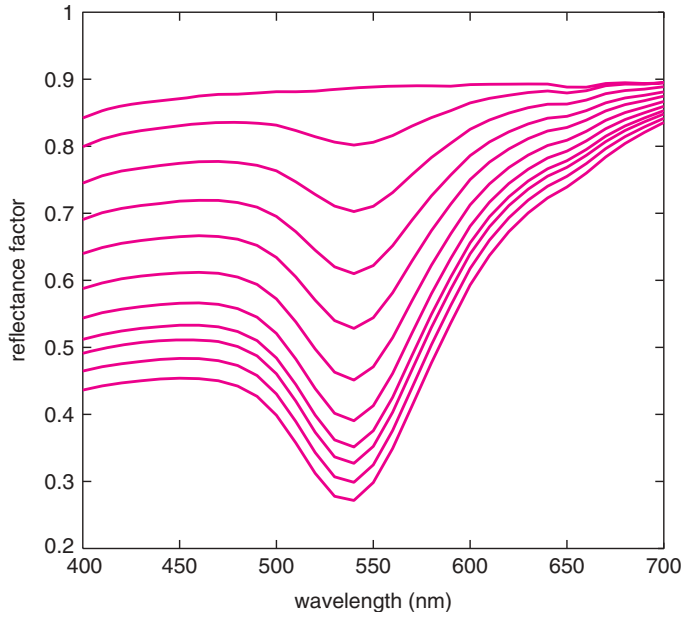

Figure 11.4 *Reflectance factors for magenta ink printed using a half-tone process over white (at coverages of 0, 0.1, 0.2, . . ., 0.9, and 1.0).*

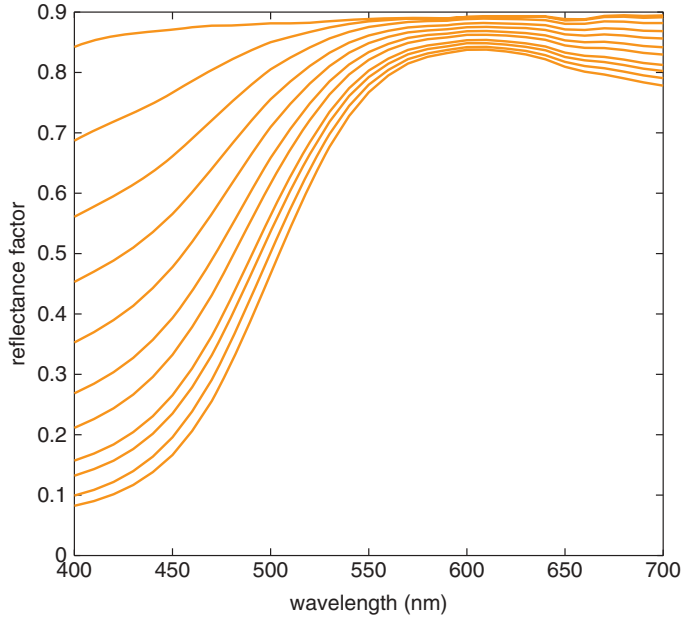

Figure 11.5 *Reflectance factors for yellow ink printed using a half-tone process over white (at coverages of 0, 0.1, 0.2, . . ., 0.9, and 1.0).*

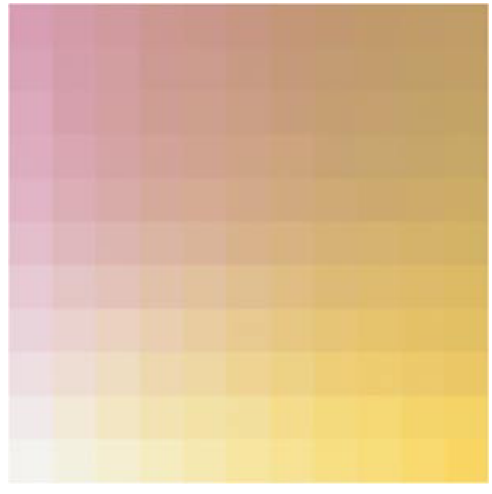

Figure 11.6 *sRGB representation of the printer output. A yellow ramp (coverages 0 to 1 and intervals of 0.1) was printed horizontally from left to right and a magenta ramp was printed vertically from bottom to top. The magenta ramp was printed first followed by the yellow ramp which was over-printed. This provides 121 different combinations of the two inks.*

A total of 121 combinations of magenta ink over-printed with yellow ink were produced (see Figure 11.6). The combinations were produced by printing two colour ramps at right angles to each other. So, in Figure 11.6, the magenta ink coverage was increased from 0 to 1 at intervals of 0.1 from bottom to top; the yellow ink coverage was increased from 0 to 1 at intervals of 0.1 from left to right. The top-right patch is solid magenta ink over-printed with solid yellow ink. The bottom-left grid patch is the white substrate. The reflectance spectra were measured at 31 wavelengths between 400 nm and 700 nm respectively. Figure 11.6 was produced by converting the reflectance factors to CIE *XYZ* values under D65 and then to sRGB values using `xyz2srgb`. A characterisation model will be developed to predict the reflectance factors of each of the 121 patches in the grid that is illustrated in Figure 11.6.

The effect of dot gain means that the first stage is to derive the relationship between the target digital coverage d and the actual dot coverage a (Equation 11.5). The bottom row of patches on the grid provide the data for determining the trc (tone-reproduction curve) for the yellow ink and those along the left-most column provide the data that will allow the trc curve to be determined for the magenta ink. Figure 11.7 and Figure 11.8 show the modelled trc that have been produced by fitting the data with a third-order polynomial. Equation 11.5 requires a value of n to de used. In this study an exceptionally high value ($n = 20$) was found to be optimal[1].

The third-order polynomial was implemented and fitted using MATLAB's® `polyfit` and `polyval` commands. Equations 11.6 and 11.7 are then used to predict reflectance for mixtures of the yellow and magenta inks at various target coverages. Note that the

[1] A value of $n = 20$ is, of course, physically unreasonable. It may suggest that some aspect of the spectral Neugebauer model needs to be modified.

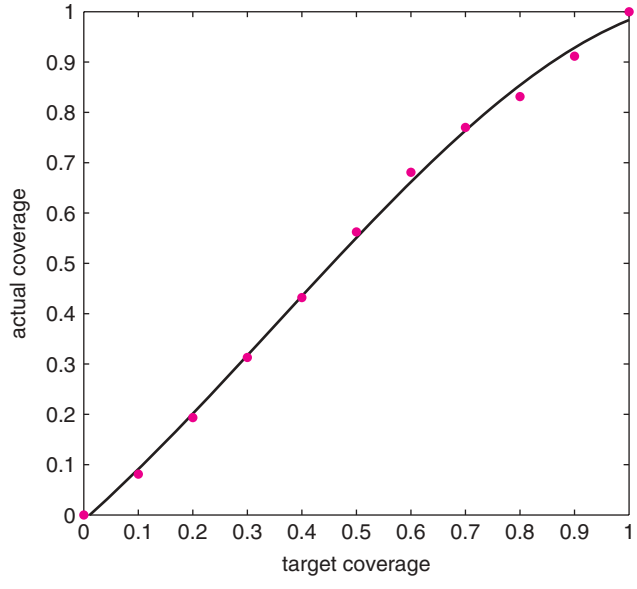

Figure 11.7 *Tone-reproduction curve for magenta ink. The target coverages are plotted on the x-axis and the actual coverages (obtained from Equation 11.5 are plotted on the y-axis. The solid line indicates a fit to the data using a third–order polynomial.*

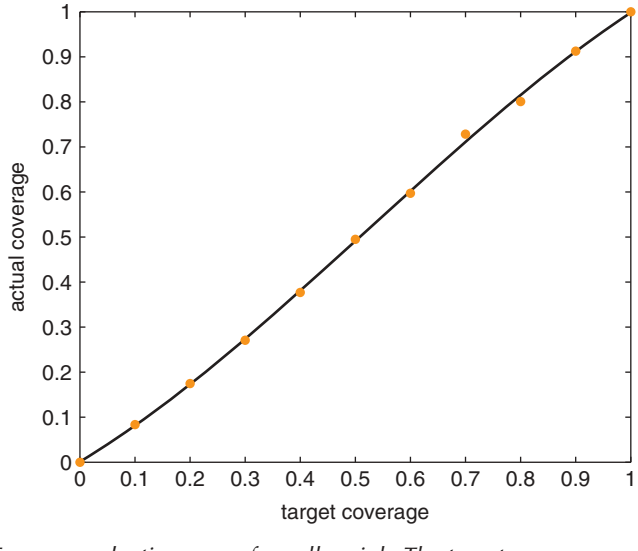

Figure 11.8 *Tone-reproduction curve for yellow ink. The target coverages are plotted on the x-axis and the actual coverages (obtained from Equation 11.5 are plotted on the y-axis. The solid line indicates a fit to the data using a third–order polynomial.*

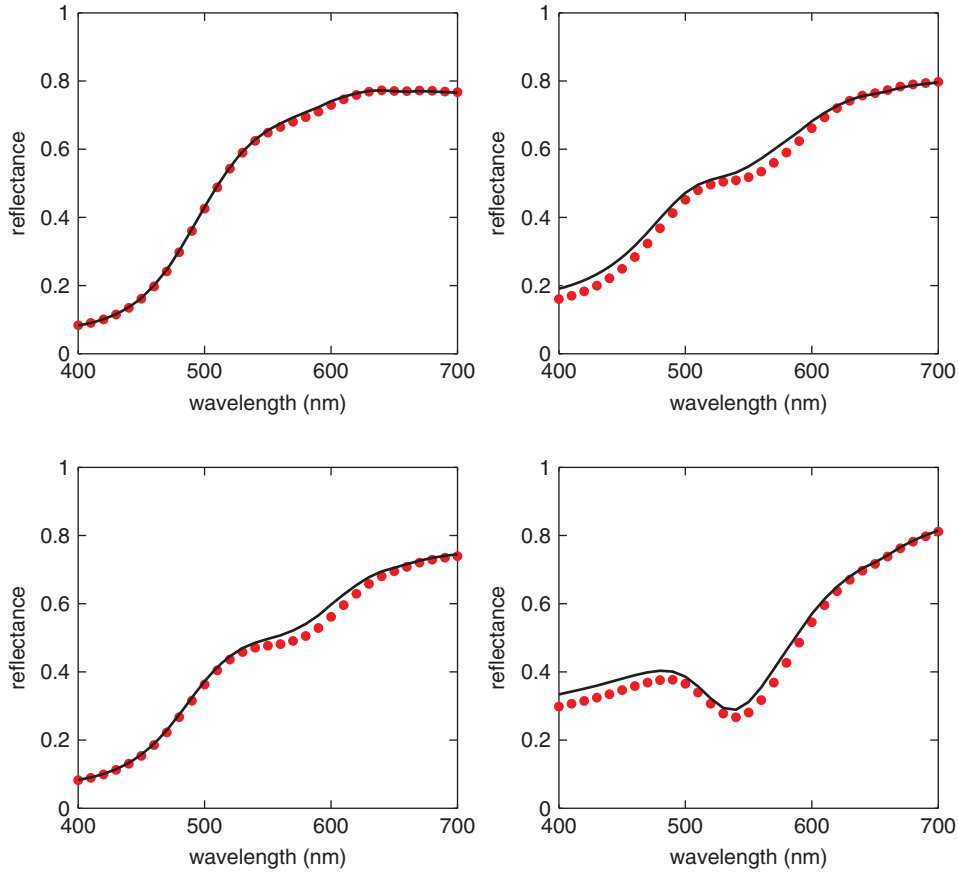

Figure 11.9 *Example predictions (solid lines) to the spectral data (symbols) for four of the 121 patches selected at random.*

overlap colour (where yellow is printed over magenta, in this case) needs to be known and the colour of the white substrate also needs to be known. In Figure 11.6 these correspond to the samples in the top-right and bottom-left positions respectively in the grid. Four examples of the performance of the model are shown in Figure 11.9. The mean CIEDE2000 colour difference between the measured and predicted reflectance spectra of the 121 samples (see Figure 11.6) was 1.56. An sRGB representation of the accuracy of the model is provided by Figure 11.10 which can be visually compared with Figure 11.6.

11.3 Characterisation of Continuous-Tone Printers

11.3.1 Kubelka-Munk Models

When inks are printed on top of each other or are mixed together and then printed, subtractive colour mixing takes place and additivity of reflectance values is not valid.

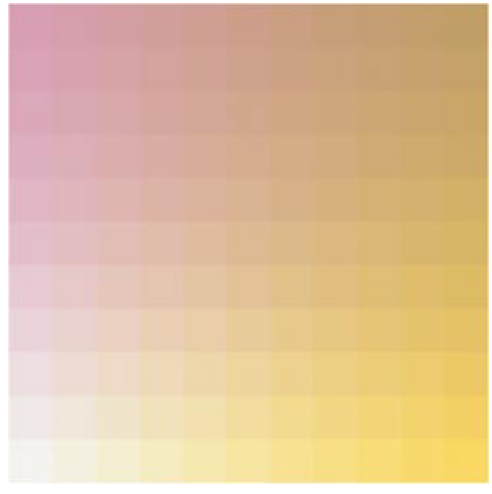

Figure 11.10 *sRGB representation of the predicted printer output (compare with Figure 11.6).*

For subtractive mixing, the densities of the inks are approximately additive where the density D is related to the reflectance P, thus:

$$D = -\log_{10} P. \tag{11.8}$$

So, for example, if two inks have reflectance 0.1 and 0.9 at a certain wavelength and they are mixed together in equal proportions, then the mean of the density contributions will be 0.5229 corresponding to a reflectance of 0.30 (this compares to a value of 0.45 if the reflectance factors are directly averaged). However, accurate prediction for subtractive mixing often requires application of the Kubelka-Munk theory of radiation transfer that characterises each ink or colorant in terms of its absorption and scattering properties.

