COLOUR DIFFERENTIATION IN DIGITAL IMAGES

Zhenliang Shen

This thesis is presented in fulfillment of the requirements
for the degree of Master of Science

School of Computer Science and Mathematics
Victoria University of Technology
Australia

2003

Declaration

I declare that, to the best of my knowledge, this thesis contains no materials that have been accepted for the award of any other degree or diploma in any university. It is submitted in fulfillment of the candidature for the degree of Masters by Research at Victoria University of Technology, Australia. The materials presented in this thesis are the products of the author's own independent research under the supervision of Dr. Alasdair McAndrew.

Zhenliang Shen

December 2003

Abstract

To measure the quality of green vegetables in digital images, the colour appearance of the vegetable is one of the main factors. In general, green colour represents good quality and yellow colour represents poor quality empirically for green-vegetable. The colour appearance is mainly determined by its hue, however, the value of brightness and saturation affects the colour appearance under certain conditions. To measure the colour difference between green and yellow, a series of experiments have been designed to measure the colour difference under varying conditions. Five people were asked to measure the colour differences in different experiments. First, colour differences are measured as two of the values hue, brightness, and saturation are kept constant. Then, the previous results are applied to measure the colour difference as one of the values hue, brightness, and saturation is kept constant. Lastly, we develop a colour difference model from the different values of hue, brightness, and saturation. Such a colour difference model classifies the colours between green and yellow.

A windows application is designed to measure the quality of leafy vegetables by using the colour difference model. The colours of such vegetables are classified to represent different qualities. The measurement by computer analysis conforms to that produced by human inspection.

Acknowledgment

I am deeply indebted to my supervisor, Dr. Alasdair McAndrew, for his patient, encouraging supervision, invaluable guidance and assistance, and constructive criticism during the research and preparation of the thesis.

I gratefully acknowledge Dr. Hao Shi, for her introduction of this project and her enlightening advice and suggestion. Also, I appreciate Mr. Graeme Thomson, sponsor of Institute of Horticultural Development, Agriculture Victoria, for his supporting of this project.

Table of Contents

Declaration……………………………………………………………………...……..i

Abstract……………………………………………………………………….……..ii

Acknowledgment…………………………………………………………..…….iii

Table of Contents…………………………………………………………. ……iv

List of Figures……………………………………………………………..…..viii

List of Tables……………….……………………………………….............……..x

1. Introduction ……………………………………………………….………..1

2. Research Background ……..…….. …….……………………………….……..3

2.1 Introduction……………………………………………………………..3

2.2 The Properties of Light..........….………………………………………4

2.3 Colour Fundamentals.……………………………………………………..6

2.3.1 Stimuli of Human Eyes……………………………………..……..7

2.3.2 Tri-chromatic Theory……………………………………….……..8

2.3.3 The CIE Chromaticity System……………………………... ……..9

2.3.4 Colour Gamut ……...……………………………………………11

2.4 Colour Model ...…………………………………………………………11

2.4.1 RGB Colour Model…………………………………………………12

2.4.2 CMY Colour Model……………………………………...……13

2.4.3 YIQ Colour Model……………………………………………….……14

2.4.4 HSV Colour Model...........14
2.4.5 CIEL*a*b* Colour Model...........16
2.5 Colour Difference and Colour Tolerance...........17
2.6 Just Noticeable Differences...........19
2.7 Summary...........20

3. Colour Displaying and Measurement...........21
3.1 Introduction...........21
3.2 Image Format...........21
3.3 CRT Monitor...........23
3.4 sRGB Colour Model...........24
3.5 Colour Measurement...........26
3.5.1 RGB Colour Measurement...........27
3.5.2 HSV Colour Measurement...........29
3.5.3 CIEL*a*b* Colour Measurement...........30
3.6 Conclusion...........31

4. Colour Model Conversions...........32
4.1 Introduction...........32
4.2 Conversion Between RGB and HSV...........32
4.3 Conversion Between RGB and CIE XYZ...........34
4.4 Conversion Between CIE XYZ and CIEL*a*b*...........37
4.5 Conversion Between CIEL*a*b* and CIEL*C*H*...........38
4.6 CIE94 Colour Difference Formula...........40
4.7 Conclusion...........41

5. Exploration of Colour Difference in MATLAB...........42
5.1 Introduction...........42
5.2 Colours with Same Brightness of RGB Colour Model in CIEL*a*b* Colour Model...........43

5.3 Colours with Same Hue of RGB Colour Model in CIEL*a*b* Colour Model …………………………………………………………… …... 47

5.4 Colour Differences with Different Values of Hue …………………… …... 50

5.5 Colour Difference with Different Saturation and Brightness ……...……53

5.5.1 Same Hue and Brightness with Varying Saturation ………. ……54

5.5.2 Same Hue and Saturation with Varying Brightness……….. ……55

5.6 Conclusion ……………………………………………………… ……56

6. Development of Colour Difference in Green Vegetable………………... ……57

6.1 Introduction ……………………………………………………...……57

6.2 Colour Difference in Different Conditions ……………………...……58

6.3 Minimum Value of Brightness and Saturation Difference by Human Perception ……………………………………………………… ……59

6.3.1 Minimum Value of Brightness with Same Hue and Saturation….59

6.3.2 Minimum Value of Saturation with Same Hue and Brightness….61

6.4 Colour Measurement with Different Hue and Saturation………….. ……62

6.4.1 Minimum Hue Difference with Different Saturation ……... ……64

6.4.2 Colour Difference Between Different Hue and Saturation ...……65

6.4.3 Colour Difference Between Same Hue with Different Saturation and Same Saturation with Different Hue…………………...……66

6.5 Colour Measurement with Different Hue and Brightness ………………70

6.6 Colour Differences with Varying Brightness and Saturation ……... ……74

6.7 Colour Difference under Any Condition ………………………….. ……77

6.8 Conclusion……………………………………………………….. ……81

7. Windows Application of the Colour Difference Model ……………….. ……82

7.1 Introduction ……………………………………………………...……82

7.2 Analysis and Design…………………………………………….. ……83

7.2.1 Procedure 1 ……………………………………………... ……84

7.2.2 Procedure 2 ……………………………………………... ……84

7.2.3 Procedure 3 ……………………………………………... ……86

7.2.4 Procedures 4 and 5 ……………………………………………… ……87
7.2.5 Procedure 6 …………………………………………………... ……89
7.3 Interface of The Application …………………………………………90
7.4 The Correctness of the Result ……………………………….. ……91
7.5 The Range of The Application……………………………...……92
7.6 Conclusion …………………………………………………… ……93

8. Conclusion and Future Work ………………………………………… ……94
8.1 Conclusions …………………………………………………...……94
8.2 Future Work…………………………………………………... ……95

References ………………………………………………………………………97

Appendix A: Source Codes of MATLAB………………………………….…..104

Appendix B: Sample Colours ………………………………….……………..116

Appendix C: Surveys and Results ……………………………….…………..132

Appendix D: Source Codes of VB ……………………………….…………..142

List of Figures

2-1 Electromagnetic spectrum……………………………………………………..……..5

2-2 Energy distribution of light with a dominant frequency………………………… ……..7

2-3 Spectral–response function of each type of cone………………………..……..8

2-4 Colour-matching function with all wavelength of the visible spectrum…………..8

2-5 CIE 1931 standard observer…………………………………………..……..9

2-6 The CIE chromaticity diagram…………………………………………..……..9

2-7 The RGB cube……………………………………………………………13

2-8 The CMY cube………………………………………………………..……13

2-9 The HSV hexcone…………………………………………………….. ……15

2-10 The CIEL*a*b* colour model……………………………………….. ……17

3-1 The relationship of tints, tones, and shades…………………………….. ……28

3-2 Colours of HSV with constant hue in CIEL*a*b*…………………………….…...30

4-1 Three directions views of RGB in CIEL*a*b*……………………………..……39

4-2 RGB colour gamut in CIEL*a*b*………………………………………. ……40

5-1 Colours of RGB in CIEL*a*b* with specific brightness………………...……46

5-2 Colours of RGB in CIEL*C*H* with 108 degree hue…………………….. ……49

5-3 Colours of RGB in CIEL*C*H* with specific hue……………………... ……51

5-4 Colours of RGB in CIEL*C*H* with specific brightness and hue……….. ……53
5-5 Colours of RGB in CIEL*C*H* with specific brightness and saturation…. ……56

6-1 Colours with 110 degree hue and 70 saturation but different brightness…..……60
6-2 Colours with 140 degree hue and 70 brightness but different saturation….. ……61
6-3 Colour patches with different hue and saturation in as the brightness is 70..……63
6-4 Colours with different hue and saturation ………………………………….…66
6-5 Lines of equation 6-5 obtained from the values of experiments…………... ……69
6-6 Colour patches with different brightness and hue as the saturation is 50…..……72
6-7 Lines of equation 6-6 obtained from the dots of experiment………….………74
6-8 Colour patches with different brightness and saturation
as the hue is 125 degree…………………………………………….……… ……75

7-1 The procedures of the application……………………………………..……83
7-2 Quality of vegetable and the sample colours…………………………….……85
7-3 Interface of the application………………………………………………… ……91
7-4 Original image and the output of graphic and text………………………… ……92

List of Tables

2-1 Colour appearance in different wavelength……………………………... ……..6
2-2 The agreement of human visual in different tolerance method……………. ……19

3-1 Numbers of colours represented by numbers of bits……………………… ……22
3-2 sRGB viewing environment conditions……………………………………. ……25
3-3 CIE chromaticities for ITU-R BT.709 reference primaries and
CIE standard illuminant…………………………………………………. ……25
3-4 Colour values in different colour models………………………………...……28

5-1 Results of measuring colours with different brightness……………..……...……47
5-2 Results of measuring colour with different hue…………………………. ……50
5-3 Results of measuring hue difference……………………………………..……52

6-1 Colours varying with different condition………………………………...……59
6-2 Minimum value of hue difference………………………………………. ……64
6-3 Sample colours with specific values…………………………………….. ……66
6-4 Results of measuring threshold value of saturation
with 3 degree hue difference …………………….…………………...……67
6-5 Average results of threshold value of saturation with different hue………..……68
6-6 Average results of threshold value of brightness with different hue………. ……73
6-7 Average results of threshold values of brightness with different saturation..……76
6-8 Colour measurement with different condition…………………………... ……78

Chapter 1

Introduction

Digital images have been widely applied in recent years, with hardware increasing in power and speed, and the cost of equipment reducing greatly. Images can provide enough details with high resolutions and a large numbers of colours for human perception and interpretation. Information from the images can be used to solve the problems from many industrial and scientific applications.

Colour vision is to provide a description of the transformation steps from light absorption to vision. A zone theory of colour is to emphasize the general principles involved in transforming light energy to visual sensation [Mas78]. The human eye can discern thousands of colour shades and intensities, compared to about few dozens shades of grey. [Fol96] Furthermore, current colour reproduction techniques (such as digital colour photo, CRT) can provide enough colours for the digital image to be indistinguishable from the scene or object it represents. Therefore, a 24-bit colour image provides enough information for human inspection, perception, and analysis. [Fol94]

Colour image processing is used in many fields of application, such as manufacturing, agriculture, commercial, and military. In agriculture, the colour of crops, forests, and flowers in images can be evaluated according to their ripeness, sizes, and quality. In this thesis, we will focus on investigating colour difference to determine the quality of leafy green vegetables, especially in regard to colours between green and yellow. [Gon02] Moreover, since more than 95% [Hun95] of people have almost the same perception of colours, a group of volunteers (with same numbers of male and female) are asked to participate will generate acceptable results for this project. [Wys67]

The thesis is divided into eight chapters. Chapter two is a brief overview of the research background, including the introduction of optics, colour science, and digital image processing.

Chapter three introduces several colour spaces, **RGB**, **HSV**, and **CIEL*a*b***. Every colour space has been designed to mimic the function of the human eye. Although the values of these colour spaces can be converted, they are designed to be used for different purposes. Comparing human perception and computer analysis, each space is investigated, and their advantages and disadvantages are discussed.

Chapter four further investigates the relationship among the different colour spaces. In particular, algorithms are demonstrated to transform between different colour spaces.

In chapter five, MATLAB is introduced to analyse colour differences. The powerful array operations of MATLAB are very suited for digital image processing. Also MATLAB can be used to produce visual results as well as abstract data.

The purpose of Chapter six is to analyse greens and yellows within a digital image, and how the computer distinguishes the colour difference compared with the variation of perception of human vision. Some synthetic digital images are created with different brightness, saturation, and hue value. The images are assessed according to human evaluation of colour difference. According to the results, a computer model is created to mimic human judgment.

Chapter seven gives an application with some leafy vegetable images. The program is coded with Visual Basic. The procedures and functions of the software are listed and explained. And Chapter eight concludes the research and discusses some directions for future work.

Chapter 2

Research Background

2.1 Introduction

Colour images provide more details than a grey scale image; as well as different intensities, different colours describe the image. There are many factors which determine the quality of image. Two of the major factors are resolution and numbers of colours. The resolution is determined by the numbers of pixels per square unit, the more pixels the image has and the more clarity the image has. The number of colours in a digital image is a function of the number of bits used to describe one pixel. Normally, a 24-bit digital colour image has enough colours to represent all possible images for human perception and interpretation; such an image has more than 16 million different colours [Cui00]. [Hea94] Although colour images are widely applied, they are more complicated to process than grey scale image. [Son99]

There are three facts to be considered for colour image processing:

1. How humans perceive natural light and colour
2. How the image is stored and displayed by the computer and displaying equipment
3. How to create a mathematical model to mimic the human vision

These problems cover the areas of optics, colour science, and image processing. [Gon02]

There are many applications using machine vision technology have been developed in agricultural sectors, such as land-based and aerial-based remote sensing for natural resources assessments, precision farming, postharvest product quality and safety detection, classification and sorting, and process automation. This is because machine vision systems not only recognize size, shape, color, and texture of objects, but also provide numerical attributes of the objects or scene being imaged. Besides imaging objects in the visible color region, some machine vision systems are also able to inspect these objects in light invisible to humans, such as ultraviolet, near-infrared, and infrared. The information received from objects in invisible light regions can be very useful in determining preharvest plant maturity, disease, or stress states. It is very useful in determining plant and vegetable variety, maturity, ripeness, and quality. It is also useful in detecting postharvest quality and safety, such as defects, composition, functional properties, diseases and contamination of plants, grains and nuts, vegetables and fruits, and animal products. Only the visible light is concerned in this thesis. [Tij01] [Yin99]

Advantages of using imaging technology for sensing are that it can be fairly accurate, nondestructive, and yields consistent results. Applications of machine vision technology will improve industry's productivity, thereby reducing costs and making agricultural operations and processing safer for farmers and processing-line workers. It will also help to provide better quality and safe foods to consumers. Machine vision discussed here is limited to camera machine vision systems. It holds great potential and benefits for the agricultural industry because of its simplicity, low cost, rapid inspection rate, and broad range of applications. Machine vision can also be performed using X-ray imaging and nuclear magnetic resonant imaging (MRI). X-ray and MRI imaging are widely used in medical applications. Even though they have potential for detecting diseases and defects in agricultural products and food [Che89], their applications in the agricultural sector are limited because of the high cost of equipment investment and low operational speed [Mar98] [Sch95].

2.2 The properties of Light

Light may be defined as electromagnetic waves in a narrow band of the electromagnetic spectrum. [Hea94] The different lights in the spectrum correspond to difference in either wavelength λ or frequency f. The wavelength and frequency of the wave are inversely proportional to each other, with the proportionality constant as the speed of light c:

$$c = \lambda f \tag{2-1}$$

The wavelengths are normally measured in nano-meters (nm); one nano-meter equals one thousand-millionth (10^{-9}) of a metre. Only the wavelengths from 400 nm to 700 nm are visible. (See Figure 2-1) Each frequency value within the spectrum corresponds to a distinct colour. Table 2-1 lists the main spectral and their approximate wavelength ranges. There is a gradual transition from one colour to another colour so that no colour in the spectrum ends abruptly, but rather each colour blends smoothly into the next. [Ado00] [Gon02]

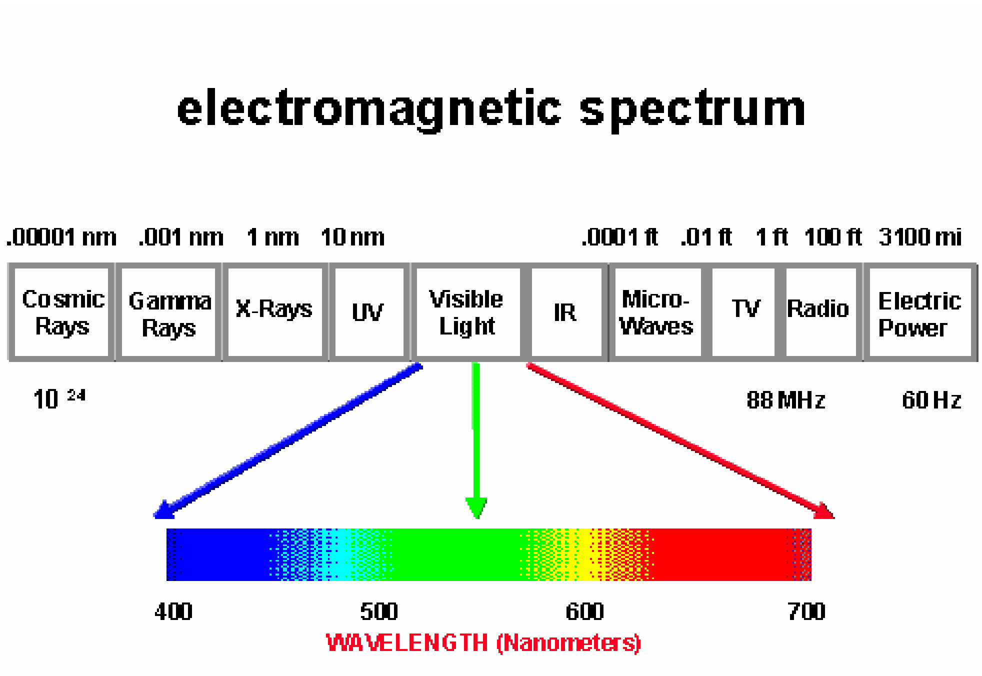


Figure 2-1 Electromagnetic spectrum [Ado00]

Colours	Wavelength
Violet	450 nm or less

Blue	450 nm to 480 nm
Blue-green	480 nm to 510 nm
Green	510 nm to 550 nm
Yellow-green	550 nm to 570 nm
Yellow	570 nm to 590 nm
Orange	590 nm to 630 nm
Red	630 nm and greater

Table 2-1 Colour appearance in different wavelength [Gon02]

A light source such as the sun or a light bulb emits white light. However, there is no wavelength in the spectrum corresponding to white light. In fact, the light of the sun or a light bulb includes all wavelengths from the spectrum, which there is no illumination bias in colorimetry. This means that the combination of all wavelengths of visible light produces white light. [Wys67] The perceived colour has a dominant frequency (or dominant wavelength) at the spectrum; this is called *hue*. For example, red colours have low dominant frequencies and green colours have high dominant frequencies. (See Figure 2-2) Besides dominant frequency, there are two factors which affect the perception of colours, *intensity* and *purity (or saturation)*. Intensity is the light energy of the dominant frequency. Purity is a measurement of the energy of the white light at that colour. [Hea94] In figure 2-2, E_D is the energy of the dominant frequency and E_W is the energy of an average white light. The purity is 100 percent when $E_W = 0$ and is 0 percent when $E_W = E_D$.