The Kubelka-Munk theory (Nobbs, 1995; 1997; McDonald, 1997b) has been used to predict the reflectance of inks, plastics, paints, textiles and other materials. The theory characterises each colourant by absorption K and scattering S coefficients that are functions of wavelength and relates these coefficients to the body reflectance of a sample. The body reflectance is the reflectance of a surface if the interactions of light at the air/medium interface are discounted. The body reflectance R is related to the measured reflectance P by the following equation:

$$R(\lambda) = \frac{P(\lambda) - r_e}{(1 - r_e)(1 - r_i) + r_i(P(\lambda) - r_e)} \tag{11.9}$$

for the case where P is measured with a spectrophotometer with the specular component included. The variables r_e and r_i are respectively the external and internal reflectance coefficients of the boundary. The inverse of Equation 11.9 is given by Equation 11.10:

$$P(\lambda) = r_e + \frac{(1 - r_e)(1 - r_i)R(\lambda)}{1 - r_e R(\lambda)}. \tag{11.10}$$

For an opaque sample the body reflectance is related to the K and S coefficients by Equation 11.11:

$$K/S = (1-R)^2/2R \tag{11.11}$$

and the inverse relationship is given by:

$$R = 1 + K/S - ((1 - K/S)^2 - 1)^{0.5} \tag{11.12}$$

Thus, for opaque samples only the ratio of K to S is required at each wavelength in order to predict the reflectance R. In the case of dyed textiles the dyes themselves do not scatter light and the only scattering comes from the textile fibres to which the dyes are applied. The application of Kubelka-Munk theory to opaque dyed textiles is therefore often referred to as the one-constant version of the theory. For many pigmented surface coatings, such as paints, the pigments both absorb and scatter and the two-constant theory is required. However, Equation 11.12 can still be used to predict the reflectance of the surface coating if it is applied at a thickness that achieves opacity. For translucent printing inks, however, the reflectance of the paper upon which the ink is printed makes a contribution to the reflectance of the system and therefore Equation 11.12 must be replaced by:

$$R = \frac{(R_g - R_\infty)/R_\infty - (R_\infty R_g - 1)e^{(1/R_\infty - R_\infty)Sx}}{(R_g - R_\infty)(R_g - 1/R_\infty)e^{(1/R_\infty - R_\infty)Sx}} \tag{11.13}$$

where R_g is the reflectance of the substrate and R_∞ is the reflectance (as defined by Equation 11.12) of an opaque layer of the pigmented layer. The scattering coefficient S is defined for a unit thickness of the layer and x is the thickness of the layer. According to the theory the values of K and S should be linearly related to the pigment volume concentration in the layer and to the thickness of the layer. However, in practice severe departures from linearity can occur (Nobbs, 1997). In order to predict the reflectance for a mixture of colorants or inks the K and S contributions are determined for each component and then assumed to be additive in order to allow the computation of K and S for the layer and thus, via Equation 11.13, the reflectance R. In all cases, once the body reflectance is known Equation 11.10 can be used to yield a prediction of the reflectance P.

The Kubelka-Munk theory is routinely used for the prediction of reflectance for systems of printing inks (for example, in lithography) and forms the basis of computer match-prediction systems used in paint, plastic and textile manufacturing industries. However, one of the difficulties in applying the theory to the characterisation of printers is in determining the values of K and S for the individual inks. One method to determine K and S is to print each colorant over two differently coloured substrates or papers (for example, a white and a black) and then to use Equation 11.13 to set up a system of two simultaneous equations with two variables (K and S). For many printing systems it is difficult to obtain these samples, especially since it is required that the surface properties (roughness etc.) of the two substrates must be identical. An alternative approach is to treat the Kubelka-Munk coefficients as free parameters and to derive their values based on an optimisation routine and a set of samples of known reflectance (Bala, 2003).

The Kubelka-Munk model could be used to predict the overlap areas in a half-tone printing process and Neugebauer-type models could then be used to predict the reflectance of a given area. The traditional Kubelka-Munk theory assumes that the printed layer is homogeneous, however, whereas for half-tone printing one ink is printed on top of another to generate a more layered structure. Nevertheless, applications of modified Kubelka-Munk models to characterisation of printers have been reported in the literature (Kang, 1994; Emmel and Hersch, 2000).

11.3.2 Interpolation of 3D Look-Up Tables

The use of interpolation with 3D look-up tables (LUTs) is commonly used in printer characterisation (Hung, 1993). Kang (2006) notes that this requires three distinct phases: packing, extraction and interpolation. Packing is the process of populating the 3D space with sample points that will constitute the LUT. Extraction is the process by which the closest points in the LUT are found with respect to a search point. Interpolation is where the extracted lattice points are used to calculate the required attributes for the search.

Distance-weighted interpolation is unconstrained by the number of colour samples that constitute the LUT (Shepard, 1968; Balasubramanian and Maltz, 1996) and is therefore appropriate for use in cases where the sample points are irregularly spaced. This approach is based on the principle that vectors near the point of interest should have a greater influence than vectors far away; so a weighting function inversely proportional to distance is used.

The distance-weighted interpolation model finds the n closest known samples to the target sample that is under consideration and computes, for each of these known samples, a weighting factor ω_i:

$$\omega_i = \sigma_i / \sum_i \sigma_j \tag{11.14}$$

and:

$$\sigma_i = 1/(d_i^e + \varepsilon) \tag{11.15}$$

where d_i is the distance of the sample i from the target sample (Figure 11.11). The term e is an exponent and ε is a small noise term so that the value of ω_i is not infinity when d_i tends towards zero.

The distance term can be computed, thus:

$$d_i = \|CMYK_{test} - CMYK_i\| \tag{11.16}$$

Finally, the predicted XYZ values for the target are computed by the weighted sum shown by:

$$\begin{aligned} X &= \sum_{i=1}^{n} \omega_i X_i \\ Y &= \sum_{i=1}^{n} \omega_i Y_i \\ Z &= \sum_{i=1}^{n} \omega_i Z_i \end{aligned} \tag{11.17}$$

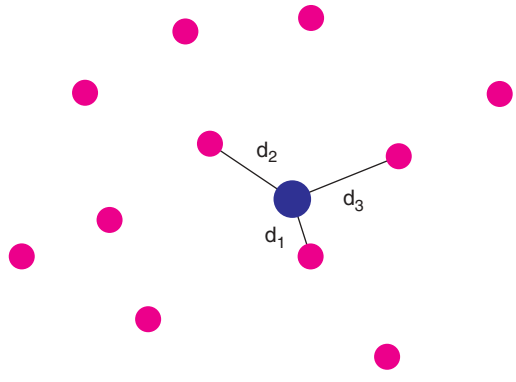

Figure 11.11 *Using distance-weighted interpolation the samples (shown by the magenta symbols) in the look-up table may be irregularly spaced. The n-closest neighbours in RGB space are found for the target colour (shown by the blue symbol).*

11.3.3 General Linear and Nonlinear Transforms

A comparison of neural networks and polynomial transforms to characterise a Kodak Color Proofer 9000A dye-sublimation printer was carried out (Westland and Ripamonti, 2004) based on data provided from Sueeprasan (2003). Neural networks with various numbers of units in the hidden layer were evaluated but optimum generalisation performance was found with 6 units in the hidden layer and the median CIELAB error on the test set was 3.16. For comparison, a third-order polynomial transform achieved a median test error of 4.01 CIELAB units. This was achieved with a training set of 729 samples.

However, a key to the success of any transform or interpolation method is the nature and quality of the data that are collected to optimise the transform or populate the LUT. Morovič *et al*. (2010) show that characterisation performance (in terms of median colour difference error) generally reduces as the number of samples in the training set[2] increases and only asymptotes when the number reaches several thousands. Johnson (1996) suggested the use of 200 colour samples as an absolute minimum but approximately 4000 is more typical. Several attempts have been made to select the training samples wisely so that as few as possible can be used (Mahy, 2000; Monga and Bala, 2008; Tastl *et al*., 2009; Morovič *et al*. 2010).

11.3.4 Example Characterisation of a Half-Tone Printer

CIE *XYZ* data were measured for the patches of an IT8.7-3CMYK characterisation chart (Figure 11.12) printed on a HP Color LaserJet 5500n printer. This chart specifies colour at each pixel in CMYK terms.

The colour patches from the IT8-7-3CMYK characterisation were separated into 746 training patches and 104 test patches. These data are provided in the toolbox as `train.mat` and `test.mat` respectively. The training samples were used to populate

[2] We use 'training set' rather generally here to indicate either the samples that are used to optimise a polynomial transform, train a neural network, or populate a LUT.

Figure 11.12 *IT8.7/3 chart used in the case study.*

the LUT and distance-weighted interpolation was then used to estimate the CIE *XYZ* values of each of the 104 training samples. The function printer_demo1 implements the distance-weighted interpolation algorithm and computes CIELAB colour differences between each of the 104 test samples and their estimated values. The mean CIELAB colour difference was found to be 4.1. Figure 11.13 shows a histogram of the 104 colour differences.

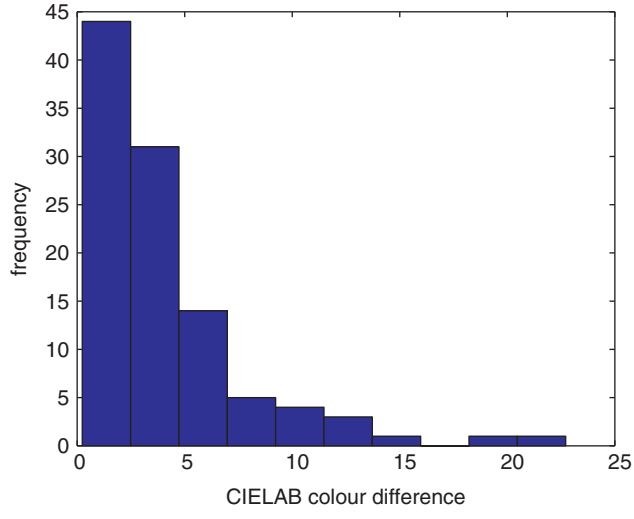

Figure 11.13 *Histogram of CIELAB colour differences for 104 test samples from the IT8.7/3 using distance-weighted interpolation.*

```
% ======================================================
% *** FUNCTION printer_demo1
% ***
% *** function printer_demo1
% *** implements distance-weighted interpolation
% *** requires train.mat and test.mat are in the path
% *** see also printer_demo
% ======================================================
function printer_demo1
e = 0.000001;
p = 4.5;
% load 746 standard xyz and cmyk values
load train.mat
% separate X,Y,Z into one single column
ciexstd = xyz(:,1);
cieystd = xyz(:,2);
ciezstd = xyz(:,3);
% separate X,Y,Z into one single column
cstd = cmyk(:,1);
mstd = cmyk(:,2);
ystd = cmyk(:,3);
kstd = cmyk(:,4);
% load 104 test xyz and cmyk values
load test.mat
ctest = cmyk(:,1);
mtest = cmyk(:,2);
ytest = cmyk(:,3);
```

```
ktest = cmyk(:,4);
% calculate distance between standard and test RGBs
for j=1:104
% compute the errors for the jth samples
for i=1:746
        distance(i) = ((ctest(j)
        cstd(i))^2+(mtest(j)-mstd(i))^2+^(ytest(j)
        ystd(i))^2+(ktest(j)-kstd(i))^2)^(1/2);
         distance(i) = (distance(i))^(p) + e;
end
% now sort the values
    temp = [ciexstd cieystd ciezstd cstd mstd ystd
       kstd distance'];
    temp1 = sortrows(temp,8);
% find the first 10 with smallest distance
temp2 = temp1(1:10,:);
for n=1:10
        cdis(n) = (ctest(j)-temp2(n,4));
        mdis(n) = (mtest(j)-temp2(n,5));
        ydis(n) = (ytest(j)-temp2(n,6));
        kdis(n) = (ktest(j)-temp2(n,7));
end
    temp3 = [temp2 cdis' mdis' ydis' kdis'];
    temp4 = sortrows(temp3,8);
    temp5 = sortrows(temp3,9);
    temp6 = sortrows(temp3,10);
    temp7 = sortrows(temp3,11);

% find the smallest positive and smallest
    % negative values for each plane
    index1=0;
for n=1:9
   if temp4(n,8)*temp4(n+1,8) <= 0
        index1 = n;
        index2 = n+1;
   end
end
if (index1==0)
        t = sprintf('no switch point found in sample
             %d cyan plane', j);
        disp(t)
        index1 = 1;
        index2 = 2;
end
    index3 = 0;
for n=1:9
   if temp5(n,9)*temp5(n+1,9) <= 0
        index3 = n;
        index4 = n+1;
```

```
   end
end
if (index3==0)
        t = sprintf('no switch point found in sample
             %d megenta plane', j);
        disp(t)
        index3 = 1;
        index4 = 2;
end
    index5 = 0;
for n=1:9
   if temp6(n,10)*temp6(n+1,10) <= 0
        index5 = n;
        index6 = n+1;
   end
end
if (index5==0)
        t = sprintf('no switch point found in sample
            %d yellow plane', j);
        disp(t)
        index5 = 1;
        index6 = 2;
end
    index7 = 0;
for n=1:9
if temp7(n,11)*temp7(n+1,11) <= 0
             index7 = n;
             index8 = n+1;
end
end
if (index7==0)
        t = sprintf('no switch point found in sample
             %d black plane', j);
        disp(t)
        index7 = 1;
        index8 = 2;
end
    temp7(1,:) = temp4(index1,:);
    temp7(2,:) = temp4(index2,:);
    temp7(3,:) = temp5(index3,:);
    temp7(4,:) = temp5(index4,:);
    temp7(5,:) = temp6(index5,:);
    temp7(6,:) = temp6(index6,:);
    temp7(7,:) = temp6(index7,:);
    temp7(8,:) = temp6(index8,:);

    sample = 1./temp7(:,8);
    sample = sample/sum(sample);
    temp7(:,8) = sample;
```

```
% find estimated XYZs for test samples
    Xest = temp7(:,1).*temp7(:,8);
    Xest = sum(Xest);
    Yest = temp7(:,2).*temp7(:,8);
    Yest = sum(Yest);
    Zest = temp7(:,3).*temp7(:,8);
    Zest = sum(Zest);
    xyzest(j,:) = [Xest Yest Zest];
end
Xest = xyzest(:,1);
Yest = xyzest(:,2);
Zest = xyzest(:,3);

lab = xyz2lab(xyz,'d65_64');
labest = xyz2lab(xyzest,'d65_64');
de = cielabde(lab,labest);
disp(mean(de))
figure
hist(de)

end
% ================================================
% *** END FUNCTION printer_demo1
% ================================================
```

12
Multispectral Imaging

12.1 Introduction

The characterisation of a digital colour camera so that the device-dependent *RGB* values may be transformed to device-independent coordinates such as *XYZ* values effectively converts the camera into an imaging colorimeter and this has many practical uses. However, in imaging science, there are limitations to this approach. Many applications require that some illuminant-independent measure, such as the spectral reflectance values, be determined at each pixel location in a scene and this may be achieved by using an imaging spectrophotometer. Although imaging spectrophotometers are becoming more commercially available, they are often expensive and there is current interest in exploring to what extent spectral values may be recovered from a standard three-channel camera system or from a camera system with relatively few channels. The term multispectral imaging is sometimes used to define this field of research. This definition is confusing, however, since even the normal *RGB* image representation may be described as being multispectral in some sense. In this chapter, however, the term multispectral imaging will be used to define techniques and methods that may be used to recover spectral information from camera systems with a small number of channels (typically in the range 3–8). We distinguish multispectral imaging from the term hyperspectral imaging which we use to describe techniques where spectral values are measured using imaging devices with a large number of channels (typically in the range 16–40). Clearly there are situations where the distinction between multispectral and hyperspectral may become blurred. Indeed, we note that the goal of both multispectral imaging and hyperspectral imaging is often the same; to recover a spectral image. Readers are directed to Hardeberg (2001) for further information about this research area. We begin this chapter with a brief review of some computational approaches to the problem of colour constancy because

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.

many of the methods of multispectral imaging were, in fact, inspired by a computational analysis of the problem of colour constancy. The problem of whether the visual system might be able to recover the spectral properties of objects in a scene from the cone excitations has been studied extensively and analyses of this problem are relevant for multispectral imaging. We describe some computational procedures for spectral recovery using multispectral imaging and finally describe some applications of these procedures for reflectance recovery and camera characterisation.

12.2 Computational Colour Constancy and Linear Models

Colour constancy is the phenomenon by which surfaces tend to retain their approximate daylight colour appearance when viewed under a wide range of different light sources. It is still a mystery how the visual system is able to discount the effect of the illumination when the colour signal that reaches the eye depends just as much on the spectral power distribution of the illuminant as it does on the spectral reflectance of the surfaces in the scene (Hurlbert, 1991). One possible mechanism that could account for colour constancy is adaptation of the light receptors or cones. Such a possibility was first put forward by von Kries and is consistent with the chromatic-adaptation transforms that were described in Chapter 6. However, adaptation is a relatively slow process and yet colour constancy seems to occur almost instantaneously as we move from one light source to another in our everyday lives. An alternative approach to adaptation was postulated by Land and McCann (1971) who suggested that the visual system may use some computational process to recover signals that are independent of the illumination in a scene. In their computational analysis, known as the Retinex theory, Land and McCann called these signals Lightnesses; biological correlates of reflectance that were computed by each of the three channels in the visual system. Later, the term integrated reflectance was introduced (McCann, McKee and Taylor, 1976) to describe the illuminant-invariant signals. A number of researchers have since investigated to what extent the visual system might actually be able to recover spectral reflectances for points in a scene from the corresponding triplets of cone excitations.

There are serious limitations on what we can achieve when we set out to estimate surface reflectance from cone excitations. For example, theoretically there are an infinite number of combinations of surface-reflectance function P and illuminant power distributions E that could produce a given colour signal S. In addition, the visual system does not measure S directly, but rather encodes it in the absorption rates of the three different cone types. It seems that if P and E were not constrained in some way the cone excitations would provide little useful information; fortunately there are some strong constraints on both P and E.

Suppose we have a device with three colour sensors, whose spectral responsivities are $R_k(\lambda),\ k = 1, 2, 3$. The three sensor responses for a colour signal $S(\lambda)$ will be:

$$\begin{aligned} r_1 &= \sum_{\lambda} R_1(\lambda) S(\lambda) \\ r_2 &= \sum_{\lambda} R_2(\lambda) S(\lambda) \\ r_3 &= \sum_{\lambda} R_3(\lambda) S(\lambda) \end{aligned} \tag{12.1}$$

which we can alternatively represent as a single matrix equation thus:

$$\mathbf{r} = \mathbf{M}\mathbf{s} \tag{12.2}$$

where $\mathbf{r}$ is a 3×1 matrix containing the sensor responses, the rows of the 3×31 matrix $\mathbf{M}$ are the sensor spectral responsivities, and the 31×1 matrix $\mathbf{s}$ is the colour signal (this assumes that the spectral sensitivities of the channels and the spectral power of the colour signal are represented at 31 wavelengths in the visible spectrum). The key question is whether it is possible to compute $\mathbf{s}$, given both $\mathbf{r}$ an $\mathbf{M}$? Recall from Chapter 3 that equations of this form can be solved in MATLAB® using the command:

```
s = M\r;
```

Unfortunately, estimates of $\mathbf{s}$ using this method are likely to be widely inaccurate. In simple terms, Equation 12.2 represents a set of three simultaneous equations with 31 unknowns; in mathematical terms, an underdetermined system.

12.2.1 Example Using MATLAB®

The Keele Natural Spectra (Westland *et al*., 2000) are provided as `keele.mat` with the toolbox. These are 404 reflectance spectra sampled at 31 wavelengths (400 nm–700 nm) and represent the reflectance factors of various natural objects (leaves, petals etc.). They are used in this chapter as an example set of spectra for the demonstrations and demonstrations.

The following MATLAB® code loads the Keele Natural Spectra and computes camera *RGB* values for each for a hypothetical trichromatic camera (the camera spectral sensitivities are included in the toolbox as `camera.mat` and are not intended to represent any real camera) under an equal-energy illuminant. Equation 12.2 is used to calculate camera *RGB* values. Equation 12.2 is then directly inverted to yield estimates of the reflectance spectra from camera responses.

```
w = linspace(400,700,31);
load keele.mat
% spectra is 404 by 31
spectra=spectra/100;
load camera.mat
% r is 3 by 31
% compute camera responses
rgb = r*spectra';
% recover estimates of spectra from RGB
metamers = (r\rgb)';
% metamers is 404 by 31
```

Figure 12.1 shows one of the 404 reflectance spectra and the estimate of the spectrum based on inverting Equation 12.2. Figure 12.1 reveals two interesting aspects concerning

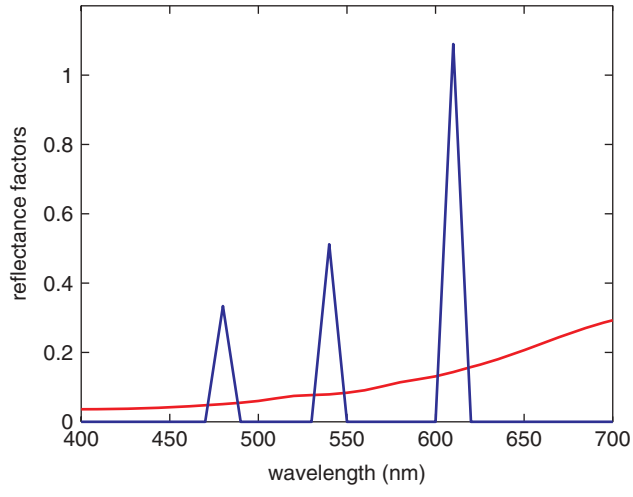

Figure 12.1 *An example spectrum (red line) from the set of Keele Natural Spectra and a recovered spectrum (metamer) obtained by inverting Equation 10.2.*

Equation 12.2. The first is that there are an infinite possible number of spectra that could correspond to any *RGB* triplet. When Equation 12.2 is inverted it is extremely unlikely that the recovered spectra will be the same as the original spectra; instead, a metamer will be recovered unless additional constraints are imposed. The second point is that there is nothing to constrain the recovered spectra to be within the bounds of 0 and 1. The example shown in Figure 12.1 demonstrates both that the recovered spectrum is a camera metamer of the original spectrum and is not constrained within the range 0–1.

12.3 Properties of Reflectance Spectra

Figure 12.2 shows a set of five typical reflectance spectra from the set of 404 Keele Natural Spectra (Westland *et al*., 2000). It is clear that they are generally smooth functions of wavelength. In fact, the spectra illustrated by Figure 12.2 were measured for surfaces of natural objects (leaves, petals, etc.) but the reflectance of the surface of the output of a CMYK printer or a painted sample would appear similarly smooth. This is because the smoothness originates from fundamental mechanisms by which matter interacts with light (Maloney, 1986). Reflectance spectra vary only slowly with wavelength. (There are also known constraints (Judd, MacAdam and Wyszecki, 1964) on the variability of natural daylight.)

An alternative way to represent the constraints of surface reflectance spectra is to state that their Fourier representations are band limited. That is, if the Fourier amplitude spectrum is computed for a reflectance spectrum the energy quickly falls off with increasing spectral frequency (spectral frequency is typically expressed in units of cycles per nanometre). Above the band limit there is zero energy. Estimates of the band limit for natural and man-made surfaces are in the region 0.15–0.20 cyc/nm (Maloney, 1986; Westland *et al*., 2000).

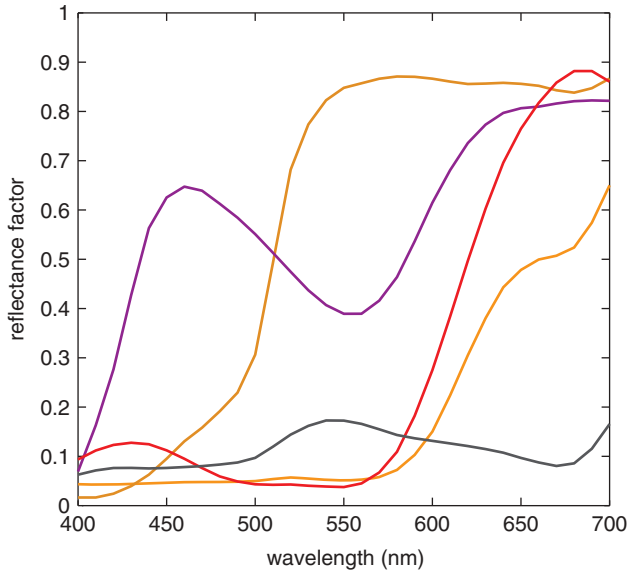

Figure 12.2 *Five selected reflectance spectra from Keele Natural Spectra (Westland et al., 2000). The spectra change only slowly with respect to wavelength.*

This leads to questions about the inherent dimensionality of reflectance spectra. Although spectra that are measured at 31 wavelength intervals can be represented in a 31-D space, it may be that they could be represented in a space of lower dimensionality without loss of information. One way that this may be explored is by representing spectra by linear models (basis functions) of low dimensionality. There are several ways to obtain appropriate sets of basis functions including Singular Value Decomposition (SVD), Principal Component Analysis (PCA), Characteristic Vector Analysis (CVA) and Independent Component Analysis (ICA).

12.3.1 PCA and SVD

PCA is a mathematical procedure that uses an orthogonal transform to convert a set of observations of possibly correlated variables into a set of values of uncorrelated variables known as principal components (Jolliffe, 1986). The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has as high a variance as possible (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to (uncorrelated with) the preceding components (see Anton, 1994). PCA can be achieved by computing the eigenvectors of the covariance matrix of a data set or by singular decomposition of the data matrix itself. (Strictly speaking, for PCA, the data should be mean centred: that is, for reflectance spectra we should calculate the mean reflectance factor at each wavelength and subtract this from the reflectance factors at each wavelength. If the data are not mean centred then the technique is called CVA.)