2.3 Colour fundamentals

Isaac Newton said, "Indeed rays, properly expressed, are not coloured." [New55] The different colours we perceive are determined by two factors: the nature of the light reflected from the object and the source of the light. The reason tomatoes look red is that they absorb most of the violet, blue, green, and yellow components of the daylight, and reflect mainly the red components. Leaves look green because they only reflect the green colours and absorb the red and blue colours. The source of light determines what colours can be reflected. Sunlight combines all lights of wavelengths, so objects appear coloured in daylight. If the light source has a single wavelength, then objects just reflect this wavelength light and no other lights. When tomatoes are viewed in a green light, they look dark or grey. This is why the leaves of trees look grey under sodium lamps at night, since the sodium lamp emits orange light. [Flo94] [Hun95]

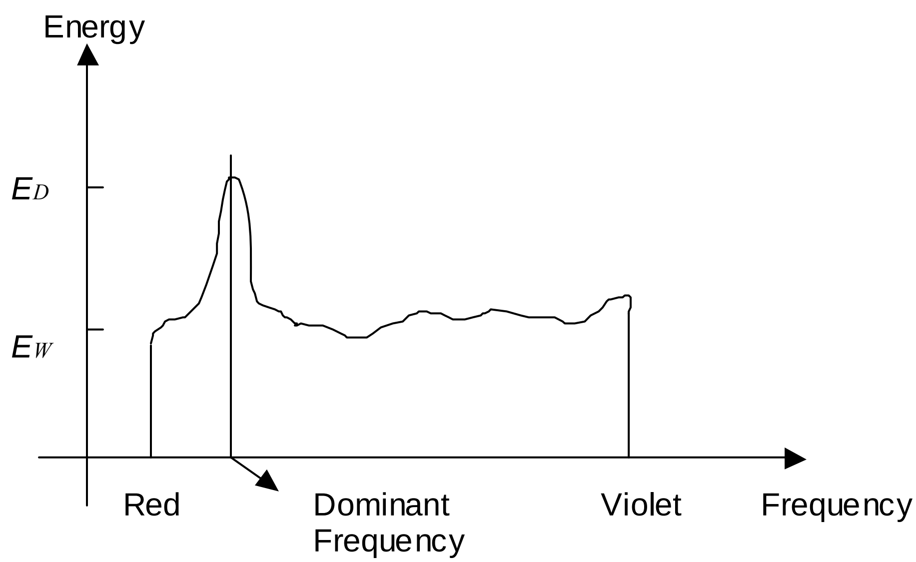


Figure 2-2 Energy distribution of light with a dominant frequency

2.3.1 Stimuli of human eyes

We shall not discuss in detail the anatomy and physiology of the human eye and visual system. Briefly, "the visible incident light enters the cornea, traverses the optic media, and penetrates the retina". [Wys67] The light stimulates the cells of retina, and then the cells produce signals to transmit to brain by the optic nerve. The retina contains two main types of light-sensitive cells, known as *rods* and *cones*. The rods mainly sense the weak light, such as moonlight or starlight. The cones respond to colours. There are three different types of cones, which absorb different lights wavelength; red, green, and blue. Figure 2-3 shows the fraction of light absorbed by each type of cone. The peak blue response is around 450 nm; that for green is about 530 nm; and that for red is about 630 nm. The curves suggest that the eye's response to blue light is much less strong than its response to red or green. [Hun95] [Wys67]

2.3.2 Tri-chromatic theory

Perception of colour is subjective. In fact, colours only exist in the brain; some animals are colour blind. [Poynt] Even individual people do not always have the same colour perception as others. [Fai98] According to human perception, colour reproduction can be accomplished by weighting sums of red, green, and blue. Figure 2-3 shows the amounts of red, green, and blue light that match the colour for all values of light in the visible spectrum by an average observer. The negative value in Figure 2-4 means that the colours cannot be matched by adding primary colours in this area. It will be discussed soon. [Nav00] [Wys67]

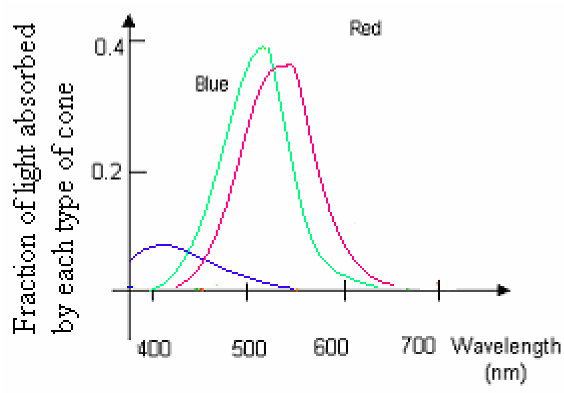


Figure 2-3 Spectral–response function of each type of cone [Fol96]

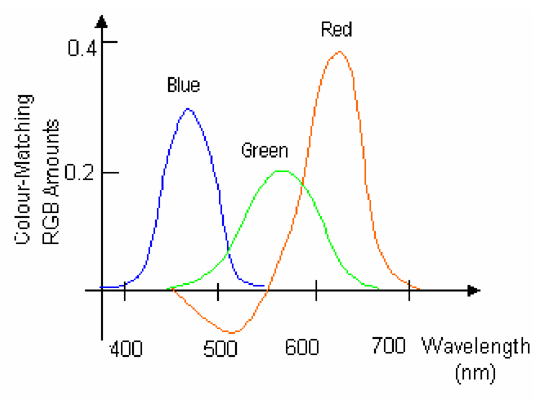


Figure 2-4 Colour-matching functions with all wavelength of spectrum [Fol96]

Two or three different colour light sources with suitably chosen intensities can be used to produce a range of other colours. [Wys67] If the two colour sources combine to produce white light, they are referred to as complementary colours, for example, red and cyan, green and magenta, and blue and yellow. Typically, "colour models that are used to describe combinations of light in terms of dominant frequency use three colours to obtain a reasonably wide range of colours". [Hun95] The three colours used to produce other colour in such a colour model are referred to as *primary colours*. It is impossible that any colour model reproduces all wavelength light in visible spectrum. The set of colours that can be obtained by a combination of primary colours is called the *colour gamut* for that model. [Cas96] [Hun95]

2.3.3 The CIE chromaticity system

In 1931, the International Commission on Illumination, CIE (Commission Internationale del'Eclairage), defined three standard primary colours to be combined to produce all possible perceivable colours. [Fol96] The three standard primaries of the 1931 CIE, called **X**, **Y**, and **Z,** are imaginary colours. [Wys67] (See Figure 2-5) They are defined mathematically with positive colour matching functions, whose values are specified the amount of each primary needed to describe any spectral colour at 5 nm intervals. The CIE primaries provide an international standard definition for all colours, and it eliminates negative value colour matching (Figure 2-4) and other problems associated with selecting a set of real primaries. The **Y** primary was intentionally defined to match the response of the human eye to brightness. More details are given following. [Fol96] [Hil90] [Nav00]

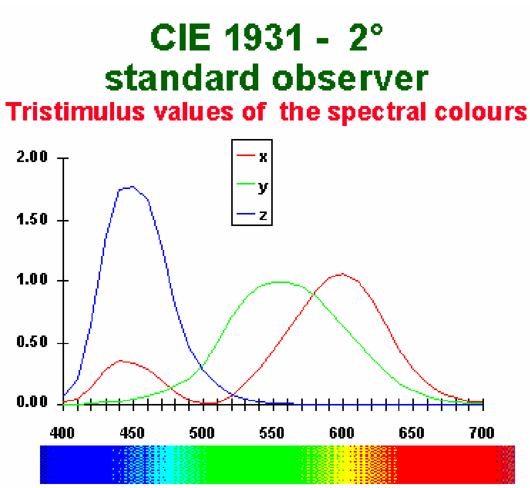


Figure 2-5 CIE 1931 standard observer [Nav00]

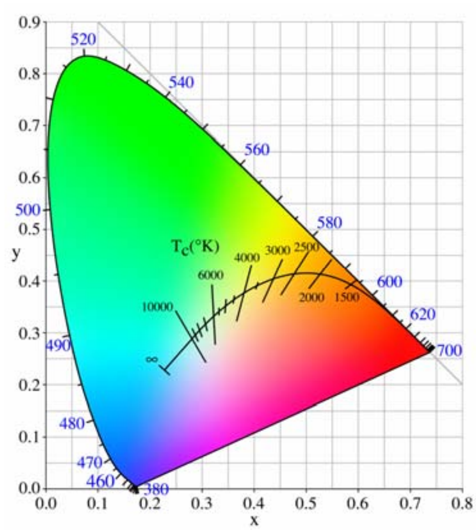


Figure 2-6 The CIE chromaticity diagram [Nav00]

The amounts of **X**, **Y**, and **Z** primaries needed to match a colour with a *spectral energy* distribution P (λ) (the spectral energy distribution is the distribution of light among various wavelengths) [Fol96], are:

$$X = k\int P(\lambda)\overline{x_\lambda}d\lambda \qquad Y = k\int P(\lambda)\overline{y_\lambda}d\lambda \qquad Z = k\int P(\lambda)\overline{z_\lambda}d\lambda \tag{2-2}$$

where, k is a constant; it is 680 lumens/watt for a CRT; the $\overline{x_\lambda}$, $\overline{y_\lambda}$, and $\overline{z_\lambda}$ are colour-matching functions. The *X*, *Y*, and *Z* values are the weights applied to the CIE primaries to match a colour **C**: [Hea94]

$$\mathbf{C} = X\,\mathbf{X} + Y\,\mathbf{Y} + Z\,\mathbf{Z}. \tag{2-3}$$

We define *chromaticity values* that depend only on dominant wavelength and saturation and are independent of the amount of luminous energy by normalizing against *X*+*Y*+*Z*, which can be thought of as the total amount of light energy: [Hea94]

$$x = \frac{X}{X+Y+Z} \qquad y = \frac{Y}{X+Y+Z} \qquad z = \frac{Z}{X+Y+Z} \tag{2-4}$$

Here, x + y + z =1. If we use (x, y, *Y*) to recover all possible colours, the corresponding (*X*, *Y*, *Z*) is: [Hea94]

$$X = \frac{x}{y}Y \qquad Y = Y \qquad Z = \frac{1-x-y}{y}Y \tag{2-5}$$

Chromaticity values depend only on dominant wavelength and saturation and are independent of the amount of luminous energy. [Hun95] By plotting x and y for all visible colours, a horseshoe-shaped diagram can be drawn which is called the *CIE chromaticity diagram*. The interior and boundary of the diagram represent all visible chromaticity values. (See Figure 2-6) The boundary of the diagram represents the 100 percent pure colours of the spectrum. The line joining the red and violet spectral points, called the purple line, is not part of the spectrum. The center point E of the diagram represents a standard white light, which approximates sunlight. Luminance values are not available in the chromaticity diagram because of normalization. Colours with different luminance but the same chromaticity have the same point. The chromaticity diagram is useful for the following: [Hil90] [Gon02]

- Comparing colour gamut for different sets of primaries.
- Identifying complementary colours.
- Determining the dominant wavelength and purity of a given colour.

2.3.4 Colour gamut

Colour gamuts are represented on the chromaticity diagram as straight-line segments or as polygons. [Hun95] Any two colours, for example B and G in Figure 2-6, can be added to produce any colour along their connecting line by varying the relative amounts of the two colours being added. The colour gamut for three points, for example the R, G, and B in Figure 2-6, colour R can be combined with various mixtures of B and G to produce the gamut of all colours in triangle **RGB** by varying relative amounts. The triangle in Figure 2-6 shows a gamut of **RGB** colour model. Three primaries, red, green, and blue, can only generate colours inside or on the bounding edges of the triangle. There is no single triangle which completely fills the CIE chromaticity diagram. That is why no set of three primaries can be additively combined to generate all visible colours. [Ado00] [Bra98] [Hun95]

2.4 Colour model

A *colour model* is a method by which humans can specify, create and visualize colour. [For98] A colour model is a specification of a 3D colour coordinate system and a visible subset in the coordinate system within which all colours in a particular colour gamut lie. For example, the **RGB** colour model is the unit cube subset of the 3D Cartesian coordinate system. (See figure 2-7) There is more than one colour model. The purpose of a colour model is to allow convenient specification of colours within some colour gamut. However, no colour model can be used to specify all visible colours. This is emphasized in the Figure 2-6, which shows that the gamut of **RGB** colour model is a subset of **CIE XYZ** color model. [Fai98] [For98] [Cas96]

The choice of a colour model is based on the application. [Poynt] Some equipment has limiting factors that dictate the size and type of colour model that can be used; for example, the **RGB** colour model is used with colour CRT monitors, the **YIQ** color model is used with the broadcast TV colour system, and the **CMY** colour model is used with some colour-printing devices. Unfortunately, none of these models are particularly easy to use comparing with human perception. According to human intuitive colour concepts, it is easy to describe the colour in terms of shade, tint, and tone, or hue, saturation, and brightness. Colour models which attempt to describe colours in this way include **HSV, HLS, CIEL*a*b*, CIEL*C*H*, CIEL*u*v***. [For98] [Gur98] [Jan89] [Poynt]

2.4.1 RGB colour Model

Based on the tri-stimulus theory of the vision of human eyes, the **RGB** (short for "red, green, and blue") colour model describes colours as positive combinations of three appropriately defined red, green, and blue primaries in a Cartesian coordinate system; this is an example of an *additive colour model*. It can be represented with a unit cube defined on **R**, **G**, and **B** axes, as shown in Figure 2-7 [Ado00]. The origin represents black, and the vertex with coordinates (1,1,1) is white. The main diagonal of the cube, which is the line from origin to point (1,1,1), with equal amounts of each primary, represents the gray levels. [Fai98] [Hea94]

As with the **XYZ** colour system, the **RGB** colour model is an additive system. Intensities of the primary colour are added to produce other colours. Each colour point within the cube can be represented as the triple values (**R**, **G**, **B**), which are assigned in the range from 0 to 1. [Fai98] [Gur98]

The colour gamut covered by the **RGB** model is defined by the chromaticity of a CRT's phosphors. Two CRTs with different phosphors will cover different gamuts. To convert colours with different gamut of the CRT, we can convert **RGB** colours with different colour gamut to **XYZ** colours. [Fai98] [[Fol94]

2.4.2 CMY colour model

The **CMY** colour model is similar with the **RGB** colour model, which uses cyan, magenta, and yellow instead of red, green, and blue, respectively. Figure 2-7 [Ado00] shows a unit cube representing the **CMY** colour model. In the **CMY** colour model, the origin represents the white, and point (1,1,1) represents black. [Frequ] [Sid95]