Consider, for example, an $n \times w$ matrix $\mathbf{P}$ that contains n spectra each sampled at w wavelengths. The matrix $\mathbf{P}$ can be decomposed using SVD (Pratt, 1978) thus:

$$\mathbf{P} = \mathbf{UVW}^{\mathbf{T}} \tag{12.3}$$

where $\mathbf{U}$ and $\mathbf{V}$ are $n \times w$ and $w \times w$ matrices respectively[1]. The matrix $\mathbf{W}$ is a $w \times w$ matrix whose elements are only nonzero on the diagonal and are called the singular values of $\mathbf{P}$. The columns of $\mathbf{U}$ are called the left singular values; they are eigenvectors of $\mathbf{PP}^{\mathbf{T}}$. The row of $\mathbf{V}$ are called the right singular values; they are the eigenvectors of the covariance matrix $\mathbf{P}^{\mathbf{T}}\mathbf{P}$ and form an orthonormal basis for the reflectance spectra.

Computer code (in the C programming language) to perform singular value decomposition of a matrix is readily available (e.g. Press *et al*., 1993). MATLAB® provides the commands `svd` and `svds` which can be used to get the SVD of a matrix. If we use a linear model to represent a set of reflectance spectra then a given sample in the set is given by the linear sum of the basis functions weighted by coefficients so that:

$$P(\lambda) = a_1 B_1(\lambda) + a_2 B_2(\lambda) + a_3 B_3(\lambda) + \ldots + a_k B_k(\lambda) \tag{12.4}$$

and, if all k basis functions are used, all the spectra in the set can be reconstructed perfectly using appropriate values of the weights $a_1 \ldots a_n$. However, the benefit of techniques such as SVD is that it is possible to represent data efficiently by only using a small number of the basis functions. The first basis function maximally represents the variance in the data, and subsequent basis functions maximally represent the remaining variance. Cohen (1964) showed that 99.2% of the variance in a set of 150 Munsell colour samples could be represented by a linear model with three parameters. More recently it was shown that a linear model with three basis functions could account for more than 99% of the variance in 462 spectra from the Nickerson-Munsell set (Maloney, 1986). Figure 12.3 shows the first three basis functions computed for the 404 Keele Natural Spectra (Westland *et al*., 2000). The figure shows the approximation of one of the spectra by one, two and three basis functions. When three basis functions are used the approximation is quite a good fit to the measured values.

It is not trivial, however, to ascertain how many basis functions are required for an accurate representation without reference to the purpose of the representation. Certainly as colour scientists it is likely that we would be as interested in the colour difference between spectra and their representations in linear models of low dimensionality as in the proportion of variance accounted for by those model. Owens (2002) compared the Keele Natural Spectra with a set of Munsell reflectance spectra. Figure 12.4(a) shows how the mean-squared error for the two sets monotonically decreases with the number of basis functions if a set of basis functions derived from the Keele data is used to represent the Keele data and a set of basis functions from the Munsell data are used to represent the Munsell data. In Figure 12.4(b), however, it can be seen that at least six basis functions are required if average CIELAB ΔE values of about 1 are required. Maloney (1986) also concluded that 6–12 basis functions were necessary for accurate representation and García-Beltrán *et al*. (1998) analysed linear models of the spectra of 5574 samples of

[1] The superscript $^{\mathrm{T}}$ denotes the transpose of a matrix.

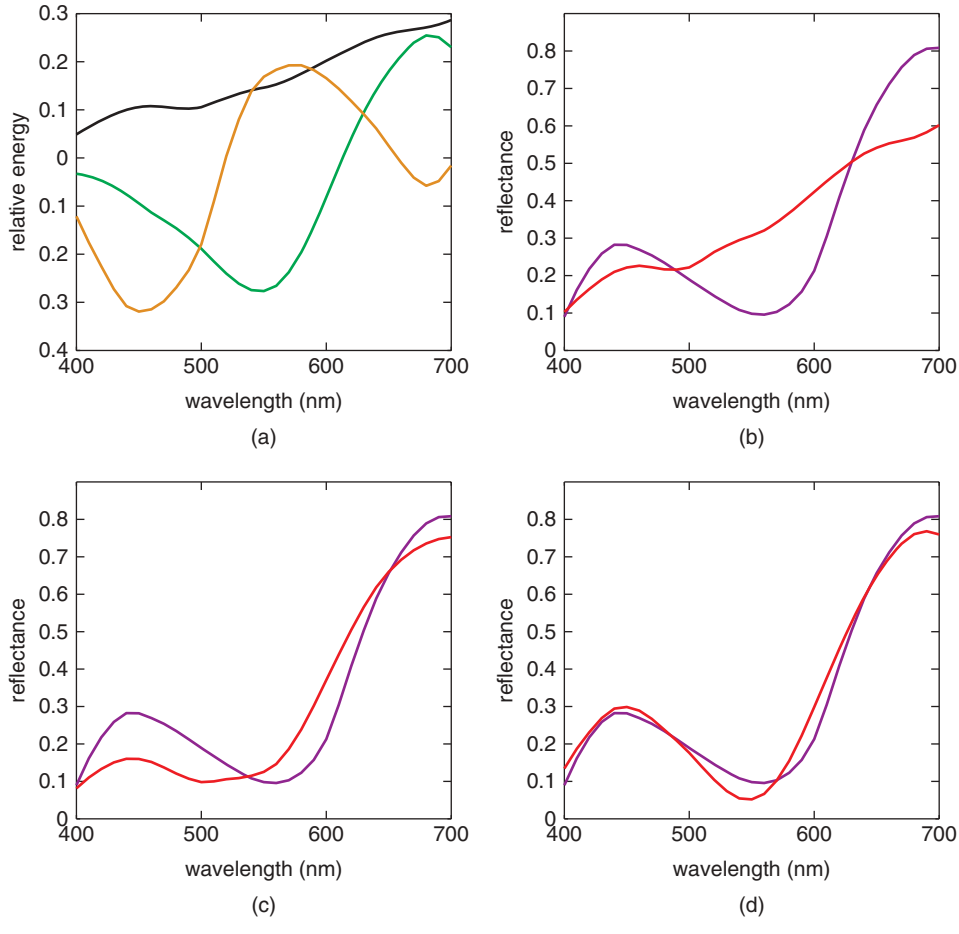

Figure 12.3 *PCA analysis for Keele Natural Spectra showing (a) the first three basis functions and spectral reconstructions for one of the spectra using (b) one, (c) two, and (d) three basis functions.*

acrylic paint and concluded that seven basis functions were required. Tzeng and Berns (2005) used PCA to show that five basis functions were required to represent a set of 56 paint samples with a mean error of less than 1 ΔE^*_{94} unit. It is clear, however, that for many purposes a linear model of reflectance with only three basis functions would be inadequate.

12.3.2 SVD Using MATLAB®

The principal components can be computed using the functions `svd` and `svds` in MATLAB®[2]. For an $n \times w$ matrix $\mathbf{P}$ that contains n spectra each sampled at

[2] The statistics toolbox for MATLAB® provides some additional functions for carrying out PCA.

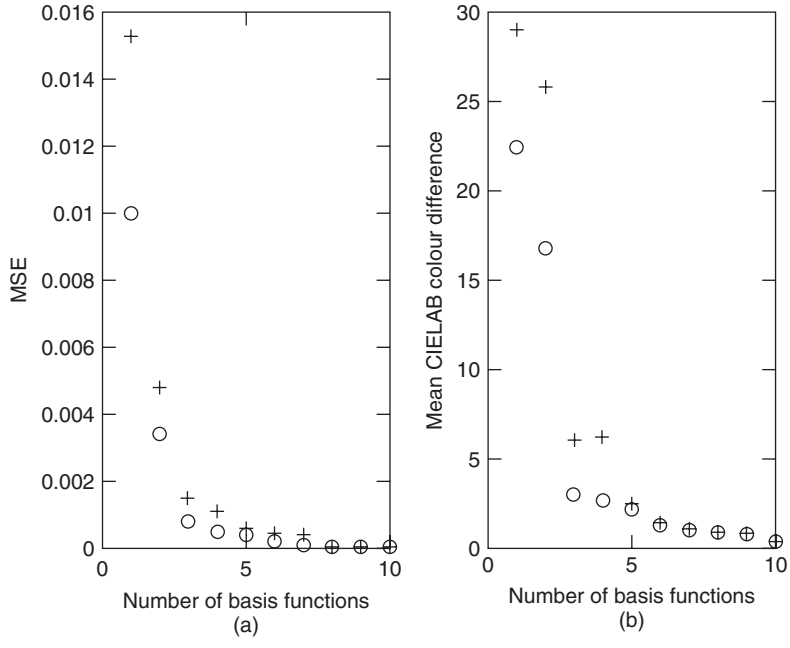

Figure 12.4 *(a) Mean-square error (MSE) as a function on number of basis functions for Keele (circle symbols) and Munsell (plus symbols) data; (b) Mean CIELAB colour difference as a function on number of basis functions for Keele (circle symbols) and Munsell (plus symbols) data.*

w wavelengths the following code segment computes the principal components by calculating the eigenvectors of the mean-centred covariance matrix of **P**.

```
% P is n by w
% subtract the mean
diff = P-ones(size(P,1),1)*mean(P);
% compute the covariance matrix
cova = (diff'*diff)/(w-1);
% find the eigenvectors and eigenvalues
[pc,e]=eig(cova);
e = diag(e);
% sort the variance in decreasing order
[junk, rindices]=sort(-1*e);
pc = pc(:,rindices);
% the columns of pc contain the eigenvectors
```

An alternative way to compute the principle components is to use the svd (or svds) command directly on the matrix **P**. Thus:

```
% P is n by w
% subtract the mean
diff = P-ones(size(P,1),1)*mean(P);
% normalise
diff = diff/sqrt(31-1);
% just call svd
[u,s,v] = svd(diff);
% the columns of v contain the eigenvectors
```

It is also possible to call `svd` using the data directly (that is, without subtracting the mean first) although slightly different eigenvectors normally result in this case.

The following code computes basis functions using svd, projects the spectra into the space of the first three basis functions and then computes spectral representations in the 3D model.

```
% P is n by w
% perform svd
[u,s,v]=svd(P);
% only keep the first three
v = v(:,1:3);
% project the spectra into the 3-d space
weights = v'*spectra';
% calculate the spectral representations
recon = weights'*v';
```

Note that `v'*spectra'` can be used instead of `pinv(v)*spectra'` because the matrix **v** is orthonormal.

The function `svd_demo` has been prepared to demonstrate the use of SVD to generate and use a set of orthonormal basis functions. The function is listed following:

```
% ====================================================
% *** FUNCTION svd_demo
% ***
% *** function svd_demo
% *** performs svd on set of reflectance spectra
% *** using original and centred data
% *** plots the first three basis functions
% *** requires keele.mat is in the path
% ====================================================
function svd_demo
numbas = 3; % the number of basis functions required
load keele.mat
% the keele data are in the 0-100 format
```

```
spectra = spectra/100;
w = linspace(400,700,31);

% perform svd
[u1,s1,v1] = svd(spectra);
% only keep the first numbas
v1 = v1(:,1:numbas);
plot(w,v1);
title('1st three basis function without centering')
xlabel('wavelength')

% perform svd on centred data
m = mean(spectra);
diff = spectra-ones(size(spectra,1),1)*m;
[u2,s2,v2] = svd(diff);
% only keep the first numbas
v2 = v2(:,1:numbas);
figure
plot(w,v2);
title('1st three basis function with centering')
xlabel('wavelength')

% find the weights
x1 = v1'*spectra';
x2 = v2'*diff';

% reconstruct the spectra from the weights
recon = x1'*v1';
recon1 = x2'*v2'+ones(size(spectra,1),1)*m;

i = round(rand*404);
figure
plot(w,recon(i,:),'b-',w,recon1(i,:),'r-',w,spectra(i,:),'ko')
axis([400 700 0 1])
xlabel('wavelength');
ylabel('reflectance factor');

% compute colour differences
xyz = r2xyz(spectra,400,700,'d65_64');
xyz1 = r2xyz(recon,400,700,'d65_64');
xyz2 = r2xyz(recon1,400,700,'d65_64');
lab = xyz2lab(xyz,'d65_64');
lab1 = xyz2lab(xyz1,'d65_64');
lab2 = xyz2lab(xyz2,'d65_64');
de1 = cielabde(lab,lab1);
de2 = cielabde(lab,lab2);

disp([median(de1) median(de2)]);
end
```

Table 12.1 *Mean CIELAB colour difference for Keele Natural Spectra using linear models derived from SVD.*

Number of basis functions	Mean CIELAB error using SVD of data (PCA)	Mean CIELAB error using SVD of mean-centred data (CVA)
1	28.28	26.69
2	18.62	17.17
3	4.36	5.08
4	2.20	1.82
5	1.91	1.85
6	0.89	0.81

The function `svd_demo` illustrates how to compute the basis functions, how to project spectra onto a low-dimensional space, and how to reconstruct low-dimensional-representations of the spectra. The demo performs SVD on the spectra directly and also on the spectra that are obtained by subtracting the mean spectrum.