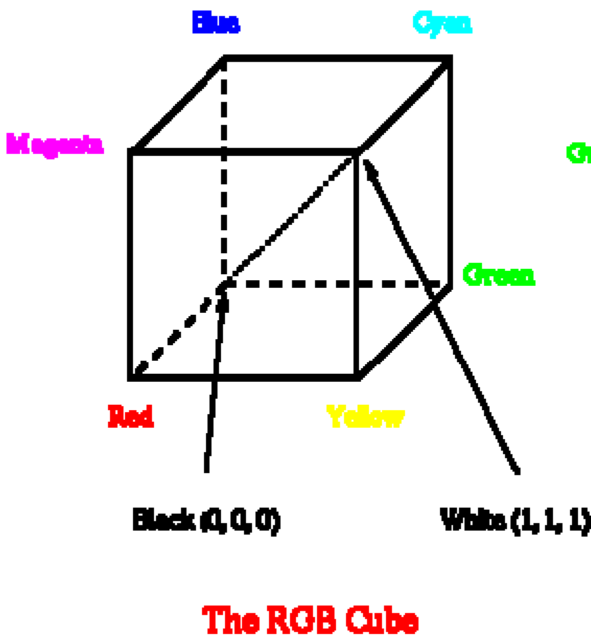


Figure 2-7 The RGB cube

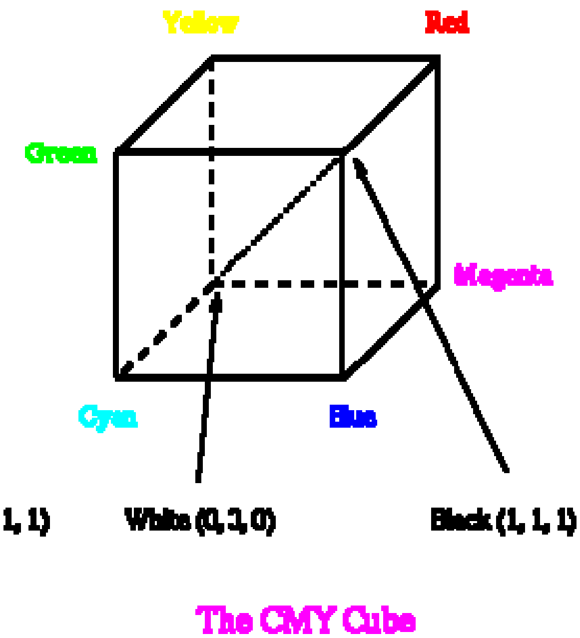


Figure 2-8 The CMY cube

The **CMY** colour model is a *subtractive* system in contrast to the additive system of the **RGB** colour model. Colours are specified by what is removed or subtracted from white light, rather than by what is added to blackness. Printing devices often use the **CMY** colour model to deposit coloured pigments onto paper. For instance, cyan is blue plus green in the **RGB** colour model; it is white minus red in the **CMY** colour model. Similarly, magenta absorbs green, so it is red plus blue; yellow absorbs blue, so it is red plus green. A surface with cyan, magenta, and yellow absorbs red, blue, and green and therefore is black. The transformation between **RGB** and **CYM** colour model is presented by the following equation: [Frequ] [Hun95]

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix} \tag{2-6}$$

Since additive color models display color as a result of light being transmitted (added) the total absence of light would be perceived as black. Subtractive color models display color as a result of light being absorbed (subtracted) by the printing inks. As more ink is added, less and less light is reflected. Where there is a total absence of ink the resulting light being reflected (from a white surface) would be perceived as white. [Fol96]

2.4.3 YIQ colour model

The **YIQ** model is used in U.S.A. commercial colour television broadcasting and is closely related to colour raster graphics, which is suited to monochrome as well as colour CRT display historically. The parameter **Y** is *luminance*, which is the same as in the **XYZ** model. Parameters **I** and **Q** are chromaticity, with **I** containing orange-cyan hue information, and **Q** containing green-magenta hue information. There are two peculiarities with the **YIQ** colour model. The first is that this system is more sensitive to changes in luminance than to changes in chromaticity; the second is that colour gamut is quite small, it can be specified adequately with one rather than two colour dimensions. These properties are very convenient for the transfer of TV signals. [For98] [Gur98] [Poynt]

2.4.4 HSV colour model

The **RGB**, **CMY**, and **YIQ** colour models are hardware-oriented. [Fairc] These do not provide an intuitive method to reproduce the colours according to human vision. For a specified colour, people prefer to use tint, shade, and tone to describe a colour. The **HSV** colour model is a colour model defined to describe the colours similarly to human vision. The three parameters of this model are hue (H), saturation (S), and value (V). [Gon02] [Wys67]

The **HSV** colour model can be derived from the **RGB** cube. By looking along the diagonal of the **RGB** cube, which is from origin to (1,1,1), a hexagonal cone is seen from the outline of the cube. (See Figure 2-9) [Ado00] The boundary of the hexagon represents the various hues, the saturation is measured along a horizontal axis, and value is along a vertical axis through the contre of the hexcone. The colour wheel is varied same as the human perception. [Fol96] [Gon02]

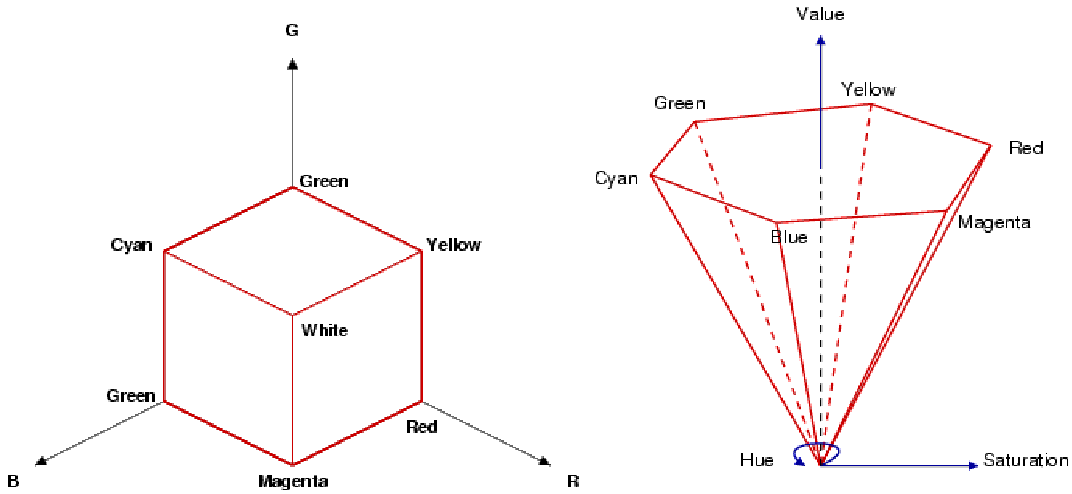


Figure 2-9 The HSV hexcone

Hue is represented by the angle around the vertical axis, with starting red at 0°, then yellow, green, cyan, blue, and magenta respectively, each interval is 60°. Any two colours with 180° difference are complementary colours. Saturation (S) varies from 0 to 1. It is the fraction of distance from center to edge of hexcone. At the S = 0, it is the grey scale. Value (V) varies from 0 to 1 at the top. At the origin, it represents black; and at the top of the hexcone, colours have their maximum intensity. As S =1, the colours have the pure hues. [Fai98] [For98] [Poynt] [Son99]

The **HSV** model is very intuitive for most users. [For98] Comparing with the painting of artists, different colours can be selected from different angles (Hue). Decreasing the value V from a pure colour means adding black. Adding the white to a pure colour produces the value of saturation. **HSV** is also important in that it separates the colour information (hue and saturation) from its brightness (intensity). There are several other names for the **HSV** model but which have the same meaning, such as **HSB** (Hue, saturation, and brightness) and HLS (Hue, lightness, and saturation). [For98] [Fol96] [Woo99]

2.4.5 CIEL*a*b* colour model

CIEL*a*b* (or **CIELAB**) is another colour model that separates the colour information in ways that correspond to the human visual system. [Rus99] It is based on the **CIEXYZ** colour model and was adopted by CIE in 1976. **CIEL*a*b*** is an *opponent colour system* (no colour can involve the opponent colours at same time) based on the earlier (1942) system of Richard Hunter called **L**, **a**, **b**. [Ado00]

CIEL*a*b* defines ***L**** as lightness; ***a**** and ***b**** are defined as the colour axes to describe the hue and saturation. The colour axes are based on the fact that a colour can't be red and green, or both blue and yellow, because these colours oppose each other. The ***a**** axis runs from red (**+** ***a***) to green (**-** ***a***) and the ***b**** axis from yellow (**+** ***b***) to blue (**-** ***b***). (See Figure 2-10) [Xri01] [Xri02] The hue values do not have the same angular distribution in **CIEL*a*b*** colour model as the hue value in **HSV**. In fact, **CIEL*a*b*** is intended to mimic the logarithmic response of the human eye. [For98] The **CIEL*a*b*** overcomes the limitations of colour gamut in the CIE chromaticity diagrams. However, in order to convert to other colour models, ***L**** is defined form 0 (black) to 100 (white), ***a**** is from –100 (green) to 100 (red), ***b**** is from –100 (blue) to 100 (yellow). **CIEL*C*H*** has the same definition with the **CIEL*a*b*** except that its values are defined in a polar coordinate system. Thus in **CIEL*C*H***, **L*** measures brightness, **C*** measures saturation and **H*** measures hue. We will use this model instead of **HSV**, as **CIEL*C*H*** is based on **CIEL*a*b*** and not on **RGB**, and hence is device-independent. [Ado00] [Bra98]

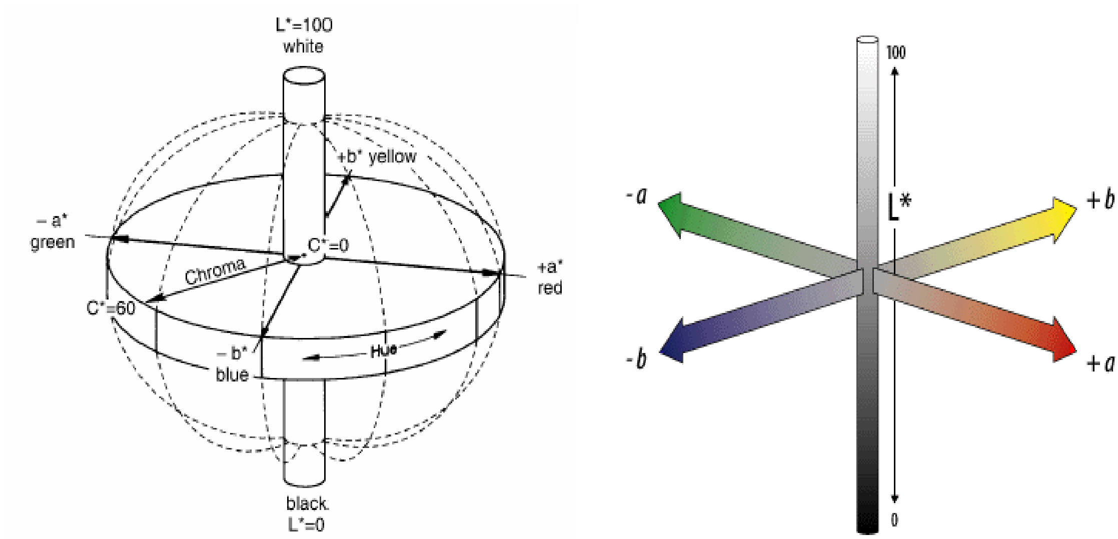


Figure 2-10 The CIEL*a*b* colour model

The colour models which are used in computer graphics, have been traditionally designed for specific devices, such as **RGB** colour model for CRT displays and **CMY** colour model for printers. They are device dependent. Therefore, it becomes meaningless to compare the colours with different devices or the same device under different conditions. [For98] **CIEL*a*b*** is a device independent colour model, and is used for colour management as the device independent model of the ICC (International Color Consortium) device profiles [Ado00]. More details of colour difference formulae are introduced in [Jai89].

2.5 Colour difference and colour tolerance

Since each person accepts or rejects colour matches based on their own colour perception experience, colour management can be confusing and frustrated when dealing with different customers, suppliers, vendors, and productions. [Xri01] [Xri02] To aid in colour decisions, the measurement of *colour difference* and *colour tolerance* (colour tolerance is the minimum acceptable colour matching) can be helpful. Colour difference is a measurement of the numerical differences between colour specifications. The perception of colour differences in **XYZ** and **RGB** colour models is highly non-uniform. [Poynt]

Since the **CIEL*a*b*** colour model was introduced in 1976, many different colour difference formulae have been refined based on **CIEL*a*b*.** [Wes00]

Colour difference in **CIEL*a*b*** is based on the Euclidean distance between two colours represented in the CIE 1976 **CIEL*a*b*** uniform colour space. [Wes00] Its definition is given by:

$$\Delta E^* = \sqrt{\Delta L^{*2} + \Delta a^{*2} + \Delta b^{*2}} \qquad (2\text{-}7)$$

where, ΔL^* is the difference in lightness/darkness value.

Δa^* is the difference on the red/green axis.

Δb^* is the difference of the yellow/blue axis

The colour difference is a rectangle in the **CIEL*a*b***. However, human are more sensitive to changes in brightness (**L***) than to changes in **a*** and **b***; human colour tolerance is thus best described as an ellipsoid in the colour model. [Xri01] [Xri02] In order to reduce the errors obtained from measuring colour differences, CIE introduced some new colour difference formulae to mimic human colour tolerance, such as **CIEL*C*H***, **CIE94**. Tolerance concerns the question "What is set of colors that are imperceptibly or acceptably close to a given reference?" If the distance measure is perceptually uniform, then the answer is simply "the set of points whose distance to the reference is less than the just-noticeable-difference (JND) threshold." This requires a perceptually uniform metric in order for the threshold to be constant throughout the gamut (range of colors). Otherwise, the threshold will be a function of the reference color—useless as an objective, practical guide.The tolerance of **CIEL*C*H*** is a wedge-shape and **CIE94** produce an ellipsoid tolerance. **CIE94** is a tolerance system rather than a colour models. It is based on **CIEL*C*H*** and provides better agreement between visual assessment and measured colour difference and was released by the CIE in 1994. Though no colour tolerance system is perfect, the **CIE94** equations best define colour differences similarly to human perception. Table 2-2 shows the statistical results of the colour tolerance of human perception. [Xri01] Moreover, **CIE94** is targeted for use in the paint and coatings industry. Therefore, for smooth and regular surfaces, **CIE94** may be the best choice. For CRT monitors, is the good choice to measure the colour difference. [Gri02] [Tho00] [Xri02] However, the definition of **CIE94** did not adequately resolve the perceptual uniformity issue, the CIE made **CIECAM97** and **CIEDE2000** at refining their definition, making it even more complicated. Note that the weights depend on the lightness in order to compensate for the non-uniform perception of lightness. [Luo01] [Raj02]

Tolerance Method	Agreement with Human Vision
CIEL*a*b*	75 %
CIEL*C*H*	85 %
CIE94	95 %

Table 2-2 The agreement of human visual in different tolerance method

2.6 Just Noticeable Differences

In image processing, just noticeable difference (JND) is the amount of lights that needs to be added so that the intensity of a pixel can be discriminated from the background intensity [Sut03]. It is also known as the *difference limen* or the *differential threshold*. It can be measured by:

$$\frac{\Delta I}{I} = k \qquad (2\text{-}8)$$

where I is the original intensity of image, ΔI is the addition to it required for the difference to be perceived (the JND), and k is a constant. From the characteristics of the HVS, the JND profile of a still image depends on the spatial frequency sensitivity, the sensitivity to light and masking effects. The spatial frequency sensitivity is generally modelled as the so-called modulation transfer function (MTF) which provides relative tolerance of the HVS to noise at different spatial frequencies. The sensitivity of light is dependence of the sensitivity on the local luminance. In general, high visibility thresholds will occur in either very dark or very bright regions, and lower thresholds will occur int the regions of gray levels close to the mid-gray intensity. [Buc03] [Sut03]

2.7 Summary

So far, there is no single colour model that is flawless, or can solve all problems. All colour models are defined to be applied in some specified situation. Also there are still no international standards in industry for colour application. CIE proposed a new colour model, which is **CIECAM97s** (The CIE 1997 Interim Color Appearance Model (Simple Version)), for device independent colour imaging applications in 1997. **CIECAM97s** has been successful in focusing researchers and practitioners in colour science and colour imaging on a single colour appearance model. However, it is not widely used for practical applications, because of its complexity. CIE is currently researching some new colour models that are both more powerful and simple to use. [Fai01] [Fairc] [Lic00]

Chapter 3

Colour displaying and measurement

3.1 Introduction

Image files are stored, manipulated, and transmitted to produce pictures or images. There are more than one hundred kinds of image formats used for different applications. [Bro95] However, most image formats are not normally used. There are several image formats that are widely applied, including BMP, GIF, TIFF, PNG, and JPEG. (These will be discussed below). Although the images are created by different hardware, for example digital cameras, scanners, video recorders, or transformed by software, basically, they always have the same two kinds of information: image sizes and colours (intensity value for black-white image). [Kay92]

Also different equipment may be used to display a specified image, for instance, television, CRT monitors, and LCD (*liquid-crystal display*) monitors. Different equipment could result in different appearance for the same image because of the different resolutions and different colour models and colour gamuts. Since digital images are mainly applied and processed by the computer, we only focus the image displayed on computer monitors using *cathode-ray tube* (abbreviated CRT) technology. [Bro95] [Fol96]

3.2 Image formats

Image files may have different formats for different purposes. Different image formats may require different methods to store, transmit, and manipulate their images. For computer graphics and Internet users, most images are stored as *Raster Image Formats* (RIFs) [Burns] The quality of image is determined to two factors as the image is created: *sampling* and *quantization*. Digitizing the coordinate value of the image is called sampling; digitizing the amplitude values of image is called quantization. A Raster Image Format breaks the image into a series of pixels determined by the sampling; and the number of bits used to define the value of each pixel is called colour depth of the image, and is determined by the quantization. [Bro95]

A greyscale image only has intensity values defined for each pixel, and can thus only display different intensities of grey. The numbers of bits to describe the intensity determine the possible number of the grey scales for the image. The more bits used to define one pixel's intensity, moer details the images have. Colour images contain more information than greyscale images. Since CRT monitors use the **RGB** colour model, most colour image formats store red, green, and blue values in each pixel. The numbers of bits used to store the information of red, green, and blue determine the maximum possble number of colours for the image. Table 3-1 shows the relationship between the numbers of bits and numbers of colours. As the table 3-1 shown, 8-bits for each red, green, blue value in **RGB** colour model can provide more than 16 million different colours. That means that each pixel uses 24 bits to store the colour information. Such an image has enough colours for human perception and interpretation. Therefore, all image files in this thesis use a 24-bit colour image format. [Bro95] [Kay92]

Bits per pixel	Numbers of Colour
3	8
4	16
8	256
15	32768
24	16,777,216
n	2^n

Table 3-1 Number of levels represented by numbers of bits

The 24-bits colour image format of Microsoft BMP (BitMaP) and JPEG (Joint Photographic Experts Group) are the main raster image formats using in computer graphics and Internet. [Burns] A BMP file is an uncompressed image file, and JPEG involves compression. Hence, the size of JPEG file is much smaller than BMP file, but the image quality of BMP can sometimes be better than JPEG. The quality of JPEG is generally depended on the compression factor. [Gon02] Another factor which influences the image quality is the number of pixels which is determined by the sampling. The concept of the image pixel is same as the pixel of CRT display equipment. Pixels are arranged in a rectangle and they are too small to distinguish individually. The more pixels an image has, the more details the image contains. Also the size of image file is larger. [Burns] [Kay92]