Table 12.1 shows the colour difference between spectra and their representations in linear models with increasing numbers of basis functions. The data in Table 12.1 show that in most (but not all) cases the colour difference error is smaller if SVD is carried out on the spectra where the mean has been removed (that is, slightly better results are obtained with PCA than with CVA). However, the differences are generally small and probably insignificant.

12.4 Application of SVD to Reflectance Recovery

Most algorithms for reflectance estimation rely on two essential components (Brill, 1979). First, we need a method of representing our knowledge about the likely surface and illuminant functions (for example, linear models). Second, most modern estimation methods assume that the illumination varies either slowly or not at all across the image. This assumption is important because it means that the illumination adds very few extra parameters that need to be estimated.

To simplify the problem considered in Equation 12.2, imagine that the surface is viewed under an equal-energy light source so that $E(\lambda) = 1$ for all λ. We can now write:

$$\mathbf{r} = \mathbf{M}\mathbf{p} \tag{12.5}$$

where the 31×1 matrix $\mathbf{p}$ represents the spectral reflectance factors of the surface. Although it may still appear that three sensor responses are insufficient to estimate $\mathbf{p}$ we additionally know there are strong constraints on the variability of the spectral reflectances of surfaces. This allows sets of reflectance spectra to be represented by linear models of low dimensionality. We can write:

$$\mathbf{P} = \mathbf{B}\mathbf{a} \tag{12.6}$$

where $\mathbf{P}$ $(31 \times n)$ is a set of n reflectance spectra, the rows of $\mathbf{B}$ $(31 \times k)$ hold k basis functions, and $\mathbf{a}$ $(k \times n)$ represents the weights. The basis functions are themselves

functions of wavelength but are not constrained to be between the range [0 1] nor even to be positive at all wavelengths. The number of basis functions k is usually quite small and the weights for each reflectance spectrum define a projection of the reflectance spectrum onto the k-dimensional space of the basis functions. Such linear models can lead to efficient simple estimation algorithms for **P** given the three sensor responses[3] **r**.

We can therefore rewrite Equation 12.2 as:

$$\mathbf{r} = \mathbf{MBa} \tag{12.7}$$

If we group together the term **MB** (multiplying a 3×31 matrix by a 31×3 matrix in the case where we have three basis functions) we can see that this is a 3×3 matrix whose entries are all known. The only unknown is **a**, the weights. We can therefore calculate **a** using standard MATLAB® commands:

```
a = (M*B)\r;
```

Once **a** has been determined the reflectance spectra can be recovered using Equation 12.6. This analysis illustrates two aspects of the role of linear models. Firstly, linear models represent a priori knowledge about the likely set of inputs. Linear models may be used to allow spectral information to be recovered from three sensor responses. Secondly, linear models work smoothly with the imaging equations. Since the imaging equations are linear, the estimation methods remain linear and simple.

Of course, this method might be expected to lead to relatively poor estimates of **P** since a linear model with just three basis functions represents the spectra **P** to a limited degree of accuracy. There are various modifications to this basic idea that can allow greater accuracy. Many of these have been inspired by models that explore whether the human visual system could recover reflectance spectra from the three classes of cone responses.

Consider an image with s distinct spatial positions. We expect to obtain three cone excitations at each position in the image so the number of measurements is $3s$ in total. If we can use a three-dimensional model for the surfaces, then there are a total of $3s$ unknown surface coefficients. If the illuminant is known, the problem is easy to solve since we have as many measurements ($3s$) as unknowns ($3s$). If the illuminant is not known and can vary from point to point there will be $6s$ unknown parameters (at each point, three parameters for the surface and three parameters for the illuminant) and the problem cannot be solved. If the illuminant is constant across the image we have only three additional parameters (thus $3s + 3$ unknowns and $3s$ measurements) and by making some modest assumptions we can proceed with the estimation algorithm.

Modern estimation algorithms work by finding a method to overcome the mismatch between the number of measurements and unknowns. The majority of algorithms infer the

[3] Note that the sensor responses could be replaced by the responses of the cones in the human visual system.

illumination parameters by making one additional assumption about the image contents. For example, if we know the reflectance function of just one object in the scene we can use the sensor responses from that object to estimate the illuminant. This is often implemented in terms of the assumption that the average of all of the surfaces in the image is grey, the so-called grey-world assumption (Land, 1986; Wandell, 1995). Other algorithms are based on the assumption that the brightest surface in the image is a uniform perfect reflector (Wandell, 1995). Another interesting idea is that we can identify specularities in the image from glossy surfaces (D'Zmura and Lennie, 1986; Tominaga and Wandell, 1990).

A second group of estimation algorithms compensate for the mismatch in measurements and parameters by suggesting ways to acquire more data. Maloney and Wandell (1986) showed that by adding a fourth sensor one can estimate the surface and illuminant. Similarly, D'Zmura and Iverson (1993a; 1993b) explored the possibility of observing the same surface under different illuminants. However, even if the illuminant is known the number of unknowns may be greater than $3s$ if it is assumed that a linear model with greater than three dimensions is required to represent the reflectance spectra. Multispectral imaging is a technique that uses more than three channels so that sufficient information about the scene is captured to allow spectral recovery to an accuracy greater than that which would be possible using a three-dimensional linear model.

12.5 Techniques for Multispectral Imaging

In this section we consider some typical techniques to allow reflectance recovery using multispectral imaging.

12.5.1 Maloney-Wandell Method

The method proposed by Maloney and Wandell (1986) assumes a linear camera model (Equation 12.1) and has also been implemented by Hardeberg (1999). For a single surface, the Maloney-Wandell method is based upon Equation 12.8, so that:

$$\mathbf{r} = \mathbf{M}\mathbf{a} \tag{12.8}$$

where $\mathbf{r}$ is an $r \times 1$ matrix of sensor responses, $\mathbf{M}$ is an $r \times k$ system matrix and $\mathbf{a}$ is an $k \times 1$ column matrix of weights that defines the surface in the space of k basis functions.

If we consider the case where $r = k = 3$, then the 3×3 matrix $\mathbf{M}$ would be obtained from the product of the 3×31 matrix of the sensor spectral sensitivities (weighted by the illuminant power distribution) and the 31×3 matrix of the basis functions of the linear model. The entries of $\mathbf{M}$ are thus known (Equation 12.9).

$$\mathbf{M} = \begin{bmatrix} \sum_{\lambda} R_1(\lambda)B_1(\lambda) & \sum_{\lambda} R_1(\lambda)B_2(\lambda) & \sum_{\lambda} R_1(\lambda)B_3(\lambda) \\ \sum_{\lambda} R_2(\lambda)B_1(\lambda) & \sum_{\lambda} R_2(\lambda)B_2(\lambda) & \sum_{\lambda} R_2(\lambda)B_3(\lambda) \\ \sum_{\lambda} R_3(\lambda)B_1(\lambda) & \sum_{\lambda} R_3(\lambda)B_2(\lambda) & \sum_{\lambda} R_3(\lambda)B_3(\lambda) \end{bmatrix} \tag{12.9}$$

The reflectance of the surface may be recovered using the MATLAB® command:

```
a = M\r;
```

However, since it is known (Maloney, 1986; Owens, 2002) that a linear model of at least six basis functions is required for accurate representation of reflectance spectra, **r** must be at least size 6×1. Most practical multispectral imaging systems therefore consist of at least six separate channels and this may be achieved by a filter wheel containing a number of different filters and a monochrome camera system. An alternative procedure to using more sensor classes is to use more than one light source. For example, if an image is taken using a trichromatic camera using one light source and then the same image is taken using a second light source then it allows the construction of a matrix **M** with six rows and consequently allows a linear model of reflectance spectra with six basis functions. It is important, however, that the spectral power distributions of the two or more light sources are as unrelated as possible and that they contain power throughout the whole visible spectrum (Connah *et al*., 2001).

Hardeberg (1999) also considered the situation where $r < k$ so that the number of sensor classes exceeds the number of basis functions in the linear model of reflectance. Such an over-determined system can actually lead to improved estimates when compared with the case $r = n$. There are two reasons why an over-determined system may be useful. Firstly, for a system based upon r sensors the r rows of **M** may not be independent. This can happen if the spectral sensitivities of the channels are correlated with each other (or, for a system using two light sources, if the spectral power distributions of the light sources are correlated). Secondly, estimates of **a** when $r = k$ may suffer if the system is noisy so that the matrix **r** is known with low precision.

12.5.2 Imai-Berns Method

Imai and Berns (1999) developed a method for reflectance recovery based directly upon Equations 12.8. They assume a linear relationship between the sensor outputs **r** of the imaging system and the representation of the surface in an r-dimensional basis space by the weights **a**. However, unlike the Maloney-Wandell method, Imai and Berns find the entries of **M** by an empirical least-squares analysis. The method is simple and effective because for the Maloney-Wandell method it is necessary to determine the space of basis functions in which the reflectance spectra will be represented, to measure the spectral power distribution of the light source, and to determine the spectral sensitivities of the imaging system. The method proposed by Imai and Berns, however, requires only the first of these steps, the determination of the basis functions, and the entries of **M** are then found by optimisation. Cheung *et al*. (2005) compared the methods of Imai-Berns and Maloney-Wandell using a images of the Macbeth ColorChecker DC (for training) and the Macbeth ColorChecker and samples of the NCS system (for testing) captured under D65 using an Agfa StudioCam device. The best colorimetric performance of the Imai-Berns method was 3.58 CIELAB units (median) whereas the respective figure for the Maloney-Wandell method was 3.64.

12.5.3 Shi-Healey Method

Assuming a trichromatic camera used under one light source the Maloney-Wandell and Imai-Berns methods assume a linear model of reflectance of low dimensionality ($k \leq 3$). Shi and Healey (2002) replace Equation 12.6 with the following relation:

$$\mathbf{P} = \mathbf{B_A a_A} + \mathbf{B_B a_B} \tag{12.10}$$

where the first three basis functions (denoted as $\mathbf{B_A a_A}$) are constrained by the responses of the camera but additional basis functions ($\mathbf{B_B a_B}$) are used to further constrain the problem.

12.5.4 Methods Based on Maximum Smoothness

One problem with methods for reflectance recovery that use basis functions is that the recovered reflectance cannot be guaranteed to be within the range [0, 1]. The representations of spectra in low-dimensional models are not always consistent with physically reasonable spectra. An alternative approach to reflectance recovery is to replace the constraint imposed by the linear model of basis functions with some other constraint. One possibility is to employ a constraint of maximum smoothness (Li and Luo, 2001; Cheung *et al*., 2005).

12.5.5 Device Characterisation Revisited

Chapter 10 discussed method for characterising digital cameras using linear and nonlinear transforms. An alternative possibility to recover CIE *XYZ* values from camera *RGB* responses is to try to recover the spectral data from *RGB* (using the techniques discussed in this chapter) since once the spectral reflectance factors are determined the calculation of CIE *XYZ* values is trivial (Shi and Healey, 2002). Cheung *et al*. (2005) investigated methods of reflectance recovery from the responses of trichromatic cameras and applied these methods to trichromatic characterisation. Four spectral recovery methods were tested; Maloney-Wandell, Imai-Berns, Shi-Healey, and Li-Luo. The best colorimetric performance was obtained from the Li-Luo method. However, none of the methods could out-perform colorimetric performance of a standard polynomial transform.

12.6 Fourier Operations on Reflectance Spectra

The Fourier properties of reflectance spectra may be computed using the MATLAB® function `fft`. For a more complete description of Fourier Analysis of discrete signals using MATLAB® readers are directed towards the text by Carlson (1998). The `fft` command decomposes a signal into its frequency and phase components. Equation 12.11 defines a hypothetical reflectance function P that is a function of wavelength λ.

$$P(\lambda) = b + A\cos(2\pi f\lambda + \phi) \tag{12.11}$$

The signal P may be represented entirely by the value of its offset b, its amplitude A, its frequency f, and its phase ϕ. The frequency may be represented in terms of cycles per

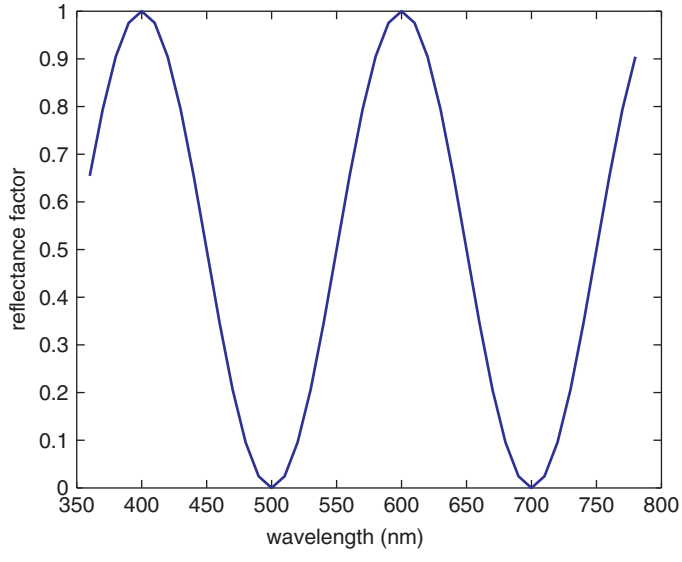

Figure 12.5 *A hypothetical reflectance spectrum with a spectral frequency of 0.005 cyc/nm.*

nanometre. So, for example, if $b = 0.5$, $A = 0.5$, $f = 0.005$ cyc/nm and $\phi = 0$, then we would obtained the signal shown (between the wavelengths 360 nm and 780 nm) by Figure 12.5.