3.3 CRT monitor

A CRT (Cathode-ray tube) has three electron guns and phosphor dots arranged in a triangular pattern, or three electron guns in a line. [Fol96] There are three types of phosphor dot for each pixel, and each electron gun can shoot an electron beam to hit only one type of phosphor dot. When the electrons of different guns hit the screen, the phosphor dots emit visible light with different colours. Each individual group of phosphor dot with different red, green, and blue is small enough to be discerned as a single colour. This uses **RGB** colour model to add the colours. The stream of electrons from the heated cathode is accelerated toward the phosphor dots by a high voltage, which determines the velocity obtained by the electrons before they hit the phosphor dot. Controlling the voltage determines how many electrons are actually in the electron beam. The more negative the voltage, the fewer electrons are in the beams. Therefore, the intensity of each colour can be controlled by the voltage. A certain colour that is produced by a particular voltage on one system could be different on any other system. Different phosphors used on the surface of CRT will generate different light spectra when voltages are applied to them. More details for CRT displays are introduced in [Fol96]. [Bro95] [Mel99]

The human eye is not sensitive to the exact amount of change between two intensities of light, only the ratio of change. [Bro95] For example, the intensity from 0.1 to 0.2 and from 0.2 to 0.4 will be perceived as an equal amount of intensity change because both have ratio of 2. However, the intensities on a CRT monitor are non-linear comparing with human vision. [Bro95] The intensity is generally modeled using a power function as below: [For98]

$$Intensity = voltage^{\gamma} \tag{3-1}$$

where intensity and voltage are normalized between 0 and 1. In order to approximate the non-linear intensity values, the CRT's intensity is modeled using the equation below: [Bro95]

$$I = I_0 + cv\gamma \tag{3-2}$$

The value I_0 is the lowest possible intensity value of the CRT in working, which is typically not exactly zero. The value c depends on the CRT, and v is the amount of energy for a given intensity. When two display systems have different γ values, they will produce different colour from the same image files. If the different γ values are known, the intensities can be modified to compensate the difference. This is called *gamma correction* as the γ information is used to correct for colour intensity. [Bro95] [For98] [Poynt]

3.4 sRGB colour model

In order to avoid the colour difference with different display systems, the IEC (International Electrotechnical Commission) introduced *sRGB* (IEC 61966-2-1) as a standard colour model solution for office, home and web markets. The sRGB model serves the needs of PC and Web based colour imaging systems and is based on the average performance of CRT displays. The **sRGB** solution is supported by the following observations: [Gon00] [IEC98] [IEC99]

- Most computer displays are similar in their phosphor chromaticities (primaries) and transfer function
- The **RGB** colour model is native to CRT displays, scanners and digital cameras, which are the devices with the highest performance constraints
- The **RGB** colour model can be made device independent in a straightforward way. It is also possible to describe colour gamuts that are large enough for all but a small number of applications.

sRGB Viewing Environment Conditions	
Condition	sRGB
Display luminance level	80 cd/m^2
Display white point	$x = 0.3127$, $y = 0.3290$ (D65) (See Fig 2-5)
Display model offset (R, G, and B)	0.0
Display input/output characteristic	2.2
Reference ambient illuminance level	64 lux
Reference ambient white point	$x = 0.3457$, $y = 0.3585$ (D50) (See Fig 2-5)
Reference veiling glare	1.0 %
Reference background	For the background as part of the display screen, the background is 20% of the reference display luminance level
Reference surround	20 % reflectance of the reference ambient

Table 3-2 sRGB viewing environment conditions

CIE chromaticities for ITU-R BT.709 reference primaries and CIE standard illuminant				
	Red	Green	Blue	D65
x	0.6400	0.3000	0.1500	0.3127
y	0.3300	0.6000	0.0600	0.3290
z	0.0300	0.1000	0.7900	0.3583

Table 3-3 CIE chromaticities for ITU-R BT.709 reference primaries and CIE standard illuminant (D65 white is supposed to be the colour of the sun)

The IEC listed some viewing reference conditions. (See table 3-2)
The expected colours of the red, green, and blue monitor phosphors and the white setting of an **sRGB** monitor are specified in chromaticity values as table 3-3. The non-linearity or gamma of the monitor is 2.2 [Gon00] [IEC98] [IEC99]

3.5 Colour measurement

As humans measure colours, there are many factors which affect perception; these include the background, light intensity, the shape of object, observing angle, and the reflecting light. [Hun95] In psychophysics, colours can described by three main terms: *dominant wavelength*, *excitation purity*, and *luminance*. [Fol96] Dominant wavelength is the wavelength of the colour that is seen in the visible light and corresponds to the hue. Excitation purity corresponds to the saturation of the colour; that is, the proportion of pure light of the dominant wavelength. (See chapter 2.2, 2.3) A completely pure colour is 100 percent saturated and thus contains no white light. Luminance is the amount or intensity of light. [Hea94] Artists often use tint, shade, and tone to create different colours. The tint means adding white pigment to a pure pigment to decrease saturation. The shade means adding black pigment to a pure pigment to decrease lightness. The tone adds both black and white pigments to a pure pigment. It can produce different colours of the same hue with varying saturation and lightness. (See Figure 3-1) [Fol96]

The **RGB** colour model is popular, being applied in computer graphics and video systems, and most digital images are stored using the **RGB** colour model. However, although the **RGB** colour model is very convenient to be applied in CRT display system, it is quite difficult to describe the colour according to human perception. However, humans normally describe the colour by brightness, saturation, and hue (hue is the most important factor in colour measuring), [Wys67] such as **HSV** colour model and **CIEL*a*b*** colour model. Table 3-4 shows seven colours with different values in **RGB**, **HSV**, and

CIEL*a*b* colour model respectively. Their features with different colour models will be analysed shortly.

3.5.1 RGB colour measurement

As we have seen, **RGB** is an additive colour system. For example, the combination of same value of red and blue produces magenta; red and green produces yellow; and blue and green produces cyan. Equal values of red, green, and blue produce grey. If the three values are different, the hue is determined by which of red, green and blue is largest. The brightness is determined by the largest value and the saturation is determined by the smallest value. As the Table 3-4 shows, colour 1 is a pure blue colour because the components of red and green are zero. Colour 2 is a pure green colour because the value of red and blue are zero. Colour 3 is similar to colour 2 in that the red and blue components are zero, but here the green value of colour 3 is half of colour 2's. That means that although colour 3 is the same pure green colour as colour 2, the brightness is just half of colour 2. The RGB values of colour 4 are [127, 255, 127]. Since the RGB is an additive colour model, colour 4 can be considered to be the sum of two RGB values, [0, 127, 0] and [127, 127, 127]. The colour [127, 127, 127] is grey which is in the middle of black and white and [0, 127, 0] is same as the colour 3. The hue of colour 4 is still same as colour 3 because [127, 127, 127] is colourless. However, there are red and blue values in the grey light, and the green is not pure as in colour 3. Thus colour 4 would appear as a green colour washed by grey light, with saturation only half of colour 3. [Bri02] [Bro95] [Cui00] [Hea94][Hun95]

So far, it seems that the **RGB** colour model can represent colours very well. However, the four colours 1, 2, 3, and 4 at the above case are very special. The **RGB** colour model can produce more than 16 million different colours, and so colours which can be easily described are very few. For more general colours, for example, the colour 5, 6, and 7, it is not easy to describe their colour properties. Colour 5 has the largest red value, so its hue is close to red rather than green and blue; and since its red value is 210, it should be quite bright. Also its smallest value is 85, so it is not a pure colour. There are similar descriptions for the colours 6 and 7. However, these descriptions are not exact enough to describe the colour difference for similar colours. In order to describe colour more understandably, a colour models must be chosen to match human intuition. [Hun95]

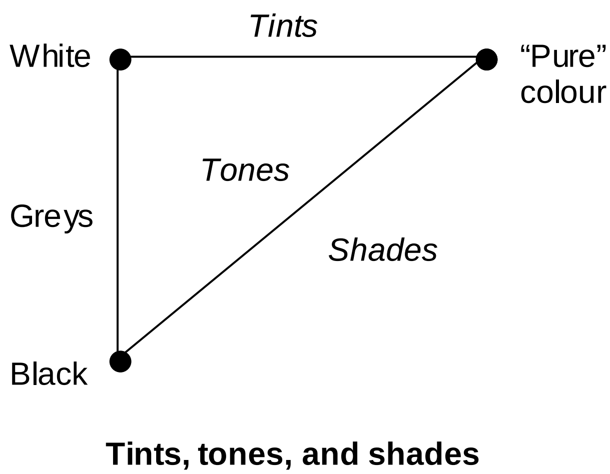


Figure 3-1 The relationship of tints, tones, and shades

Colour	1	2	3	4	5	6	7
Red	0	0	0	127	210	187	103
Green	0	255	127	255	180	227	144
Blue	255	0	0	127	85	105	20
H	0.667	0.333	0.333	0.333	0.127	0.221	0.222
S	1	1	1	0.5	0.60	0.537	0.861
V	1	1	0.5	1	0.823	0.890	0.565
CIE L*	32.30	87.74	45.88	90.58	74.19	85.26	54.95
CIE a*	79.19	-86.19	-51.41	-60.29	-0.390	-31.48	-31.52
CIE b*	-107.85	83.19	49.62	50.10	51.48	54.56	54.48
CIE C*	133.81	119.78	71.45	70.39	51.48	62.99	62.94
CIE H*	-0.94	2.37	2.37	2.49	1.58	2.09	2.10

Table 3-4 Colour values in different colour models

3.5.2 HSV colour measurement

The **HSV** colour model uses three primary values, based on human perception, to represent the colour. Colour 1 in table 3-4 is represented that it has the maximum values of intensity and saturation and its hue is presented at 240 degree in a circle hue panel, which is blue. As we have seen in figure 2-9, the hue value in **HSV** is defined from 0 to 1, which may be considered as the ratio from 0 to 360 degree in a circle. Colours 2, 3, and 4 have the same hue value (0.333), but different saturation and brightness. Colour 3 is darker than colour 2 and 4; and colour 4 is not a pure colour, as colours 2 and 3. **HSV** colour model provides exactly same information as the **RGB** colour model but using different description.

In the different colour model, the quantisation of acceptable different of colour is not uniform. The advantage of the **HSV** colour model is not only to give more details for general colours, but also to represent exactly the colour difference according to humans' perception. The hue of colour 5 is 0.127, so this colour is located in the colour between orange and yellow in the hue circle. Its brightness is 0.823; and its saturation is 0.60, which means that this colour is plain compared with the pure colours. Colours 6 and 7 are hard to distinguish their difference exactly using the **RGB** colour model. It is quite clear to distinguish them in the **HSV** colour model. Since their hue values are 0.221 and 0.222 respectively, both colours have almost the same colour appearance in the visible spectrum. Colour 6 is brighter than colour 7 because its intensity value is bigger than colour 7. Although both colours are not pure, colour 7 is more colourful because its saturation value is larger. In short, the **HSV** model gives a more useful description for representing colours comparing with the **RGB** colour model; furthermore it has specified values to definitely compare what the colour differences are. However, the values of **HSV** have some disadvantages compared to human perception. Since the **HSV** colour model is directly transformed from the **RGB** cube and its values are based on the **RGB** values, in fact, its values correspond the changes of the CRT display rather than human perception. The best colour model to describe colours according to human perception is **CIEL*a*b*** (or **CIELAB**), which is based on the **CIE XYZ** model. [Fai98] [Fol96] [For98]

3.5.3 CIEL*a*b* colour measurement

Colours 1 and 2 of table 3-4 have same brightness and saturation, only their hues differ, being blue and green respectively. However, the human eye has different intensity perception with different dominant wavelength. As the Figure 2-3 shows, green colours normally appear brighter than blue colours with the same values of brightness and saturation. According to the definition of **CIEL*a*b*,** ***L*** represents the brightness of the colour. Therefore the brightness of colour 2 (Table 3-4) is much stronger than colour 1, and this result is same as the human perception.

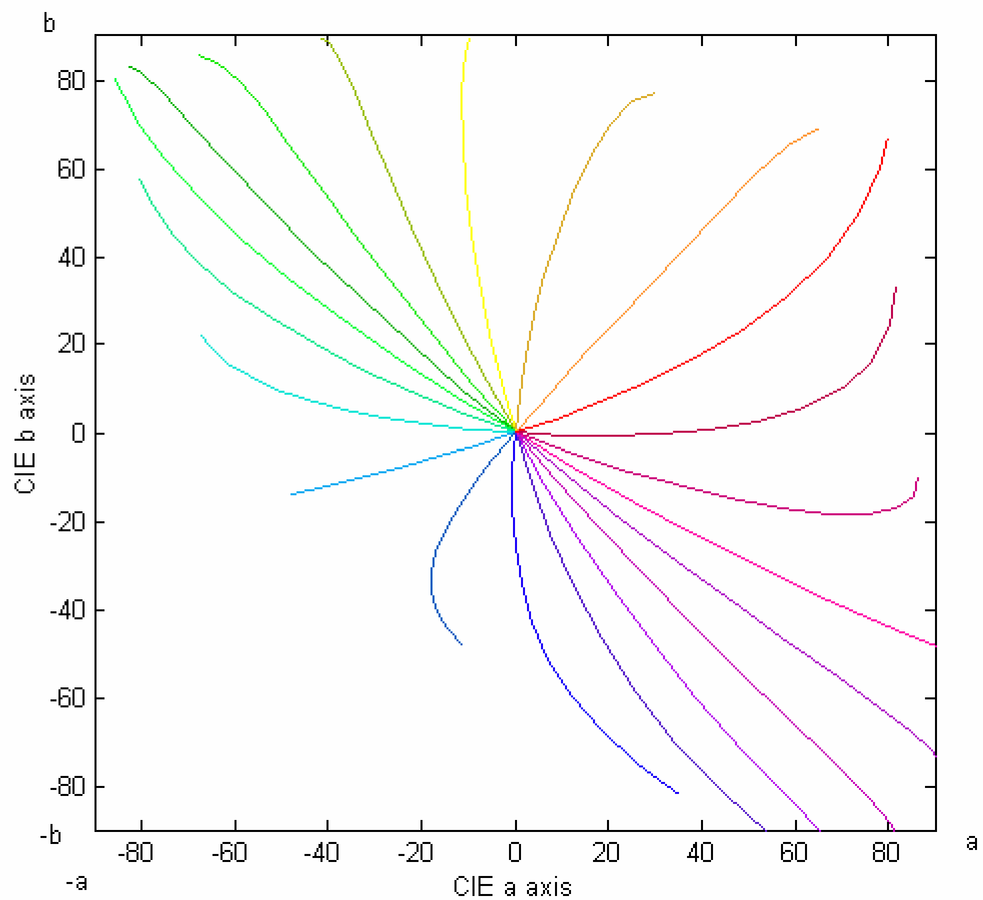


Figure 3-2 Colours of HSV with constant hue in CIEL*a*b*

The hue of **CIEL*a*b*** has the same definition as ***H*** in **HSV**. However, the hue of **CIEL*a*b*** value is non-linear, which means ***CIE H*** value is not constant for the same dominant wavelength as the brightness and saturation varies. In Figure 3-2, twenty curves are plotted with constant brightness, and hue and saturation varying for twenty different colours. As the figure shows, each individual colour has the different hue values as the saturation varies. This means that humans do not have the same hue perception for light with the same frequency as the saturation is varying. In fact, the **HSV** colour model represents the hue according to the physical properties of light, and the **CIEL*a*b*** or **CIEL*C*H*** represents the hue varying acccording to human perception. [Bra98] [Kue98] More the JND for green-yellow colour is referred in [Buc03] [Sut03].

3.6 Conclusion

In conclusion, different colour models are used for different purposes. There is no single colour model which can reproduce the colour and represent the colour differences perfectly under all conditions. **CIE XYZ** is the foundation of colour science and **CIEL*a*b*** is the best model to represent colour difference so far. Many advanced colour models and formulae are based on **CIEL*a*b***, such as **CIEL*C*H***, **CIE94**, **CIECAM97s**, and **CIEDE2000**. [Mel00]

Chapter 4

Colour model conversions

4.1 Introduction

As we have discussed in the previous chapter, colour models are designed to reproduce colours or differentiate colours for different purposes. Typically, there are two kinds of colour models: device-dependent and device-independent. [Poynt] **RGB** and **CMY** are typical device-dependent colour model because their colour reproduction are determined by the output device. For example, the different phosphors of monitors and different inks of printers require the use of colour models with different colour gamuts. Since IEC has standardized the **sRGB** colour model used in CRT displays, any monitor using this standard can be considered to be device-independent. Therefore, the values of different colour models can be transformed to each other. [IEC98] [IEC99]

4.2 Conversion between RGB and HSV

The **HSV** colour model can be considered as a different view of the **RGB** cube. Hence the values of **HSV** can be considered as a transformation from **RGB** using geometric methods. (See Figure 2-9) The diagonal of the **RGB** cube from black (the origin) to white corresponds to the ***V*** axis of the hexcone in the **HSV** model. For any set of **RGB** values, ***V*** is equal to the maximum value in this set. The **HSV** point corresponding to the set of **RGB** values lies on the hexagonal cross section at value ***V***. The parameter ***S*** is then determined as the relative distance of this point from the V axis. The parameter ***H*** is determined by calculating the relative position of the point within each sextant of the hexagon. The values of **RGB** are defined in the range [0, 1], the same value range as **HSV**. The value ***H*** is the ratio converted from 0 to 360 degree. The algorithm of the conversion is as below: [Fol96] [Hea94]

Find the maximum and minimum values from the **RGB** triplet. The saturation, ***S***, is then:

$$S = \frac{(\max - \min)}{\max} \tag{4-1}$$

The Value, ***V***, is

$$V = \max \tag{4-2}$$

The Hue, ***H***, is calculated as follows. First calculate R'G'B'

$$R' = \frac{\max - R}{\max - \min}$$

$$G' = \frac{\max - G}{\max - \min} \tag{4-3}$$

$$B' = \frac{\max - B}{\max - \min}$$

If saturation, ***S***, is zero then hue is undefined, which means that the colour has no hue (it is a grey value), otherwise:

if R = max and G= min

$$H = 5 + B' \tag{4-4}$$

else if R = max and $G \neq$ min

$$H = 1 - G' \tag{4-5}$$

else if G = max and B = min

$$H = R' + 1 \tag{4-6}$$

else if G = max and $B \neq$ min

$$H = 3 - B' \tag{4-7}$$

else if R = min

$$H = 3 + G' \tag{4-8}$$

otherwise

$$H = 5 - R' \tag{4-9}$$

The numbers *1*, *3*, and *5* in the formulae from (4-4) to (4-9) are the ratios of degree of 360˚, for which *1* is equal to 60˚, *3* is equal to 180˚, and *5* is equal to 300˚. To convert back from **HSV** to **RGB** is as below: [Fol96] [Hea94]

Firstly, there are some variables must be defined as follows:

The *primary colour*, which is the integer component of the value Hue

$$secondary\ colour = Hue - primary\ colour \tag{4-10}$$

$$a = (1 - S)\ V \tag{4-11}$$

$$b = (1 - (S * secondary\ colour))\ V \tag{4-12}$$

$$c = (1 - (S * (1 - secondary\ colour)))\ V \tag{4-13}$$

Then, the **RGB** values can be calculated:

If *primary colour* = 0 then

$$R = V,\ G = c,\ B = a \tag{4-14}$$

If *primary colour* = 1 then

$$R = b,\ G = V,\ B = a \tag{4-15}$$

If *primary colour* = 2 then

$$R = a,\ G = V,\ B = c \tag{4-16}$$

If *primary colour* = 3 then

$$R = a,\ G = b,\ B = V \tag{4-17}$$

If *primary colour* = 4 then

$$R = c,\ G = a,\ B = V \tag{4-18}$$

If *primary colour* = 5 then

$$R = V,\ G = a,\ B = b \tag{4-19}$$

4.3 Conversion between RGB and CIE XYZ

There are two **RGB** colour model values between image files and the phosphor dots of a monitor. The digital image files store 8-bit linear **RGB** values and the phosphor dots of a monitor represents non-linear **sRGB** value. There are two steps to convert **RGB** to **CIE XYZ** values: [IEC98] [IEC99]

Step 1: Convert linear **RGB** value to non-linear **sRGB** value

$$
\begin{aligned}
R'_{sRGB} &= R_{8bit} \div 255.0 \\
G'_{sRGB} &= G_{8bit} \div 255.0 \\
B'_{sRGB} &= B_{8bit} \div 255.0
\end{aligned}
\tag{4-20}
$$

if R'_{sRGB} , G'_{sRGB} , B'_{sRGB} ≤ 0.04045

$$
\begin{aligned}
R_{sRGB} &= R'_{sRGB} \div 12.92 \\
G_{sRGB} &= G'_{sRGB} \div 12.92 \\
B_{sRGB} &= B'_{sRGB} \div 12.92
\end{aligned}
\tag{4-21}
$$

else if R'_{sRGB} , G'_{sRGB} , B'_{sRGB} > 0.04045

$$
\begin{aligned}
R_{sRGB} &= \left(\frac{R'_{sRGB} + 0.055}{1.055} \right)^{2.4} \\
G_{sRGB} &= \left(\frac{G'_{sRGB} + 0.055}{1.055} \right)^{2.4} \\
B_{sRGB} &= \left(\frac{B'_{sRGB} + 0.055}{1.055} \right)^{2.4}
\end{aligned}
\tag{4-22}
$$

Step 2: Convert to **CIE XYZ**:

$$
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{D65} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R_{sRGB} \\ G_{sRGB} \\ B_{sRGB} \end{bmatrix}
\tag{4-23}
$$

Converting **CIE XYZ** to **RGB** is as follows:

$$\begin{bmatrix} R_{sRGB} \\ G_{sRGB} \\ B_{sRGB} \end{bmatrix} = \begin{bmatrix} 3.2406 & -1.5372 & -0.4986 \\ -0.9689 & 1.8758 & 0.0415 \\ 0.0557 & -0.2040 & 1.0570 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{D65} \tag{4-24}$$

if R_{sRGB} , G_{sRGB} , $B_{sRGB} \leq 0.0031308$

$$\begin{aligned} R'_{sRGB} &= 12.92 \times R_{sRGB} \\ G'_{sRGB} &= 12.92 \times G_{sRGB} \\ B'_{sRGB} &= 12.92 \times B_{sRGB} \end{aligned} \tag{4-25}$$

else if R_{sRGB} , G_{sRGB} , $B_{sRGB} > 0.0031308$

$$\begin{aligned} R'_{sRGB} &= 1.055 \times R_{sRGB}{}^{(1.0/2.4)} - 0.055 \\ G'_{sRGB} &= 1.055 \times G_{sRGB}{}^{(1.0/2.4)} - 0.055 \\ B'_{sRGB} &= 1.055 \times B_{sRGB}{}^{(1.0/2.4)} - 0.055 \end{aligned} \tag{4-26}$$

The non-linear *sR'G'B'* values are converted to digital code values which are determined by two factors, ***WDC*** and ***KDC***. Shown as below:

$$\begin{aligned} R_{8bit} &= \left((WDC - KDC) \times R'_{sRGB}\right) + KDC \\ G_{8bit} &= \left((WDC - KDC) \times G'_{sRGB}\right) + KDC \\ B_{8bit} &= \left((WDC - KDC) \times B'_{sRGB}\right) + KDC \end{aligned} \tag{4-27}$$

where ***WDC*** represents the white digital count and ***KDC*** represents the black digital count. The IEC standard specifies a black digital count of 0 and a white digital count of 255 for 24-bit image encoding. So the resulting **RGB** values are:

$$\begin{aligned} R_{8bit} &= 255.0 \times R'_{sRGB} \\ G_{8bit} &= 255.0 \times G'_{sRGB} \\ B_{8bit} &= 255.0 \times B'_{sRGB} \end{aligned} \tag{4-28}$$

4.4 Conversions between CIE XYZ and CIEL*a*b*

CIEL*a*b* is based directly on **CIE XYZ** (1931). It is non-linear and is intended to mimic the logarithmic response of the human eye. The transformation is as below: [Hof03]

$$L^* = \begin{cases} 116 \times \left(\frac{Y}{Y_n}\right)^{1/3} - 16 & if \quad \frac{Y}{Y_n} > 0.008856 \\ 903.3 \times \left(\frac{Y}{Y_n}\right) & if \quad \frac{Y}{Y_n} \le 0.008856 \end{cases} \tag{4-29}$$

$$a^* = 500 \times \left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right) \right) \tag{4-30}$$

$$b^* = 200 \times \left(f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right) \right) \tag{4-31}$$

where

$$f(t) = \begin{cases} t^{1/3} & if \quad t > 0.008856 \\ 7.787 \times t + \frac{16}{116} & if \quad t \le 0.008856 \end{cases} \tag{4-32}$$

L* scales from 0 to 100 for relative luminance *(Y / Yn)* scaling 0 to 1.

X_n, Y_n, and Z_n are the values of reference white, which can be obtained from formula (4-23). In the CRT displaying system, the white point is defined as 1 for the each primary value. Therefore, X_n, Y_n, and Z_n can be obtained from formula 4-23 as:

$$\begin{bmatrix} X_n \\ Y_n \\ Z_n \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{4-33}$$

The **CIEL*a*b*** model converts to **XYZ** as the follows: [Hof03]

$$Y' = \frac{L^* + 16}{116} \qquad X' = \frac{a^*}{500} + Y' \qquad Z' = -\frac{b^*}{200} + Y' \tag{4-34}$$

$$X = \begin{cases} X_n \times (X')^3 & if \quad X' > 0.206893 \\ X_n \times \dfrac{X' - 16/116}{7.787} & if \quad X' \leq 0.206893 \end{cases} \tag{4-35}$$

$$Y = \begin{cases} Y_n \times (Y')^3 & if \quad Y' > 0.206893 \\ Y_n \times \dfrac{Y' - 16/116}{7.787} & if \quad Y' \leq 0.206893 \end{cases} \tag{4-36}$$

$$Z = \begin{cases} Z_n \times (Z')^3 & if \quad Z' > 0.206893 \\ Z_n \times \dfrac{Z' - 16/116}{7.787} & if \quad Z' \leq 0.206893 \end{cases} \tag{4-37}$$

4.5 Conversions between CIEL*a*b* and CIEL*C*H*

The conversion between **CIEL*a*b*** and **CIEL*C*H*** is a transformation between Cartesian coordinates and polar coordinates. Polar parameters more closely match the visual perception of colours. Their transformation is listed as below:

$$C^* = \sqrt{a^{*2} + b^{*2}} \tag{4-38}$$

$$H^* = \arctan\left(\frac{b^*}{a^*}\right) \tag{4-39}$$

$$a^* = \cos(H^*) \times C^* \tag{4-40}$$

$$b^* = \sin(H^*) \times C^* \tag{4-41}$$

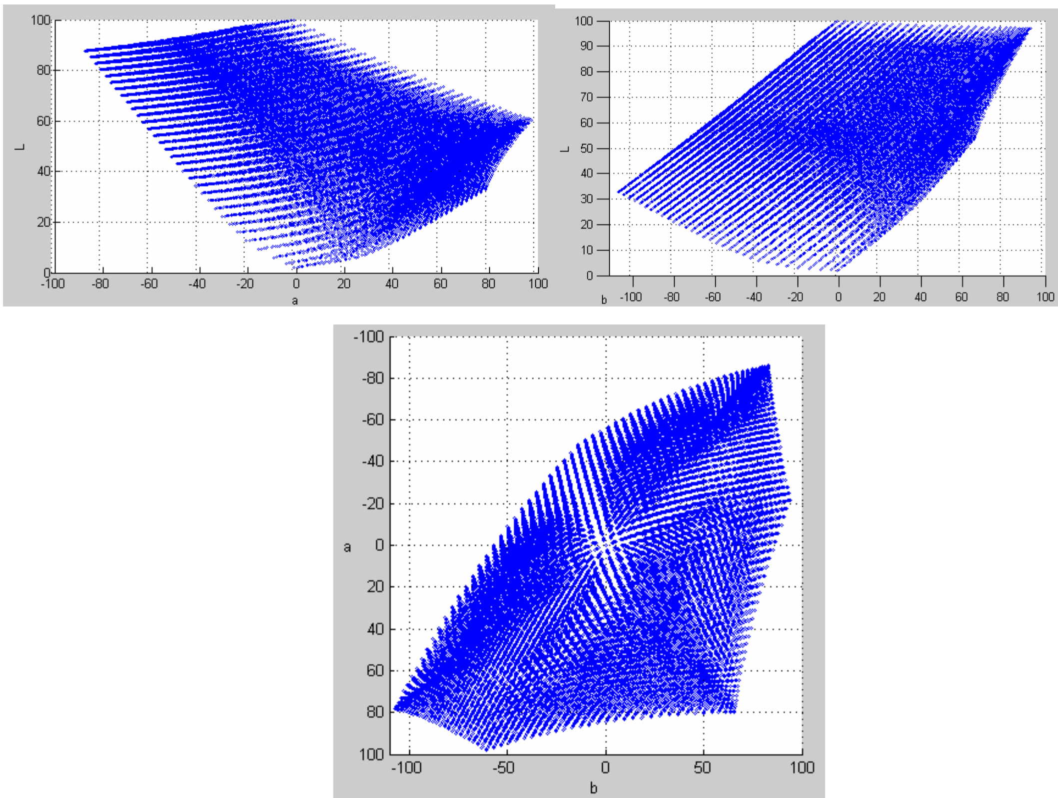


Figure 4-1 Three directions views of RGB in CIEL*a*b*

Different colour models can be converted to each other according to the above algorithms and formulae. Since the colour gamut of **RGB** colour model is part of **CIE XYZ**, and **CIEL*a*b*** has the same colour gamut as **CIE XYZ**, the transformation of **RGB** colour can never fit within the **CIEL*a*b*** coordinate precisely. Figure 4-1 shows three projections of the **RGB** gamut in the **CIEL*a*b*** coordinate system. The upper left shows the relationship with the values ***L**** and ***a****; the upper right figure shows the coordinate with the values ***L**** and ***b****; and the bottom figure represents the values in ***a**** and ***b**** coordinate. Figure 4-2 shows four different three-dimensioned views. Hence, it is clear that as the values are converted from the **CIEL*a*b*** colour model to the 24-bit

RGB colour model, some of the values are out of the **RGB** gamut and truncated to fit it. Thus there could be errors when values are converted from **CIEL*a*b*** to **RGB**.

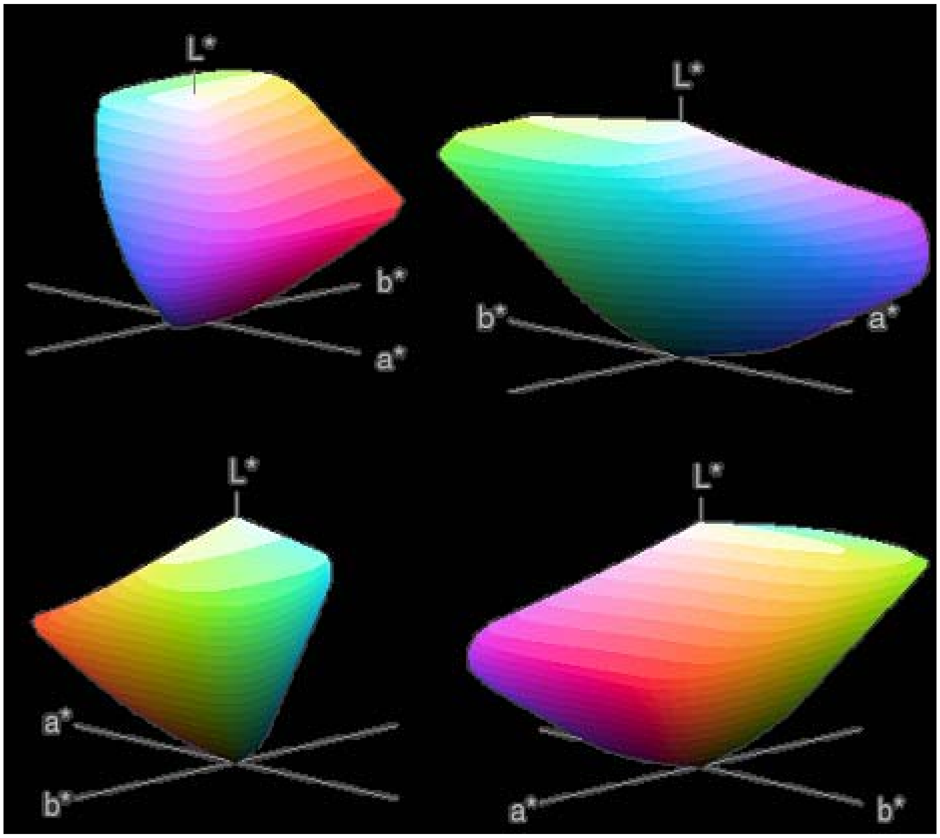


Figure 4-2 RGB colour gamut in CIEL*a*b* [Hof03]

4.6 CIE94 colour difference formula

CIE94 is a colour tolerance system rather than a colour model and it is based on the value of **CIEL*a*b***. Its formula is shown as below: [Gri02] [Luoro]

$$\Delta E_{94} = \sqrt{\left(\frac{\Delta L^*}{K_L S_L}\right)^2 + \left(\frac{\Delta C^*_{ab}}{K_C S_C}\right)^2 + \left(\frac{\Delta H^*_{ab}}{K_H S_H}\right)^2} \qquad (4\text{-}42)$$

where

$$\begin{bmatrix} \Delta L^* \\ \Delta a^* \\ \Delta b^* \end{bmatrix} = \begin{bmatrix} L_1^* - L_2^* \\ a_1^* - a_2^* \\ b_1^* - b_2^* \end{bmatrix} \qquad \begin{bmatrix} C_1^* \\ C_2^* \end{bmatrix} = \begin{bmatrix} \sqrt{a_1^{*2} + b_1^{*2}} \\ \sqrt{a_2^{*2} + b_2^{*2}} \end{bmatrix}$$

$$\overline{C_{ab}}^* = \sqrt{C_1^* C_2^*},$$

$$S_L = 1, \quad S_C = 1 + 0.045\overline{C_{ab}}^*, \quad S_H = 1 + 0.015\overline{C_{ab}}^*$$

$$\Delta C_{ab}^* = C_1^* - C_2^*, \quad \Delta H_{ab}^* = \sqrt{\Delta a^{*2} + \Delta b^{*2} - \Delta C_{ab}^{*2}}$$

The CIE have defined reference conditions using this formula, that is: [Heggi]

1. The specimens are homogeneous in colour.
2. The colour difference (**CIEL*a*b***) is <= 5 units.
3. They are placed in direct edge contact.
4. Each specimen subtends an angle of at least degrees to the assessor, whose colour vision is normal.
5. They are illuminated at 1000 lux, and viewed against a background of uniform grey, with L* of 50, under illumination simulating *D65*. (D65 is the light source that is defined to simulate day-light and has x=0.312727 and y=0.329024 in figure 2-6)

4.7 Conclusion

The perception of colours is quite influenced by many external factors. It is recommended to use a single colour model or colour difference formulae to reproduce or measure colours. Some organizations, such as CIE, IEC, have released and standardized some colour models and formulae to make the transformation among the different models possible. However, no single colour model or colour difference formula is perfect so far. Each model or formula just gives an approximate result compared with human perception. Furthermore, nobody accepts or rejects colours because of numbers, it is the colour's appearance which counts. The final results must be confirmed by human visual judgments. [Hof03] [Xri01] [Xri02]

Chapter 5

Exploration of colour difference in MATLAB

5.1 Introduction

The aim of this chapter is the application of MATLAB to analyse colour differences in digital images with the 24-bit **RGB** colour model, especially for the yellow-green colours, which we have discussed in section 2.3 and 3.5.