The spectrum shown in Figure 12.6 consists of a single spectral frequency of 0.005 cyc/nm or $0.005 \times 300 = 1.5$ cycles in the visible spectrum (400–700 nm). Fourier analysis involves taking a signal (such as a reflectance curve) and decomposing it into an amplitude spectrum and a phase spectrum. The amplitude spectrum provides information about the spectral frequencies that are present in the reflectance data and the phase spectrum provides information about the phases of these components. If the data are bandlimited, then there will be a limiting spectral frequency (known as the band limit) above which there is no further energy. The amplitude and frequency information can be obtained using the following two MATLAB® commands:

```
four_amp = abs(fft(p));
four_phase = angle(fft(p));
```

where **p** is a $1 \times w$ row matrix and w is the number of wavelength intervals at which the signal is represented. If **p** is a 1×31 row matrix (representing reflectance data at 10 nm intervals in the range 400–700 nm) then **four_amp** and **four_phase** will also be 1×31 row matrices. Figure 12.6 shows a typical reflectance spectrum (Figure 12.6a) and the 31-dimensional vector **four_amp** that results. The first 16 values are the amplitudes at evenly spaced spectral frequencies between zero (also known as the DC) and the Nyquist limit (see Carlson, 1998). The Nyquist limit is half the sampling rate and since

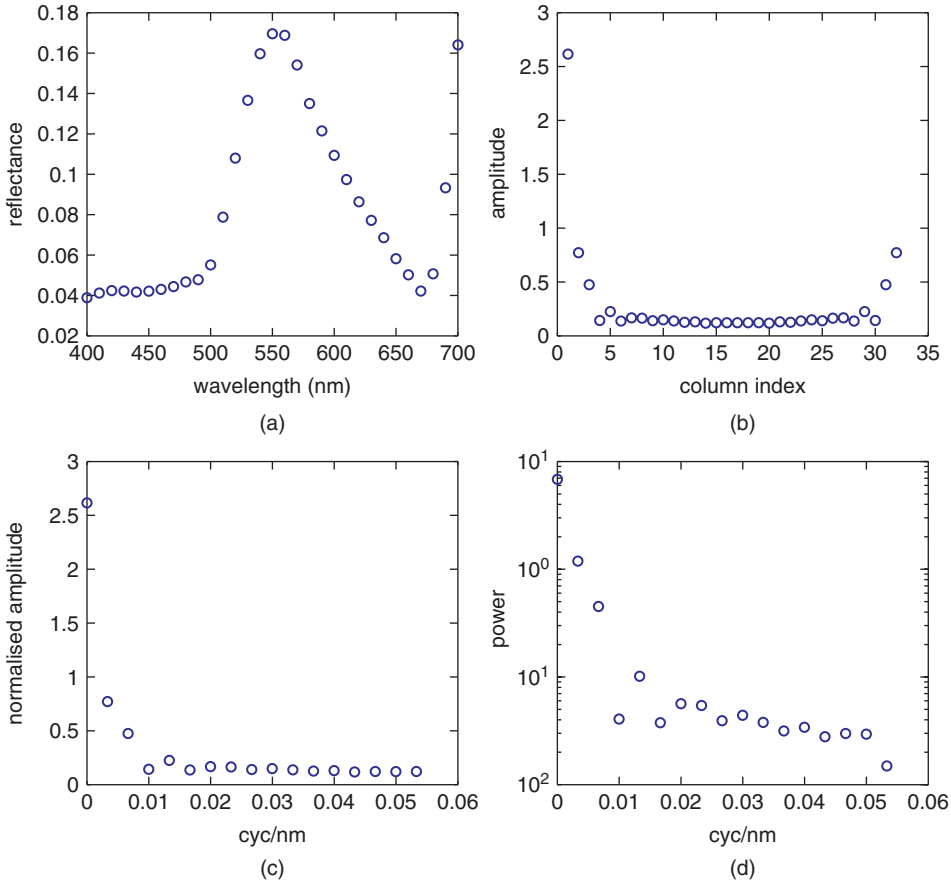

Figure 12.6 *A hypothetical reflectance spectrum with a spectral frequency of 0.005 cyc/nm. Fourier decomposition of a reflectance spectrum showing (a) original reflectance data at intervals of 10 nm, (b) the vector that results from the function four_amp, (c) the normalised amplitude spectrum and (d) the power spectrum.*

the sampling rate is 31, the Nyquist limit is 31/2 cycles per 300 nm or 0.0517 cyc/nm. The subsequent 15 values are the amplitudes for the negative frequencies; these are a mirror-image of those for the positive frequencies and are thus often discarded to create the plots shown in Figure 12.6(c) and also Figure 12.6(d).

Figure 12.6(c) shows that the amplitude generally falls off as the spectral frequency increases indicating that the reflectance data are band limited. Apart from the DC component, Figure 12.6(c) also shows that the frequency with the greatest amplitude is 0.0034 cyc/nm, which corresponds to about a single complete cycle in the range 400–700 nm.

The Fourier representation is useful for analysing properties of reflectance spectra and also for performing various operations such as smoothing (convolution in the wavelength

domain may be achieved by a multiplication in the Fourier domain). Following operations upon the matrices **four_amp** and **four_phase**, the corresponding data in the reflectance domain may be obtained using the commands:

```
four_x = four_amp.*cos(four_phase);
four_y = four_amp.*sin(four_phase);
rec_p = real(ifft(complex(four_x,four_y)));
```

Appendix A

Table of White Points of Illuminants used in r2xyz and Other Functions

	1931			1964		
	X	Y	Z	X	Y	Z
A	109.850	100.00	35.585	111.144	100.00	35.200
C	98.074	100.00	118.232	97.285	100.00	116.145
D50	96.422	100.00	82.521	96.720	100.00	81.427
D55	95.682	100.00	92.149	95.799	100.00	90.926
D65	95.047	100.00	108.883	94.811	100.00	107.304
D75	94.072	100.00	122.638	94.416	100.00	120.641
F2	99.186	100.00	67.393	103.279	100.00	69.027
F7	95.041	100.00	108.747	95.792	100.00	107.686
F9	100.962	100.00	64.350	103.863	100.00	65.607

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.

Appendix B
Colour Toolbox

B.1 Where to Find the Toolbox

We have gone to great lengths to develop what we believe is a correct implementation of the formulae and methods discussed in the book. However, the data and the software are provided as is with no warranty whatsoever. No part of the tool box is an official CIE or ASTM implementation. The toolbox can be downloaded from MATLAB Central FileExchange - http://www.mathworks.co.uk/matlabcentral/fileexchange/

For further information about the toolbox and to report bugs please visit http://colourmatlab.wordpress.com/

B.2 How to Install the Toolbox

The toolbox should be downloaded and placed in a suitable directory such as matlab/toolboxes/colour

The MATLAB path environment should be set to include the new directory using MATLAB, FILE, SET PATH.

B.3 Summary of Toolbox Files

B.3.1 Computing CIE Tristimulus Values

```
sprague.m
r2xyz.m
cieplot.m
```

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.

B.3.2 CIELAB and Colour Difference

```
xyz2lab.m
lab2xyz.m
xyz2luv.m
cielabde.m
cmcde.m
cie94de.m
cie00de.m
cielabplot.m
```

B.3.3 Chromatic-Adaptation Transforms and Colour Appearance

```
cmccat97.m
cmccat00.m
cmccat00r.m
ciecat02.m
```

B.3.4 Physiological Colour Spaces

```
rgb2lms.m
lms2rgb.m
lms2rgbBM.m
rgbBM2lms.m
```

B.3.5 Colour Management

```
srgb2xyz.m
xyz2srgb.m
```

B.3.6 Display Characterisation

```
crtdemo.m
```

B.3.7 Characterisation of Cameras

```
camera_demo.m
camera_demo1.m
```

B.3.8 Characterisation of Printers

```
printer_demo.m
printer_demo1.m
```

References

Aleksander, I. and Morton, H. (1991) *An Introduction to Neural Computing*, Chapman and Hall.

Alsam, A. and Finlayson, G.D. (2008) Integer programming for optimal reduction of calibration targets, *Color Research and Application*, **33** (3), 212–220.

Anton, H. (1994) *Elementary Linear Algebra*, John Wiley & Sons, Ltd, Chichester.

ASTM (2001) E308-01, *Standard Practice for Computing the Colors of Objects by Using the CIE System*.

Bala, R. (1999) Optimization of the spectral Neugebauer model for printer characterization, *Journal of Electronic Imaging*, **8** (2), 156–166.

Bala, R. (2003) Device Characterization, in *Digital Color Imaging Handbook*, G. Sharma (ed.), CRC Press.

Balasubramanian, R. and Maltz, M.S. (1996) Refinement of printer transformations using weighted regression, *Proceedings of SPIE*, **2658**, 334–340.

Barlow, H.B. (1982) What causes trichromacy? A theoretical analysis using comb-filtered spectra, *Vision Research*, **22**, 635–644.

Barnard, K. and Funt, B. (2002) Camera characterization for color research, *Color Research and Application*, **27** (3), 152–163.

Bartleson, C.J. (1978) Comparison of chromatic adaptation transforms, *Color Research and Application*, **3** (3), 129–136.

Bartleson, C.J. and Breneman, E.J. (1967) Brightness perception in complex fields, *Journal of the Optical Society of America*, **57** (7), 953–957.

Bastani, B., Cressman, B. and Funt, B. (2005) Calibrated color mapping between LCD and CRT displays: A case study, *Color Research and Application*, **30** (6), 438–447.

Berns, R.S. (1993) The mathematical development of CIE TC 1–29 proposed colour difference equation: CIELCH, *Proceedings of the 7th Congress of International Colour Association*, **B**, C19.1–C19.4.

Berns, R.S. (2000) *Billmeyer and Saltzman's Principles of Color Technology*, John Wiley & Sons, Ltd, Chichester.

Berns, R.S. and Katoh, N. (2002) Methods for characterizing displays, in *Colour Engineering*, P. Green, and LW, MacDonald (eds) John Wiley & Sons, Ltd, Chichester.

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.

Berns, R.S., Motta, R.J. and Gorzynski, M.E. (1993a) CRT Colorimetry, part I: Theory and practice, *Color Research and Application*, **18** (5), 299–314.

Berns, R.S., Motta, R.J. and Gorzynski, M.E. (1993b) CRT Colorimetry, part II: Metrology, *Color Research and Application*, **18** (5), 315–325.

Bianco, S. and Schettini, R. (2010) Two new von Kries based chromatic adaptation transforms found by numerical optimization, *Color Research and Application*, **35** (3), 184–192.

Borse, G.J. (1997) *Numerical Methods with MATLAB: A Resource for Scientists and Engineers*, International Thomson Publishing.

Bowmaker, J. (2002) The retina, in *Signals and Perception: The Fundamentals of Human Sensation*, D. Roberts (ed.), Palgrave.

Boynton, R.M. (1996) History and current status of a physiologically based system of photometry and colorimetry, *Journal of the Optical Society of America A*, **13** (8), 1609–1621.

Brainard, D.H. (1996) Cone contrast and opponent modulation color spaces in *Human Color Vision*, P.K. Kaiser, and R.M. Boynton (eds), 2nd edn, Optical Society of America, Washington, DC.

Brainard, D.H. and Wandell, B.A. (1990) Calibrated processing of image color, *Color Research and Application*, **15** (5), 266–271.

Brill, M.H. (1979) Further features of the illuminant invariant trichromatic photosensor, *Journal of Theoretical Biology*, **78**, 305–308.

Brill, M.H. (2006) Irregularity in CIECAM02 and its avoidance, *Color Research and Application*, **31** (2), 142–145.

Buchsbaum, G. and Gottschalk, A. (1983) Trichromacy, opponent colours coding and optimum colour information transmission in the retina, *Proceedings of the Royal Society (London) B*, **220**, 89–113.

Capilla, P., Malo, J., Luque, M.J. and Artigas, J.M. (1998) Colour representation spaces at different physiological levels: a comparative analysis, *Journal of Optics*, **29** (4), 324–338.

Carlson, G.E. (1998) *Signal and Linear System Analysis*, John Wiley & Sons, Ltd, Chichester.

Chaparro, A., Stromeyer, C.F. III, Huang, E.P., Kronauer, R.E., Eskew, R.T. Jr. (1993) Colour is what the eye sees best, *Nature*, **361**, 348–350.

Cheung, T.L.V. and Westland, S. (2002) Color camera characterization using artificial neural networks, *Proceedings of the IS&T/SID's 10th Color Imaging Conference*, 117–120, Scottsdale, Arizona.

Cheung, V. and Westland, S. (2006) Optimal color selection methods, *Journal of Imaging Science and Technology*, **50** (5), 481–488.

Cheung, V., Westland, S., Chiangjun, Li C., Hardeberg, J. and Connah, D. (2005) Characterization of trichromatic color cameras by a new multispectral imaging technique, *Journal of the Optical Society of America A*, **22** (7), 1231–1240.