MATLAB is very powerful for array operations. Since images are stored as two dimensional arrays, images can be processed very quickly in MATLAB. Moreover, some abstract data can be easily shown in a visual form using MATLAB. According to the algorithms of colour conversion introduced in the last chapter, MATLAB can be used to create images with different colours and using different colour models; these images will be viewed by different people to distinguish the colour differences. The results are used to analyse the relationship between human perception and the varying values in 24-bit **RGB** colour model.

The **RGB** colour model as we have seen can be modeled as a cube. However, the variation of its values does not conform to human perception. [Bro95] If all colours in 24-bit **RGB** colour model are located in the **CIEL*a*b*** colour model with their corresponding values, it is a quite irregular shape. (See Figure 4-1 and Figure 4-2) There is no simple relationship between the 24-bit **RGB** colour model and the **CIEL*a*b*** colour model and human perception as discussed in the last chapter. In this chapter, several experiments are designed to analyse the relationship between human perception and the variation of brightness, saturation, and hue. Ten people are asked to test the colour difference in the each experiment. The maximum and minimum test values are discarded in each experiment. The medium of other eight test results is used to analyse. More details are listed in Appendix B and C. The three dimensions of the 24-bit **RGB** colour model are decomposed to different layers in **CIEL*a*b*** colour model according to the varying brightness, saturation, and hue value to show how the colours are perceived with different parameters. Since the experiment is subjective test, there are many factors to affect the results. In this thesis, all experiments are tested in the same enviroment, for example, the same brightness and the same backgroud. The same monitor is used for the all experiments and the voltage and the other conditions are all exactly same. The motivation of the experiments is to use human perception to measure the colour difference in RGB gamut.

5.2 Colours with same brightness of 24-bit RGB colour model in CIEL*a*b* colour model

This experiment is designed to divide the **CIEL*a*b*** colour model into different layers so that each layer has an identical value of brightness, and each layer only displays the colours located in the 24- bit **RGB** model. The purpose of this experiment is to find how the brightness affects the colour appearance.

In order to convert 24-bit **RGB** values to **CIEL*a*b***, some new functions are created with MATLAB:

Function *rgb2lab* converts **RGB** values to **CIEL*a*b*** values. The input 24-bit **RGB** values can be either a single vector or a 3D array and the output **CIEL*a*b*** values have the same type as input. Similarly, function *lab2rgb* converts **CIEL*a*b*** values to 24-bit **RGB** values. The algorithm is shown in section 4.2, 4.3, and the MATLAB code is listed in Appendix A. Since the **CIEL*a*b*** colour model is larger than the 24-bit **RGB** model, as the values of **CIEL*a*b*** convert to 24-bit **RGB** values, some of them are out of the range [0, 255]. In general, values less than 0 are considered as 0, and values more than

255 are considered as 255. Therefore, it is not always correct to use 24-bit **RGB** values to represent the colours in **CIEL*a*b*** colour model. The MATLAB source codes are listed in Appendix A.

The function *lab2rgb_inrgbgamut* is almost same as the function *rgb2lab* but only colours are located in the 24-bit **RGB** colour model are converted; any colours out of the 24-bit **RGB** colour model are set to zero. The function *lab2rgb_inrgbgamut* thus can display the relationship between 24-bit **RGB** and **CIEL*a*b***. (The source codes are listed in Appendix A) To show 24-bit **RGB** values in **CIEL*a*b*** colour model with the same brightness, it can be done as below:

1. Create two 201 by 201 matrices to represent ***CIE a**** and ***CIE b**** values, because the values of ***CIE a**** and ***CIE b**** are defined from –100 to +100. The ***CIE a**** matrix starts column 1 as value –100, the value of each next column is increased by 1 from the current column, and end with column 201 as value 100. Each row of the ***CIE a**** matrix is exactly same. The ***CIE b**** is created similarly as ***CIE a****, which has the same value for each column, and the values of each row are progressively increased by 1 from –100 to 100. (See below)

$$\begin{bmatrix} -100 & -99 & \cdots & 0 & \cdots & 99 & 100 \\ -100 & -99 & \cdots & 0 & \cdots & 99 & 100 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ -100 & -99 & \cdots & 0 & \cdots & 99 & 100 \end{bmatrix}_{CIE\ \ a^*} \qquad \begin{bmatrix} -100 & -100 & \cdots\cdots & -100 \\ -99 & -99 & \cdots\cdots & -99 \\ \cdots & \cdots & \cdots\cdots & \cdots \\ 100 & 100 & \cdots\cdots & 100 \end{bmatrix}_{CIE\ \ b^*}$$

Their values can be expressed:

$$CIEa^*_{i,j} = j - 100 \qquad (5\text{-}1)$$

$$CIEb^*_{i,j} = i - 100 \qquad (5\text{-}2)$$

where, matrix ***CIE a**** and ***CIE b**** have the exactly same size with the row and column, *i* and *j* are varied in the same range [0, 200]

2. Create a matrix of the same size for the brightness values of **CIEL*a*b*.** Elements of this matrix are constant with the constant in the range [0, 100]. Using the function *rgb_in_lab* converts all these **CIEL*a*b*** values to **RGB**.
3. Since the output 24-bit **RGB** values are matrices which have the same size as the input **CIEL*a*b*** values, each element of the matrices has the same coordinates corresponding to the **CIEL*a*b*** colour model. The output 24-bit **RGB** values can be considered as an image which presents one layer in **CIEL*a*b*** with the specified brightness. In this image, all colours in the 24-bit **RGB** gamut are displayed with their exact appearance, other colours out of the 24-bit **RGB** are set to zero which appears black.

Function *rgbinlab_brightness(arg1,arg2,arg3)* can create different groups of colours of 24-bit **RGB** with same brightness in **CIEL*a*b*** colour model, using the same three steps as above. The parameters *arg1* and *arg2* are the range of values of brightness, and parameter *arg3* is the step of the brightness varying in that range. The value of *arg1* must be less than *arg2*. If there is only one input parameter, only one image with that input brightness value is created. By default, the step *arg3* is initialized to value one. The source code is listed in Appendix A. Figure 5-1 lists six images with different brightness. (Appendix B lists more images with special values of brightness) As these images show, colours are not uniformly distributed on each plane. At the low brightness for which the value L* is less than 20 and the high brightness for which the value ***L**** is more than 90, the 24-bit **RGB** only has few colours in the **CIEL*a*b*** colour with the same brightness. Furthermore, the blue colours appear mainly at low brightness values, and the yellow colours are mainly located at high brightness value. Since this project is concerned with measuring colours of green vegetables, the main purpose is to distinguish the colour difference in the green and yellow areas. Five people were asked to distinguish the colours with different brightness in these images: whether the colours are always same with the same brightness but different hue and saturation. The results present that people are not sensitive the colour difference as the colour is too dark or too bright (Table 5-1).

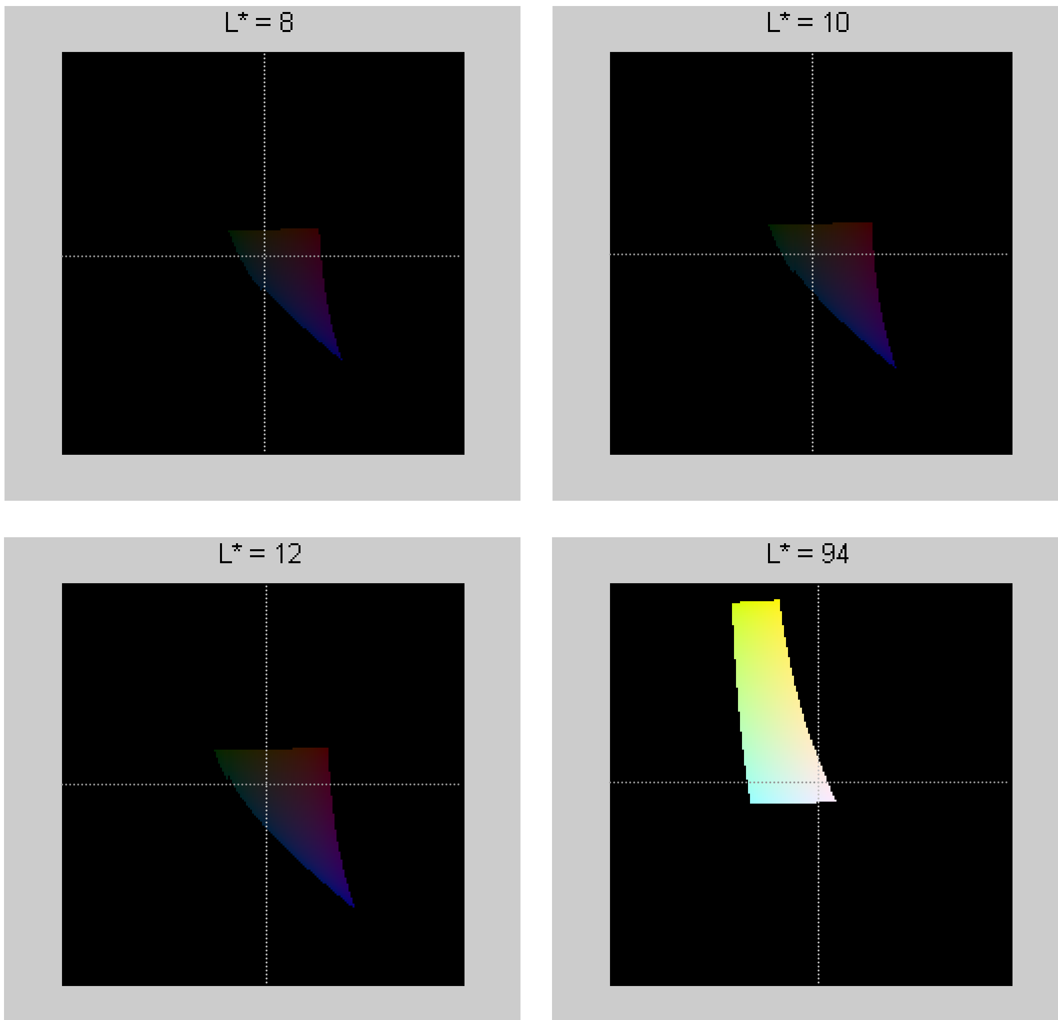
These results can be deduced from Table 5-1:

Result 1:

Any colour in 24-bit RGB colour model for which the value of brightness in CIEL*a*b* is less than 10 can be considered as background because such a colour cannot be perceived at the green and yellow parts, and the blue colours do not appear in green vegetable.

Result 2:

Any colour in 24-bit RGB colour model for which the value of brightness in CIEL*a*b* more than 95 appears as yellow or white.



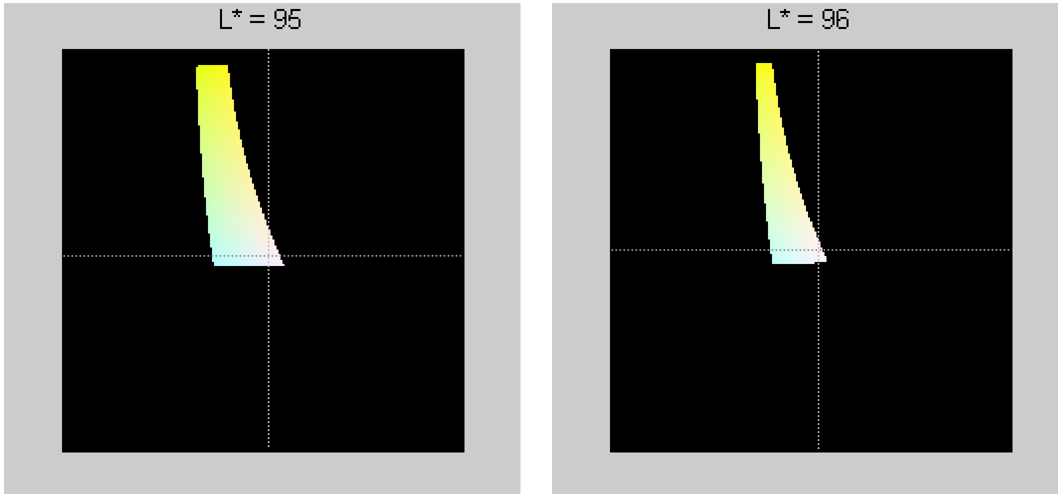


Figure 5-1 Colours of 24-bit RGB in CIEL*a*b* with specific brightness

Person	**Level of brightness, at which colours are too dark to distinguish**	**Level of brightness, at which colours have no green appearance**
1	**12**	**93**
2	**10**	**94**
3	**12**	**95**
4	**10**	**93**
5	**12**	**94**

Table 5-1 Results of measuring colours with different brightness

5.3 Colours with same hue of 24-bit RGB colour model in CIEL*a*b* colour model

The previous experiment showed how the colours are distributed at different brightness levels. However, even with constant brightness, the number of colours is too great to describe their details. The next experiment sets a constant hue value of **CIEL*a*b*** and investigates the relationship between brightness and saturation. The perceived colour is mainly determined by the value of hue. However, saturation and brightness also are very important factors if the hue values are similar [Hun95]. In the next chapter, we shall discuss the effects of brightness and saturation on colours with different hue values. Here, all colours are supposed to have the same value of hue. As before, we shall conduct the experiment using MATLAB routines:

Function *rgbinlch_hue(arg1,arg2,arg3,arg4)* converts all colours in 24-bit **RGB** colour model with the same hue value to **CIEL*C*H*** colour model and displays the colours according to the coordinate of **CIEL*C*H***. This function is similar with the function *rgbinlab_brightness*:

1. Create two 101 by 101 matrices to represent ***C**** and ***L**** value because the value of ***C**** and ***H**** are varied in the range [0,100]. The values of matrices ***L**** and ***C**** are similarly defined to the matrices ***CIE a**** and ***CIE b**** in the last experiment, they can be expressed as the formulas below:

$$L*_{i,j} = i \tag{5-3}$$

$$C*_{i,j} = j \tag{5-4}$$

   where, the values of i and j are varied in the range [0,100]. Every column of matrix ***L**** is same and the values of row are varied from 0 to 100; every row of matrix ***C**** is same and the values of column are varied from 0 to 100.

2. Create a 101 by 101 matrix with the same size as ***C**** and ***L**** to hold the hue values. Hue values, ***H****, are constant comparing with the above ***C**** and ***L****. Changing the **CIEL*C*H*** values to **CIEL*a*b*** and then using the function *lab2rgb_inrgbgamut* converts **CIEL*a*b*** values to 24-bit **RGB** values.
3. Display the output 24-bit **RGB** colours. All colours in the image have the same hue values. The constant of the hue value can be chosen from 0 to 360 degree to display more images as above for different hue values.

The parameter *arg1* in the function *rgbinlch_hue(arg1,arg2,arg3,arg4)* is the size of the output image. Since the numbers of row and column in the above matrices are same, the output images are square and one parameter can describe their size. This value is same as the numbers of pixels of the side in the image. Parameters *arg2* and *arg3* are the minimum and maximum of the hue range. Parameter *arg2* must be less than *arg3*. Parameter *arg4* is the step of the hue in its range. There are at least two and no more than four input parameters in this function. If only two parameters are given, there is only one output image, for which the hue value is defined by the second parameter. By default, the hue step is one degree. The source code is listed in Appendix A. Figure 5-2 represents colour appearance in the 24-bit **RGB** colour model for different brightness and saturation and with a constant hue, ***H**** set to 108 degree. This hue value has a yellow-green appearance.

In figure 5-2, the brightness doesn't affect the colour appearance very much. However, as the saturation decreases to zero, this colour becomes paler. Five people were asked to describe the colour difference with different saturation in this image and the result is listed in table 5-2. We created five similar images, each with a constant hue value from 0 to 180 degrees (See Appendix B). Each image represents one major colour that is located in the red, orange, yellow, and green areas of the **CIEL*C*H*** colour model. Five people were asked to measure the colour difference of these images and the results are almost the same as the result of Table 5-2.