Cheung, V., Westland, S., Connah, D. and Ripamonti, C. (2004) A comparative study of the characterization of colour cameras by means of neural networks and polynomial transforms, *Coloration Technology*, **120** (1), 19–25.

Chou, W., Lin, H., Luo, M.R., Rigg, B., Westland, S. and Nobbs, J.H. (2001) The performance of the new CIE lightness difference formula, *Journal of Coloration Technology*, **117** (1), 19–29.

CIE (1986a) Publication No. S2, *CIE Standard Colorimetric Observers*, Vienna, Austria, Central Bureau of the CIE.

CIE (1986b) Publication No. 15.2, *Colorimetry* 2nd edn Vienna, Austria, Central Bureau of the CIE.

CIE (1987) Publication No. 17.4, *CIE International Lighting Vocabulary*, Vienna, Austria, Central Bureau of the CIE.

CIE (2001) Publication No. 142, *Improvement to Industrial Colour-Difference Evaluation*, Vienna, Austria, Central Bureau of the CIE.

CIE (2003) CIE TC8-01 Technical Report, *A Color Appearance Model for Color Management Systems: CIECAM02*, Vienna, Austria, Central Bureau of the CIE.

CIE (2004) Publication No. 15, *Colorimetry* 3rd edn Vienna, Austria, Central Bureau of the CIE.

CIE (2005) Publication No. 167, *Recommended Practice for Tabulating Spectral Data for use in Colour*, Vienna, Austria, Central Bureau of the CIE.

CIE (2007) Technical Report 170–1, *Fundamental Chromaticity Diagram with Physiological Axes–Parts 1 and 2*, Vienna, Austria, Central Bureau of the CIE.

Clarke, F.J.J., McDonald, R. and Rigg, B. (1984) Modification to the JPC79 Colour–difference Formula, *Journal of the Society of Dyers and Colourists*, **100** (4), 128–132.

Cohen, J. (1964) Dependency of the spectral reflectance curves of the Munsell color chips, *Psychological Science*, **1**, 369–370.

Connah, D., Westland, S. and Thomson, M.G.A. (2001) Recovering spectral information using digital camera systems, *Journal of Coloration Technology*, **117** (6), 309–312.

Cui, G., Luo, M.R., Rigg, B. and Li, W. (2001) Colour-difference evaluation using CRT colours. Part I: Data gathering and testing colour-difference formulae, *Color Research and Application*, **26** (5), 394–402.

Day, E.A., Taplin, L. and Berns, R.S. (2004) Colorimetric characterization of a computer-controlled liquid crystal display, *Color Research and Application*, **29** (5), 365–373.

DeMarco, P., Pokorny, J. and Smith, V.C. (1992) Full spectrum cone sensitivity functions for X-chromosome linked anomalous trichromats, *Journal of the Optical Society of America A*, **9** (9), 1465–1476.

Derrington, A.M., Krauskopf, J. and Lennie, P. (1984) Chromatic Mechanisms in Lateral Geniculate Nucleus of Macaque *Journal of Physiology*, **357**, 241–265.

Dominy, N.J. and Lucas, P.W. (2001) Ecological importance of trichromatic vision to primates, *Nature*, **410**, 363–366.

D'Zmura, M. and Iverson, G. (1993a) 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.

D'Zmura, M. and Iverson, G. (1993b) Color constancy. II. Results for two-stage linear recovery of spectral descriptions for lights and surfaces, *Journal of the Optical Society of America A*, **10** (10), 2166–2180.

D'Zmura, M. and Lennie, P. (1986) Mechanisms of colour constancy, *Journal of the Optical Society of America A*, **3** (10), 1662–1672.

Emmel, P. and Hersch, R.D. (2000) A unified model for color prediction of halftoned prints, *Journal of Imaging Science and Technology*, **44** (4), 351–359.

Fairchild, M.D. (1998) *Color Appearance Models*, 1st edn, Addison Wesley Longman.

Fairchild, M.D. (2005) *Color Appearance Models*, 2nd edn, John Wiley & Sons, Ltd, Chichester.

Fairchild, M.D. and Lennie, P. (1992) Chromatic adaptation to natural and incandescent illuminants, *Vision Research*, **32** (11), 2077–2085.

Fairchild, M.D. and Reniff, L. (1995) Time course of chromatic adaptation for color-appearance judgements, *Journal of the Optical Society of America A*, **12** (5), 824–833.

Fairchild, M.D. and Wyble, D.R. (1998) Colorimetric characterization of the Apple Studio Display (Flat Panel LCD) *Munsell Color Science Laboratory Technical Report*, Available online at http://www.cis.rit.edu/mcsl/research/PDFs/LCD.pdf (accessed 16 February, 2012).

Finlayson, G.D. and Süsstrunk, S. (2000) Performance of a chromatic adaptation transform based on spectral sharpening, *Proceedings of the IS&T/SID's 8th Colour Image Conference*, 49–55, Scottsdale, Arizona.

Fu, C-C., Wu, C-C., Tseng, S-S., Tzou, C-L. and Huang, J-M. (2008) Channel-dependent GOG model for colorimetric characterization of LCDs, *SID Symposium Digest of Technical Papers*, **39** (1), 1510–1513.

García-Beltrán, A., Nieves, J.L., Hernández-Andrés, J. and Romero, J. (1998) Linear bases for spectral reflectance functions of acrylic paints, *Color Research and Application*, **23** (1), 39–45.

García-Suarez, L. and Ruppertsberg, A.I. (2010) Why higher resolution graphic cards are needed in colour vision research, *Color Research and Application*, **36** (2), 118–126.

Green, P. (2002a) Colorimetry and colour difference, in *Colour Engineering*, P. Green, and LW, MacDonald (eds), John Wiley & Sons, Ltd, Chichester.

Green, P. (2002b) Overview of characterization methods, in *Colour Engineering*, P. Green, and LW, MacDonald (eds) John Wiley & Sons, Ltd, Chichester.

Green, P. (2002c) Characterizing hard copy printers, in *Colour Engineering*, P. Green, and LW, MacDonald (eds), John Wiley & Sons, Ltd, Chichester.

Green, P. (2010) *Color Management: Understanding and Using ICC Profiles*, John Wiley & Sons, Inc., Hoboken, NJ.

Green, P. (2011) Specification of sRGB, Available online at: http://www.color.org/chardata/rgb/srgb.pdf (accessed 16 February, 2012).

Green, P., Holm, J. and Li, W. (2008) Recent developments in ICC color management, *Color Research and Application*, **33** (6), 444–448.

Green, P. and McDonald, L.W. (2002) *Colour Engineering*, John Wiley & Sons, Ltd, Chichester.

Hardeberg, J.Y. (1999) *Acquisition and reproduction of colour images: colorimetric and multi-spectral approaches*, PhD Thesis, Ecole Nationale Supérieur des Télécommunications (France).

Hardeberg, J.Y. (2001) *Acquisition and reproduction of colour images: colorimetric and multi-spectral approaches*, Universal Publishers.

Haykin, S. (1994) *Neural Networks; A Comprehensive Foundation*, Macmillan.

Heptinstall, A. (1999) *The Perception of Achromatic Lightness Differences*, MPhil Thesis, Keele University.

Holm, G. (2006) Capture color analysis gamuts, *Proceedings of the IS&T/SID's 14th Color Imaging Conference*, 108–113, Scottsdale, Arizona.

Hong, W., Luo, M.R. and Rhodes, P.A. (2000) A study of digital camera colour characterization based on polynomial modelling, *Color Research and Application*, **26** (1), 76–84.

Hung, P-C. (1993) Colorimetric calibration in electronic imaging devices using a look-up table method and interpolations, *Journal of Electronic Imaging*, **2** (1), 53–61.

Hunt, R.W.G. (1952) Light and dark adaptation and the reception of color, *Journal of the Optical Society of America*, **42**, 190–199.

Hunt, R.W.G. (1997) Reversing the Bradford chromatic adaptation transform, *Color Research and Application*, **22** (5), 355–356.

Hunt, R.W.G. (1998) *Measuring Colour*, 3rd edn, Fountain Press.

Hunt, R.W.G., Li, C. and Luo, M.R. (2005) Chromatic adaptation transforms, *Colour Research and Application*, **30** (1), 69–71.

Hunt, R.W.G. and Pointer, M.R. (1985) A colour-appearance transform for the 1931 standard colorimetric observer, *Color Research and Application*, **10** (3), 165–179.

Hunt, R.W.G. and Pointer, M.R. (2011) *Measuring Colour*, 4th edn, John Wiley & Sons, Ltd, Chichester.

Hurlbert, A. (1991) Deciphering the colour code, *Nature*, **349**, 191–193.

IEC (2003) *Multimedia Systems and Equipment–Colour Measurement and Management*, IEC 6- 966–9.

Iino, K. and Berns, R.S. (1998) Building color management modules using linear optimization I. Desktop color system, *Journal of Imaging Science and Technology*, **42** (1), 79–94.

Imai, F.H. and Berns, R.S. (1999) Spectral estimation using trichromatic digital cameras, *Proceedings of the International Symposium on Multispectral Imaging and Color Reproduction*, 42–49.

Jiménez, J.R., Reche, J.F., Díaz, J.A., Jiménez del, Barco L. and Hita, E. (1998) Optimization of color reproduction on CRT-color monitors, *Color Research and Application*, **24** (3), 207–213.

Johnson, A.J. (1996) Methods for characterizing colour printers, *Displays*, **16** (4), 193–202.

Johnson, A.J. (2002) Methods for characterizing colour scanners and digital cameras, in *Colour Engineering*, P. Green, and LW, MacDonald (eds) John Wiley & Sons, Ltd, Chichester.

Jolliffe, I.T. (1986) *Principal Components Analysis*, Springer-Verlag, New York.

Judd, D.B., MacAdam, D.L. and Wyszecki, G.W. (1964) Spectral distribution of typical daylight as a function of correlated colour temperature, *Journal of the Optical Society of America*, **54** (8), 1031–1040.

Kaiser, P.K. and Boynton, R.M. (1996) *Human Colour Vision*, Optical Society of America.

Kang, H.R. (1994) Applications of colour mixing models to electronic printing, *Journal of Electronic Imaging*, **3** (3), 276–287.

Kang, H.R. (2006) *Computational Color Technology*, The International Society for Optical Engineering.

Kang, H.R. and Anderson, P.G. (1992) Neural network applications to the colour scanner and printer calibrations, *Journal of Electronic Imaging*, **1** (2), 125–134.

Kohonen, T. (1988) *Self-Organization and Associative Memory*, Springer Verlag.

König, A. and Dieterici, C. (1886) Die Grundempfindungen und ihre Intensitäts-Vertheilung im Spectrum, *Sitzungsberichte Akademie der Wissenschaften*, Berlin, 805–829.

Krauskopf, J., Williams, D.R. and Heeley, D.W. (1982) Cardinal directions of color space, *Vision Research*, **22**, 1123–1131.

Lagarias, J.C., Reeds, J.A., Wright, M.H. and Wright, E. (1998) Convergence properties of the Nelder-Mead Simplex method in low dimensions, *SIAM Journal of Optimization*, **9** (1), 112–147.

Lam, K.M. (1985) Metamerism and Colour Constancy, *PhD Thesis*, University of Bradford.

Land, E.H. (1986) Recent advances in retinex theory, *Vision Research*, **26** (1), 7–21.

Land, E.H. and McCann, J.J. (1971) Lightness and the retinex theory, *Journal of the Optical Society of America*, **61** (1), 1–11.

Li, C.J. and Luo, M.R. (2001) The estimation of spectral reflectances using the smoothness constraint condition, *Proceedings of the IS&T/SID's 9th Colour Image Conference*, 62–27, Scottsdale, Arizona.

Li, C.J. and Luo, M.R. (2005) Testing the robustness of CIECAM02, *Color Research and Application*, **30** (2), 99–106.

Li, C.J., Luo, M.R. and Hunt, R.W.G. (2000) A revision of the CIECAM97s model, *Color Research and Application*, **25** (4), 260–266.

Li, C.J., Luo, M.R., Rigg, B. and Hunt, R.W.G. (2002) CMC 2000 chromatic adaptation transform: CMCCAT2000, *Color Research and Application*, **27** (1), 49–58.

Luo, M.R. (2002a) Development of colour-difference formulae, *Review of Progress in Coloration*, **32**, 28–39.

Luo, M.R. (2002b) The CIE 1997 colour appearance model: CIECAM97s, in *Colour Engineering*, P. Green, and LW, MacDonald (eds) John Wiley & Sons, Ltd, Chichester.

Luo, M.R. (2003) *Private Communication*.

Luo, M.R., Cui, G. and Li, C. (2006) Uniform colour spaces based on CIECAM02 colour appearance model, *Color Research and Application*, **31** (4), 320–330.

Luo, M.R., Cui, G. and Rigg, B. (2001) The development of the CIE 2000 colour-difference formula: CIEDE2000, *Color Research and Application*, **26** (5), 340–350.

Luo, M.R. and Hunt, R.W.G. (1998a) The structure of the CIE 1997 colour appearance model (CIECAM97s) *Color Research and Application*, **23** (3), 138–146.

Luo, M.R. and Hunt, R.W.G. (1998b) A chromatic adaptation transform and a colour inconstancy index, *Color Research and Application*, **23** (3), 154–158.