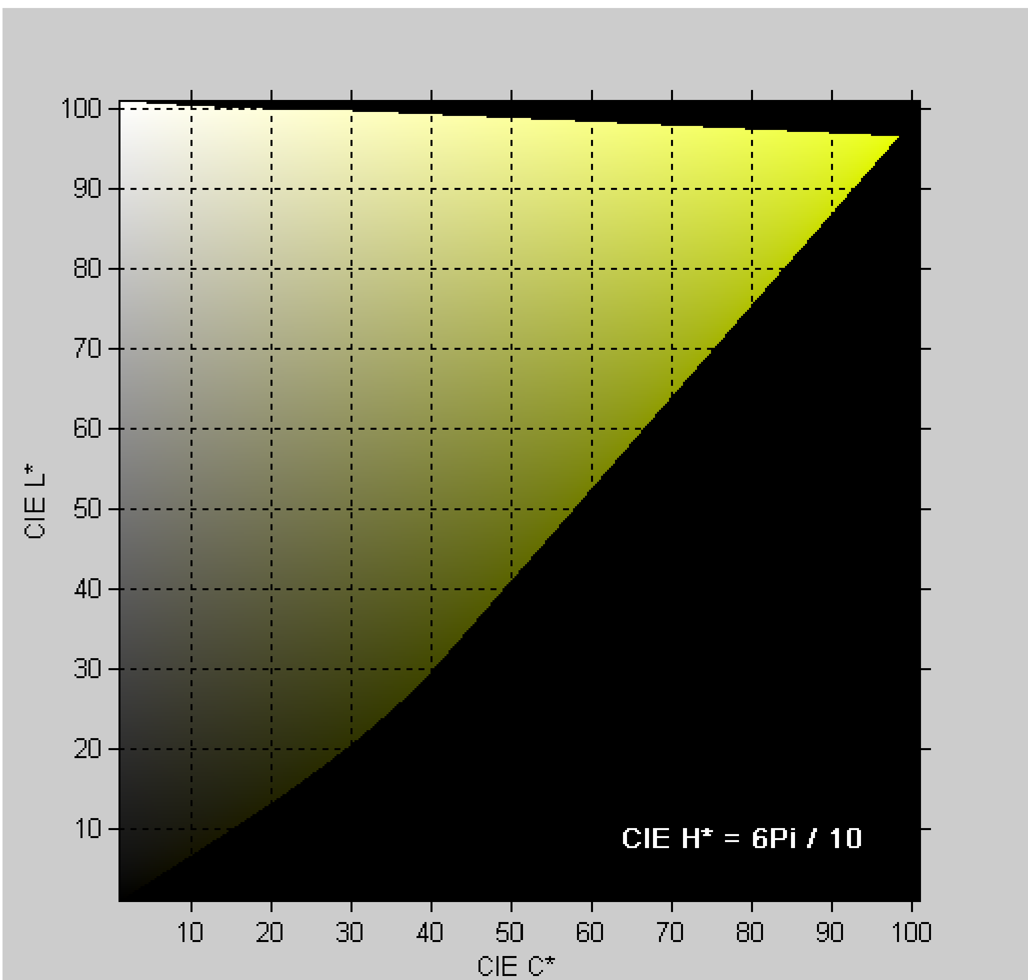


Figure 5-2 Colours of RGB in CIEL*C*H* with 108 degree hue

We can conclude two further results:

Result 3:

If the colour is located in the red, yellow, and green area of the CIEL*C*H* colour model, any colour in the 24-bit RGB colour model for which the value of saturation in CIEL*C*H* is less than 10 can be considered as background because such a colour is too pale to distinguish its colour appearance by human perception.

Result 4:

If the colour is located in the red, yellow, and green area of the CIEL*C*H* colour model, any colour in the 24-bit RGB colour model for which the value of saturation in CIEL*C*H* is more than 80 can be considered as having the same colour appearance because their colour differences are too small to distinguish by human perception.

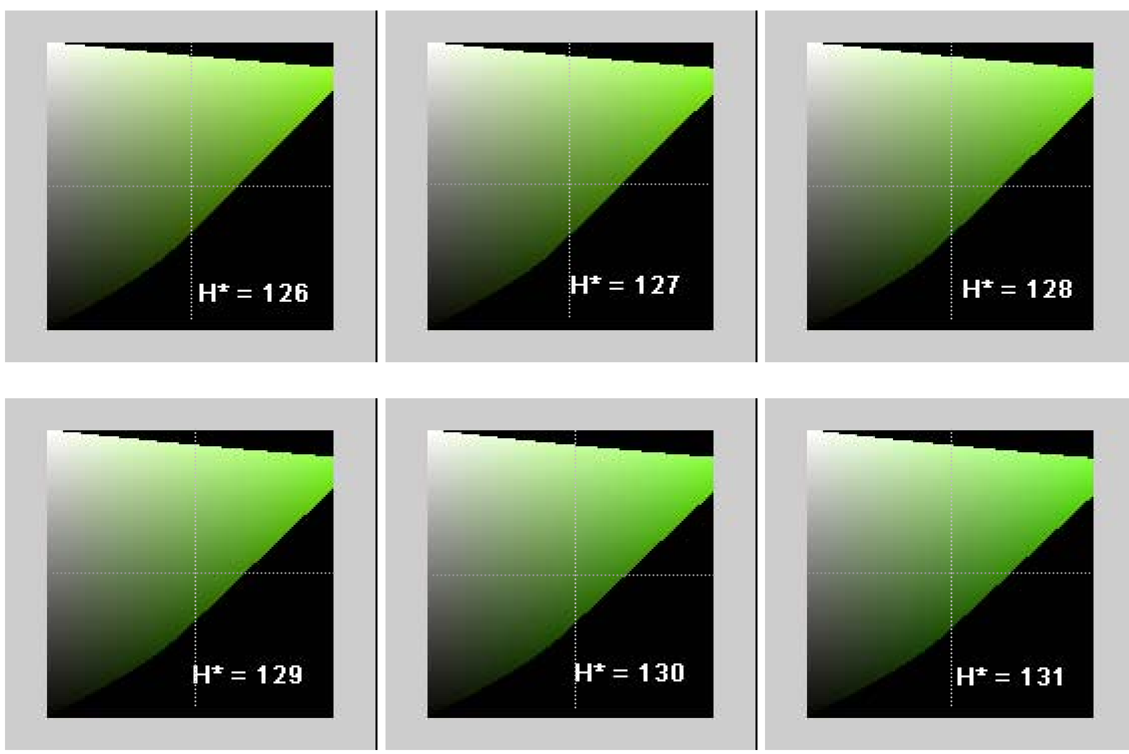
5.4 Colour difference with different value of hue

The five images of the last experiment have their major hue appearance to represent red, yellow, and green respectively. However, their hue differences are easy to distinguish. The purpose of this experiment is designed to find the minimum hue difference that the human eye can distinguish.

Person	Maximum level of saturation at which colours appear the same	Minimum level of saturation at which colours appear the same
1	10	80
2	10	75
3	15	75
4	10	80
5	15	75

Table 5-2 Results of measuring colour with different hue

The MATLAB function *rgbinlch_hue* is used to create an image in which colours have the same hue value, but different S, V. There are five groups of images created in this experiment. Each group has nine images with one degree hue difference to the next one. Their hue values are 42-50, 70-78, 100-108, 126-134, and 150-158 degrees respectively. Figure 5-3 shows one group of images whose hue values are from 126-134. Ten people were asked whether they can distinguish the colour difference in these nine images, which are shown on the same monitor. (The hue values are not displayed on the images and the orders of the images are random as these images were measured) The information is collected and arranged in table 5-3:



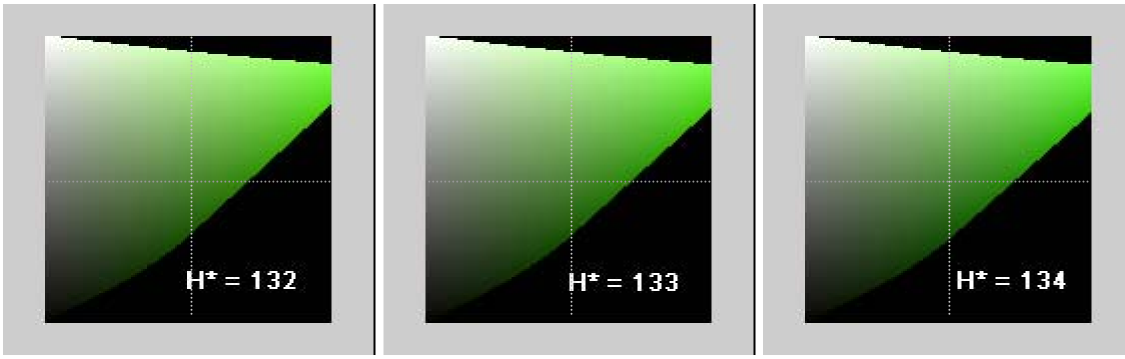


Figure 5-3 Colours of 24-bit RGB in CIEL*C*H* with specific hue

The images are compared in irregular order, for example, hue 128 may be compared to hue 129, 126, 134 or 130. The testing people do not know the hue value. Compared with measuring the other groups of images, green and yellow colours have the same results. Although the measurements of red colours are not same as the green and yellow colour are distinguished their colour differences, they are not considered in this thesis because we only focus the yellow and green colours.

	The value of hue difference as two images are measured		
Person	Almost same	Don't know	Different
1	Less than (including) 4	5	More than 5
2	Less than (including) 3	4 and 5	More than 5
3	Less than (including) 4	5	More than 6
4	Less than (including) 4	Nil	More than 5
5	Less than (including) 3	4	More than 4

Table 5-3 Results of measuring hue difference

Thus, a new result can be concluded from the table 5-3

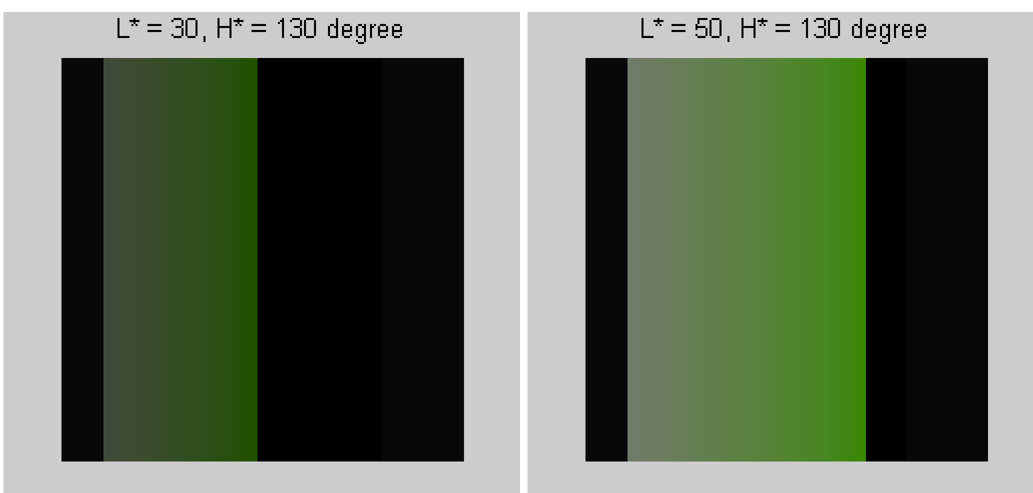
Result 5:

For colours located in yellow and green areas of the CIEL*C*H* colour model, if two such colours have the same values of brightness and saturation, their colour appearance can be considered the same if their hue difference is less than 4 degrees.

5.5 Colour difference with different saturation and brightness

Humans as we have seen in section 3.5 like to describe colour differences using brightness, hue, and saturation. [Hun96] In this project, as the quality of green vegetables are measured, the greener the colour of the vegetable, the higher the quality the vegetable. Yellow and orange colours usually indicate poor quality. So, it is most important to measure the hue of the vegetable to determine its quality. To measure the hue degree of two colours, it can be determined which colour is closer to orange, yellow, or green. However, it is not yet clear how saturation and brightness affect human perception of the vegetable. We will investigate this in the next chapter.

The next experiment investigates how people describe the colour difference if the colours have same hue and brightness but different saturation, or the same hue and saturation but different brightness. According to the five previous results, any colours whose saturation is more than 80 and less than 10, or whose brightness is less than 10 do not need to be considered. There are too many colours to measure. To make the experiment manageable, five samples of hue are chosen to represent the main hue appearance which range over the red, yellow, and green areas in the **CIEL*C*H*** colour model. Their values are 110, 120, 130, 140 and 150 degrees.



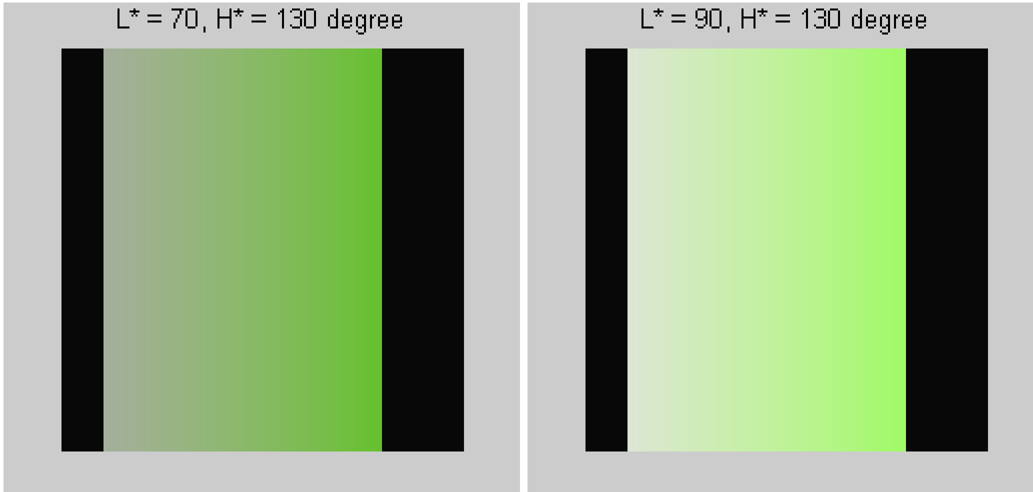


Figure 5-4 Colours of RGB in CIEL*C*H* with specific brightness and hue

5.5.1 Same hue and brightness with varying saturation

In figure 5-2, colours do not only have varying saturation but also the varying brightness. Therefore, colours with a given hue must be perceived with their saturation varying under different brightness. The MATLAB function *diffc_inlh* is designed to create these kinds of colours. The input parameters are the values of brightness and hue. The output images have all colours with different saturation with the chosen values of brightness and hue (for details, see the source code in Appendix A). In this experiment, four brightness values are chosen to display the saturation varying in the each of the five sample hues, their values are 30, 50, 70, and 90 respectively in the **CIEL*C*H*** colour model. Figure 5-4 is one of the examples for which the sample value is 130 degrees (more images are listed in Appendix B). Five people were asked to view four such images to describe the colour differences as the saturation varied. All answers indicate that the higher saturation of the colour, the greener the colour; or the smaller saturation of the colour, the yellower the colour (See the survey 1 of the Appendix C). Hence, a new result can be deduced from this experiment:

Result 6:

If any colour has green and yellow appearance, and if their hue and brightness value are constant, then the higher the saturation of this colour, the greener this colour appears; inversely, the smaller the the saturation of this colour, the yellower this colour appears.

As the saturation of the colour decreases, the colour gradually becomes pale and finally becomes grey, which means no colour. Referring to our results 3, 4, and 6, we can deduce:

If two colours have similar hue value, and if their brightness values are same, the higher the saturation they have, the more easily they can be distinguished.

5.5.2 Same hue and saturation with varying brightness

The following experiment is similar to that discussed in the section 5.4.1. The MATLAB function *diffl_inch* is designed to create these kinds of colours. The input parameters are the values of saturation and hue. The output images have all colours with different brightness for such saturation and hue (the source code is listed in Appendix A). The same five sample hue values and four saturation values are chosen to measure the colour difference with the varying brightness; the saturation values are 20, 35, 50, and 65 in the **CIEL*C*H*** colour model. Figure 5-5 shows two sample images of this kind (others are given in Appendix B). As five people were asked to view these images, there was the same answer: the darker the colour is, the greener it looks with the same hue and saturation values (see the survey 2 of the Appendix C). Here is the new result:

Result 7:

If the hue and saturation of colours are constant, the appearance of green and green-yellow colours tends to a green hue as the brightness decreases and to a yellow hue as the brightness increases.

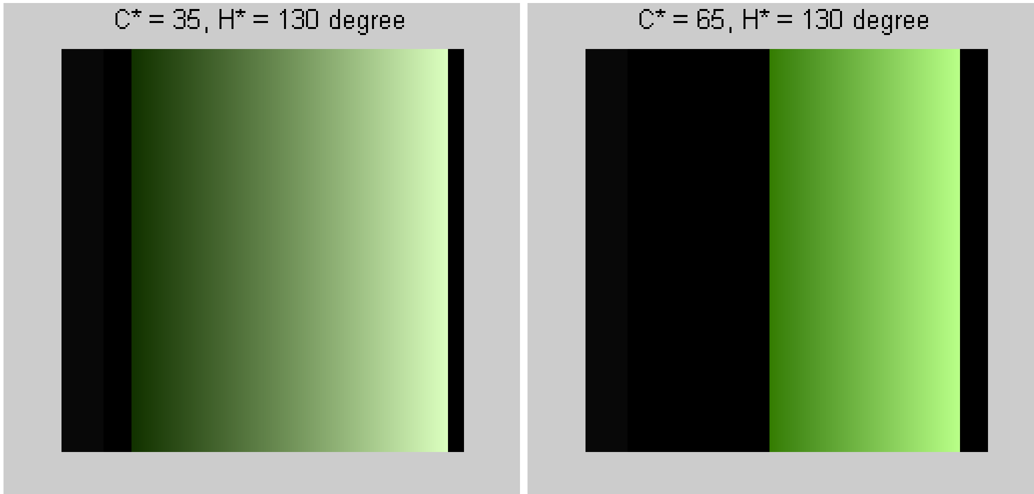


Figure 5-5 Colours of RGB in CIEL*C*H* with specific brightness and saturation

5.6 Conclusion

Using various colour properties, five experiments have been designed using MATLAB to measure the colour difference with different conditions. Seven results have been obtained from these experiments to measure colours in the **CIEL*a*b*** and **CIEL*C*H*** colour models. These results conclude the general information of colour difference as the brightness, hue, and saturation vary, and exclude some colours in the 24-bit **RGB** colour model which are not available. It provides some simple methods for the further research and provides in foundation for our next work. However, these results are obtained from a particular condition: that is, that at least one of the brightness, hue, and saturation is constant. Therefore, these results cannot be applied to general colour differences. In the next chapter, colour differences will be analysed and discussed with all parameters varying.