McCamy, C.S., Marcus, H. and Davidson, J.G. (1976) A color-rendition chart, *Journal of Applied Photographic Engineering*, **2** (3), 95–99.

McCann, J.J., McKee, S.P. and Taylor, T.H. (1976) Quantitative studies in retinex theory. A comparison between theoretical predictions and observer responses to the "colour mondrian" experiments, *Vision Research*, **16** (5), 445–458.

McDonald, R. (1997a) *Colour Physics for Industry*, The Society of Dyers and Colourists.

McDonald, R. (1997b) Recipe prediction for textiles, in *Colour Physics for Industry*, R. McDonald (ed.) Society of Dyers and Colourists.

McDowell, D.Q. (2002) Standards activities for colour imaging, in *Colour Engineering*, P. Green and L.W. MacDonald (eds), John Wiley & Sons, Ltd, Chichester.

MacLeod, D.I.A. and Boynton, R.M. (1979) Chromaticity diagram showing cone excitation by stimuli of equal luminance, *Journal of the Optical Society of America*, **69** (8), 1183–1186.

Mahy, M. (2000) Analysis of color targets for output characterization, *Proceedings of the IS&T 8th Color Imaging Conference*, 348–355, Scottsdale, Arizona.

Mahy, M. and Delabastita, P. (1996) Inversion of the Neugebauer equations, *Color Research and Application*, **21** (6), 404–411.

Maloney, L.T. (1986) Evaluation of linear models of surface spectral reflectance with small numbers of parameters, *Journal of the Optical Society of America*, **3** (10), 1673–1683.

Maloney, L.T. and Wandell, B.A. (1986) Color constancy: A method for recovering surface spectral reflectance, *Journal of the Optical Society of America A*, **3**(1), 29–33.

Marchand, P. (1999) *Graphics and GUIs with MATLAB*, CRC Press.

Melgosa, M. (2006) Improvement of CMC upon CIEDE2000 for a new experimental dataset, *Color Research and Application*, **31** (3), 239–243.

Melgosa, M. and Huertas, R. (2004) Relative significance of the terms in the CIEDE2000 and CIE94 color-difference formulas, *Journal of the Optical Society of America A*, **21** (12), 2269–2275.

Mollon, J.D. (1989) "Tho' she kneel'd in that place where they grew . . ." The uses and origins of primate colour vision, *Journal of Experimental Biology*, **146**, 21–38.

Monga, V. and Bala, R. (2008) Sort-select-damp: an efficient strategy for color look-up table lattice design, *Proceedings of the IS&T/SID's 16th Color Imaging Conference*, 247–253, Portland, Oregon.

Moroney, N. (2000) Usage guidelines for CIECAM97s, *Proceedings of the IS&T's Image Processing, Image Quality, Image Capture, Systems (PICS) Conference*, 164–168, Portland, Oregon.

Moroney, N., Fairchild, M.D., Hunt, R.W.G., Li, C., Luo, M.R. and Newman, T. (2002) The CIECAM02 color appearance model, *Proceedings of the IS&T/SID's 10th Color Imaging Conference*, 23–27, Scottsdale, Arizona.

Morovič, J. (2002) Colour gamut mapping, in *Colour Engineering*, P. Green, and L.W. MacDonald (eds), John Wiley & Sons, Ltd, Chichester.

Morovič, J., Arnabat, J., Richard, Y. and Albarrán, Á (2010) Sampling optimization for printer characterization by greedy search, *IEEE Transactions on Image Processing*, **19** (10), 2705–2711.

Nayatani, Y. (2006) Development of chromatic adaptation transforms and concept for their classification, *Color Research and Application*, **31** (3), 205–214.

Nayatani, Y. and Sakai, H. (2008) An integrated color-appearance model using CIELUV and its applications, *Color Research and Application*, **33** (2), 125–134.

Nayatani, Y., Takahama, K., Sobagaki, H. and Hashimoto, K. (1990) Color appearance model and chromatic adaptation transform, *Color Research and Application*, **15** (4), 210–221.

Nayatani, Y., Yano, T., Hashimoto, K. and Sobagaki, H. (1999) Proposal of an abridged colour-appearance model, *Color Research and Application*, **24** (6), 422–438.

Nobbs, J.H. (1985) Kubelka-Munk theory and the prediction of reflectance, *Review of Progress in Coloration*, **15** (1), 66–75.

Nobbs, J.H. (1997) Colour-match prediction for pigmented materials, in *Colour Physics for Industry*, R. McDonald (ed.), Society of Dyers and Colourists.

Nobbs, J.H. (2002) A lightness, chroma and hue splitting approach to CIEDE2000 colour differences, *Advances in Colour Science and Technology*, **5** (2), 46–53.

Ohta, N. and Robertson, A.R. (2005) *Colorimetry Fundamentals and Applications*, John Wiley & Sons, Ltd, Chichester.

Osorio, D. and Vorobyev, M. (1996) Colour vision as an adaptation to frugivory in primates, *Proceedings of the Royal Society B*, **263**, 593–599.

Oulton, D.P. (2010) *Selected papers on colorimetric theory and colour modelling*, PhD Thesis, University of Manchester.

Oulton, D.P. (2011) *Private Communication*.

Oulton, D.P. and Porat, I. (1992) Control of Colour, by Using Measurement and Feedback, *Journal of Textile Institute*, **83** (3), 454–461.

Owens, H.C. (2002) *Spatiochromatic processing in humans and machines*, PhD Thesis, University of Derby.

Pointer, M.R. and Attridge, G.G. (1998) The number of discernible colours, *Color Research and Application*, **23** (1), 52–54.

Post, D.L. and Calhoun, C.S. (1989) An evaluation of methods for producing desired colors on CRT monitors, *Color Research and Application*, **14** (4), 172–186.

Poynton, C. (2002) *Frequently asked questions about gamma*, Available at: http://www.poynton.com/PDFS/GammaFAQ.,pdf (accessed 16 February, 2012).

Pratt, W.K. (1978) *Digital Image Processing*, John Wiley & Sons, Ltd, Chichester.

Press, W.H., Flannery, B.P., Teulolsky, S.A. and Vetterling, W.T. (1993) *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press.

Purdy, D McL. (1931), Spectral hue as a function of intensity, *American Journal of Psychology*, **43** (4), 541–559.

Rich, D. (2002) Instruments and methods for colour measurement, in *Colour Engineering*, P. Green, and L.W. MacDonald (eds), John Wiley & Sons, Ltd, Chichester.

Ripamonti, C., Woo, W.L., Crowther, E. and Stockman, A. (2009) The S-cone contribution to luminance depends on the M- and L-cone adaptation levels: silent surrounds? *Journal of Vision*, **9** (3), 10, 11–16.

Rumelhart, D.E. and McClelland, J.L. (1986) *Parallel Distributed Processing*, Vols I and II, MIT Press.

Rushton, W.A.H. (1972) Pigments and signals in colour vision, *Journal of Physiology*, **220**, 1–31P.

Schanda, J. (2007) *Colorimetry: Understanding the CIE System*, Wiley Blackwell.

Seve, R. (1991) New formula for the computation of CIE 1976 hue difference, *Color Research and Application*, **16** (3), 217–218.

Sharma, G. (2002) Comparative evaluation of color characterization and gamut of LCDs versus CTs, *Proceedings of SPIE, Color Imaging: Device-Independent Color, Color Hardcopy, and Applications VII*, R. Eschbach, and G. Marcu (eds), 177–186.

Sharma, G. (2011) *The CIEDE2000 Color-Difference Formula*, Available at: http://www.ece.rochester.edu/~gsharma/ciede2000/ (accessed 16 February, 2012).

Sharma, G., Wu, W. and Dalal, E.N. (2005) The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations, *Color Research and Application*, **30** (1), 21–30.

Shaw, M., Sharma, G., Bala, R. and Dalal, E.N. (2003) Color printer characterization adjustment for different substrates, *Color Research and Application*, **28** (6), 454–467.

Shepard, D. (1968) A two-dimensional interpolation function for irregularly spaced data, *Proceedings of 23rd National Conference of the ACM*, 517–524.

Shen, S. and Berns, R.S. (2011) Color-difference formula performance for several datasets of small color differences based on visual uncertainty, *Color Research and Application*, **36** (1), 15–26.

Shi, M. and Healey, G. (2002) Using reflectance models for color scanner calibration, *Journal of the Optical Society of America A*, **19**, 645–656.

Smith, K.J. (1997) Colour-order systems, colour spaces, colour difference and colour scales, in *Colour Physics for Industry*, R. McDonald (ed.), The Society of Dyers and Colourists.

Smith, V.C. and Pokorny, J. (1975) Spectral sensitivity of the foveal cone photopigments between 400 and 500 nm, *Vision Research*, **15**, 161–171.

Stearns, E.I. and Stearns, R.E. (1988) An example of a method for correcting radiance data, *Color Research and Application*, **13** (4), 257–259.

Stevens, J.C. and Stevens, S.S. (1963) Brightness function: effects of adaptation, *Journal of the Optical Society of America*, **53** (3), 375–385.

Stiles, W.S. and Burch, J.M. (1959) NPL colour-matching investigation: Final report, *Optica Acta*, **6**, 1–26.

Stockman, A. and Sharpe, L.T. (2000) Spectral sensitivities of the middle- and long-wavelength sensitive cones derived from measurements in observers of known genotype, *Vision Research*, **40**, 1711–1737.

Stokes, M., Anderson, M., Chandrasekar, S. and Motta, R. (1996) *A standard default color space for the internet*, Available at: http://www.w3.org/Graphics/Color/sRGB.html (accessed 16 February, 2012).

Sueeprasan, S. (2003) *Evaluation of colour appearance models and indices for evaluation of daylight simulators to provide predictable cross-media colour reproduction*. PhD Thesis, Derby University.

Sun, Q. and Fairchild, M.D. (2001) Statistical characterization of spectral reflectances in spectral imaging of human portraiture, *Proceedings of the IS&T/SID's 9th Colour Image Conference*, 73–79, Scottsdale, Arizona.

Tastl, I., Recker, J.L., Zhang, Y. and Beretta, G. (2009) An efficient high quality color transformation, *Proceedings of the IS&T/SID's 17th Color Imaging Conference*, 111–116, Albuquerque, New Mexico.

Terstiege, H. (1972) Chromatic adaptation: a state-of-the-art report, *Journal of Color Appearance*, **1** (4), 19–23 (cont. 40).

Thomas, J-B., Hardeberg, J.Y., Foucherot, I. and Gouton, P. (2008) The PLVC display color characterization model revisited, *Color Research and Application*, **33** (6), 449–460.

Thomson, M.G.A. and Westland, S. (2002) Color-imager calibration by parametric fitting of sensor responses, *Color Research and Application*, **26** (6), 442–449.

Tominaga, S. and Wandell, B.A. (1990) Component estimation of surface spectral reflectance, *Journal of the Optical Society of America A*, **7** (2), 312–317.

Tzeng, D.-Y. and Berns, R.S. (2005) A review of principal component analysis and its application to color technology, *Color Research and Application*, **30** (2), 84–98.

Venable, W.H. (1989) Accurate tristimulus values from reflectance data, *Color Research and Application*, **14** (5), 260–267.

Viggiano, J.A.S. (1990) Modeling the color of multi-colored halftones, *Proceedings of TAGA*, 44–62.

von Kries, J. (1905) Die Gesichtsempfindungen in W. Nagel (ed.), *Handbuch der Physiologie des Meschen* pp. 10982. Vieweg Braunschweig.

Vos, J.J., Estévez, O. and Walraven, P.L. (1990) Improved color fundamentals offer a new view on photometric additivity, *Vision Research*, **30**, 936–943.

Wandell, B.A. (1995) *Foundations of Vision*, Sinauer Associates Incorporated.

Westland, S. and Ripamonti, C. (2004) *Computational Colour Science using MATLAB*, John Wiley & Sons, Ltd, Chichester.

Westland, S., Shaw, A.J. and Owens, H.C. (2000) Colour statistics of natural and man-made surfaces, *Sensor Review*, **20** (1), 50–55.

Williams, D., MacLeod, D.I.A. and Hayhoe, M. (1981) Punctate sensitivity of the blue sensitive mechanism, *Vision Research*, **21**, 1357–1375.

Wyszecki, G. and Stiles, W.S. (1982) *Color Science*, John Wiley & Sons, Inc., New York.

Xu, H., Luo, M.R. and Rigg, B. (2003) Evaluation of daylight simulators Part 1: Colorimetric and spectral variations, *Coloration Technology*, **119** (2), 59–69.

Ye, P. and Gu, C. (1999) *Optics of Liquid Crystal Displays*, John Wiley & Sons, Inc., New York.

Yoonessi, A. and Kingdom, F.A.A. (2007) Faithful representation of colours on a CRT monitor, *Color Research and Application*, **32** (5), 388–393.

Yule, J.A.C. (1967) *Principles of Color Reproduction–Applied to Photomechanical Reproduction, Color Photography, and the Ink, Paper, and Other Related Industries*, John Wiley & Sons, Inc., New York.

Yule, J.A.C. and Nielsen, W.J. (1951) The penetration of light into paper and its effect on halftone reproduction, *Proceedings of TAGA*, 65–76.

Index

Computational Colour Science using MATLAB®, Second Edition. Stephen Westland, Caterina Ripamonti and Vien Cheung.