Chapter 6
Development of colour difference in green vegetables

6.1 Introduction

Freshness is an important factor in measuring the quality of the vegetables and freshness is mainly determined by the colour appearance of the vegetable. For a green leafy vegetable, such as broccoli, celery, and Chinese cabbages, the greener the colour of the vegetable, the higher quality it has. Withered or rotten vegetables always have a yellow or orange appearance. In this project, we only discuss how the colours indicate the quality of vegetables. We shall show that the quality of green vegetables can be determined by the number and appearance of green and yellow colours.

In the previous chapter, colour differences are explored according to their brightness, hue, and saturation using MATLAB. Seven results were obtained from experiments. These results provide general information of colour differences, especially for colours with red, yellow, and green appearance. However, these results are only applicable under the conditions that at least one of hue, saturation or brightness is constant. In this project, colours are generated from images of leafy vegetables. Hence, colours must be measured under more general conditions.

6.2 Colour difference in different conditions

Normally, the appearance of a colour is mainly determined by its hue, for example green, blue, or red. In the **CIEL*C*H*** colour model, the hue value increases as the colour changes from yellow to green. Hence, the bigger the hue value of the colour, the greener its appearance. Basically, yellow and green colours have the hue value between 70 to 150 degrees. According to result 5 in the chapter 5, colours appear the same if the hue difference between them is less than 3 degrees; hence, only 28 hue levels can be distinguished between green and yellow colours.

It is quite easy to check the colour difference by their hue difference. However, that result is based on colours which have the same value of brightness and saturation. According to other results in chapter 5, colour appearance is also determined by the brightness and saturation. Therefore, hue measurement is not the only factor to affect the colour appearance. If the hue difference is not evident, the brightness and saturation are the main factors to affect the colour appearance. For example, if two colours (A and B) have a green-yellow appearance, and if the hue value of colour A is five degrees greater than the hue value of colour B, the colour A can be considered to have greener hue appearance than colour B. However, if the saturation of the same colour A is much smaller than colour B, colour A is paler than colour B. Does the pale colour A appear greener than the vivid colour B?

The main purpose of this chapter is to investigate the colour appearance under all possible conditions with green and yellow colours. To compare the colour difference between two colours, there are seven possibilities to be checked for brightness, hue, and saturation. Table 6-1 lists the all these possibilities. The colour differences in the conditions 1, 2, and 3 have been analysed and discussed in chapter 5, where we have obtained result 5 for condition 1, result 6 for condition 2, and result 7 for condition 3. Condition 4 is obtained based on condition 2, and condition 5 is obtained based on condition 3 to explore colour difference further. Condition 6 measures the colour differences such as the colours in figure 5-2. Finally, in condition 7, colour differences are explored and analysed in a real environment. The following experiments are designed to measure the colour difference step by step under all these possible conditions.

Condition	Brightness	Hue	Saturation
1	Same	Different	Same
2	Same	Same	Different
3	Different	Same	Same
4	Same	Different	Different
5	Different	Different	Same
6	Different	Same	Different
7	Different	Different	Different

Table 6-1 Colours varying with different condition

6.3 Minimum value of brightness and saturation difference by human perception

In chapter 5, the minimum value of hue difference by human perception with the same brightness and saturation has been limited at three degrees. If the minimum values of brightness and saturation difference by human perception also can be limited, then the sample colours can be exactly chosen to measure their difference.

6.3.1 Minimum value of brightness difference with same hue and saturation

To determine a minimum value of brightness difference, all possible colours with different hue and saturation have to be considered. We will not analyse all possible values of hue and saturation, but restrict our analysis to colours in the green and yellow area in **CIEL*C*H***, using a minimum hue difference of three degrees. In this experiment, hue values are chosen from 70 to 150 degree with a 15 degree step giving six hue values; saturation values are chosen from 20 to 80 in steps of 20. Since not all colours in the **CIEL*C*H*** colour model are available in the **RGB** colour model (see figure 4-2), not all values of brightness can be measured. There are 24 groups of colours corresponding to the possible values of hue and saturation. We create the MATLAB function *sc_lch* to produce the different colours. The input parameters of this function are the three values of the **CIEL*C*H*** colour model. The output is an image with a single colour of the input values (See the Appendix A for more details). Figure 6-1 lists some of one group of colours for which the hue is 110 degree and the saturation is 70. In this case, the values of brightness vary from 64 to 96. In this experiment, colours that have same hue and saturation are set in the same group. Colours within the same group are compared with each other only. As five people were asked to measure the colours from these groups, most people were able to distinguish a difference of the colours even when the colours have brightness difference by one or two. Hence we may conclude:

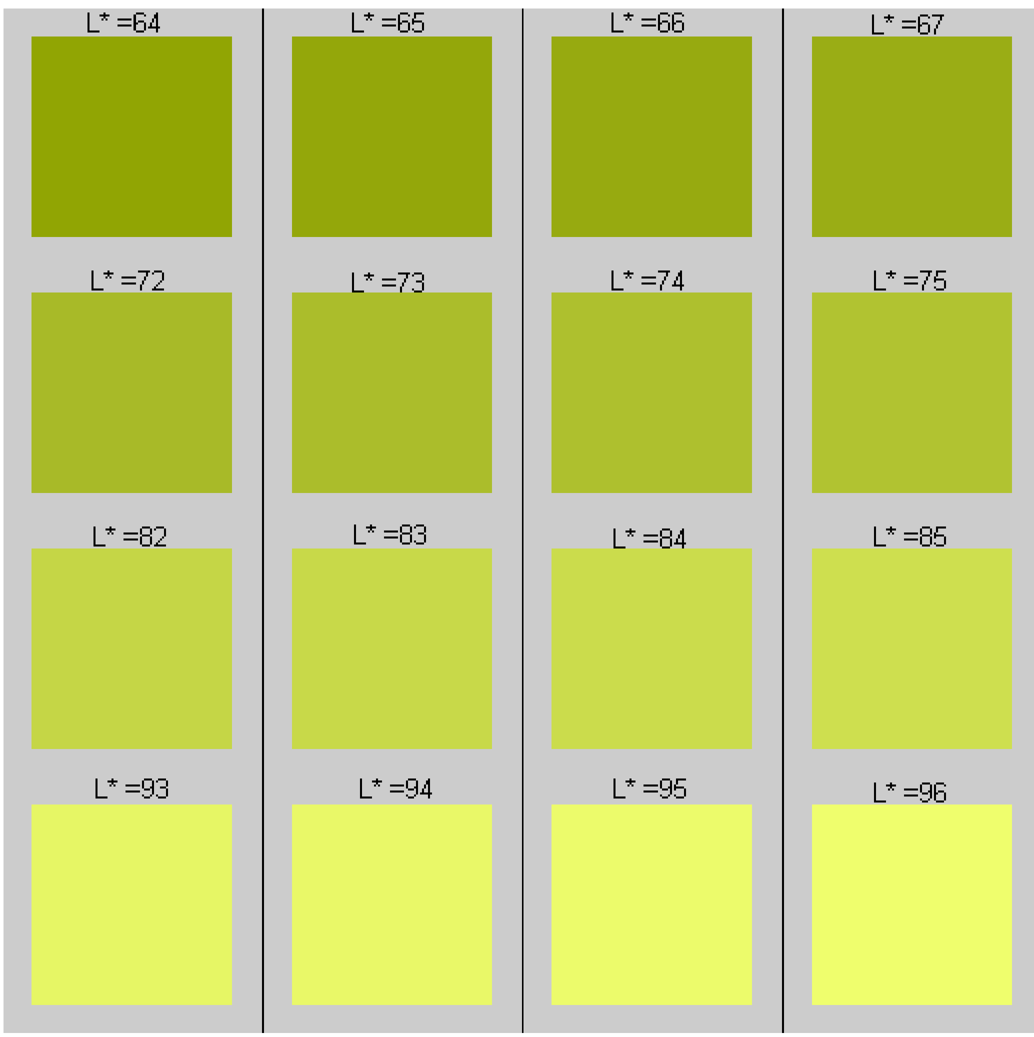


Figure 6-1 Colours with 110 degree hue and 70 saturation but different brightness

Result 8:

If colours have a green or yellow appearance, and if their values of hue and saturation are constant, a brightness difference of one can be perceived.

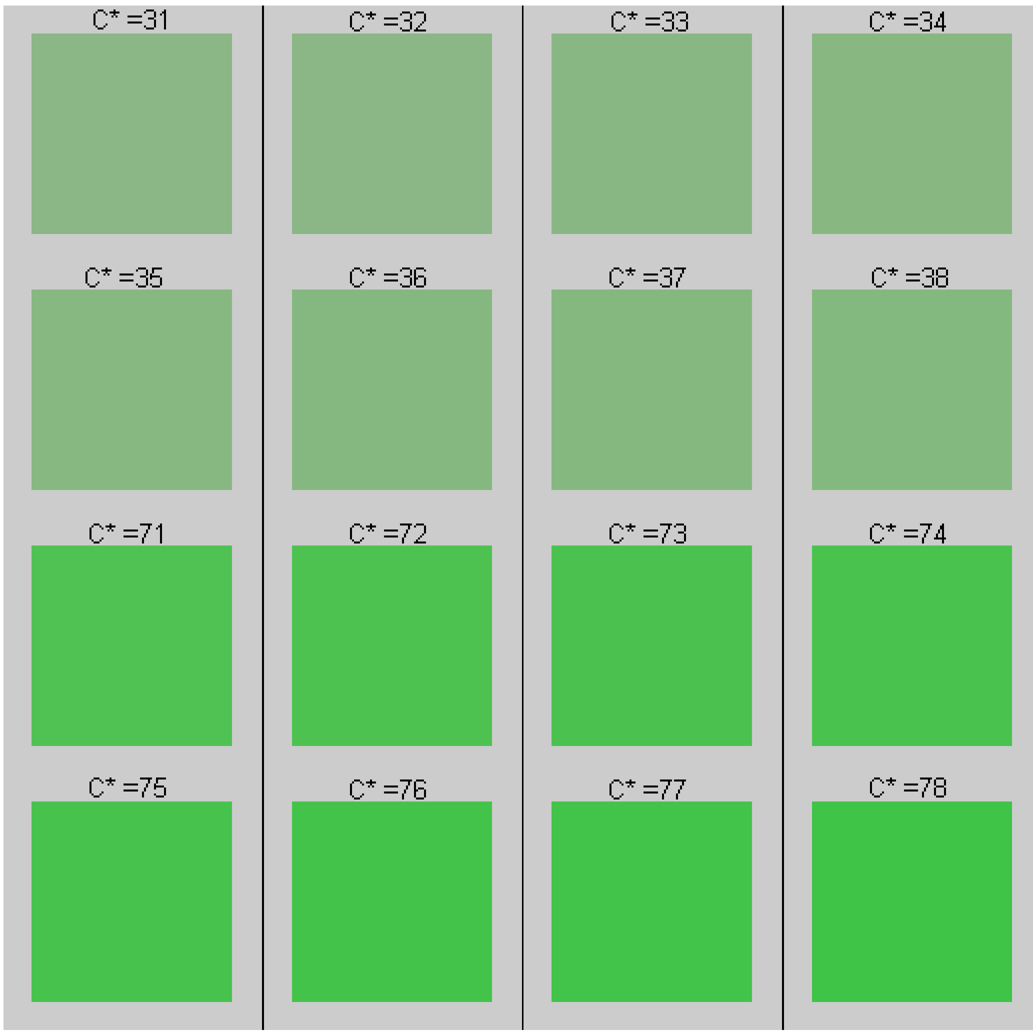


Figure 6-2 Colours with 140 degree hue and 70 brightness but different saturation

6.3.2 Minimum value of saturation difference with same hue and brightness

The measurement of the minimum saturation difference with the same hue and brightness requires a similar experiment as the last one. Six hue values are chosen from 70 to 150 degree with a 15 degree step, and the values of brightness are chosen from 20 to 80 with a step of 20. Also not all colours with all saturation values can be displayed in the **RGB** colour model. The MATLAB *sc_lch* function is used to create these colours. Figure 6-2 lists some colours with different saturation for which the hue is 140 degrees and the brightness is 70. Five people were asked to measure the colour difference from all groups, nobody could distinguish the colour difference if the saturation difference is less than 4. Here we may conclude:

Result 9:

If colours have a green or yellow appearance, and if their values of hue and brightness are constant, their minimum perceivable saturation difference is 4.

According to result 8 and result 9, humans can distinguish colour difference with only 1 unit brightness difference and 4 units saturation difference. Therefore, in the **CIEL*C*H*** colour mode, we may deduce that human eyes are more sensitive to varying brightness than to varying saturation. [Fol96]

6.4 Colour measurement with different hue and saturation

The purpose of this experiment is to find how the difference of both hue and saturation simultaneously affect the appearance of colours with the same brightness. There are two variables in this experiment, so it is more complicated than before. We create four groups of colours with the brightness 40, 55, 70, and 85 respectively. In the **RGB** colour model, as the value of brightness decreases, the maximum value of saturation also decreases, especially for the green and yellow colours, because the **RGB** colour gamut is smaller than that of **CIEL*C*H*** (See figure 5-2). If the brightness is chosen to be too small, the range of saturation is too small to obtain a general result. According to the previous results, the colour difference can be distinguished as the saturation varies within [10,80] with the same hue and brightness, and the minimum saturation difference observable by the human eye is 4. Therefore, we choose sample colour patches in this experiment whose saturation varies from 10 to 82 with a step of 4 units. Figure 6-3 shows part of the colour patches for which the hue values are varied from 110 to 129 degree with 1 degree step. Since the minimum hue difference is 3 degrees and the minimum observable saturation difference is 4 units, such colour patches almost cover enough colours for sampling green-yellow colours with brightness of 70. There are four parts to this experiment.

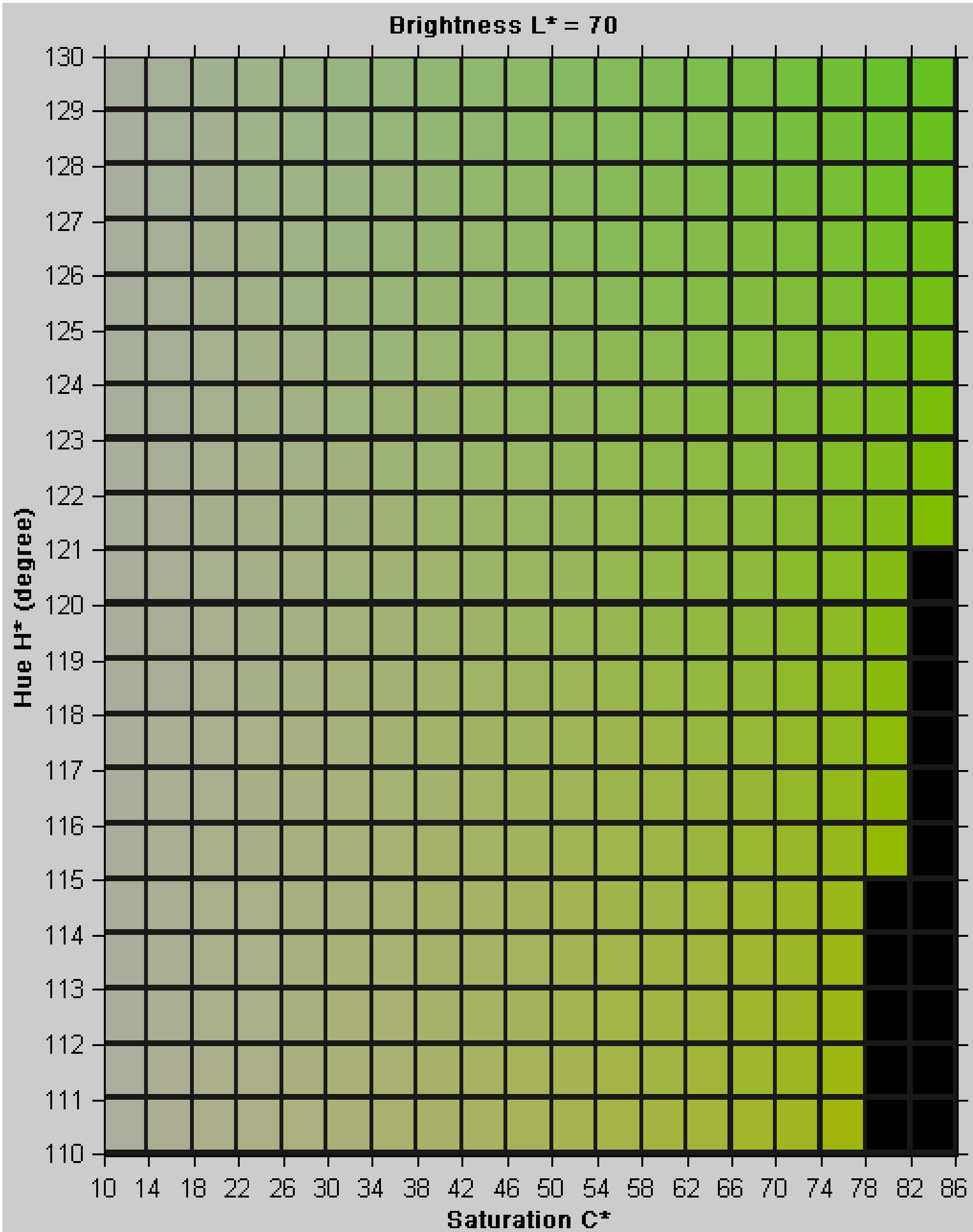


Figure 6-3 Colour patches of varying hue and saturation with a fixed brightness value of 70

6.4.1 Minimum hue difference with different saturation

According to the result 5 in chapter 5, the minimum perceivable difference of hue with colours of the same brightness and saturation is 3 degrees. Since the lower the saturation of a colour, the paler it is, and pale colours are more difficult to distinguish than vivid colours with the same brightness and hue (colours are too pale to distinguish their appearance if the value of saturation is less than 10; see results 3 and 4); the minimum perceivable hue difference depends on saturation. To check the minimum hue difference under different saturation, colour patches with different hue and different saturation are compared with each other. Table 6-2 has the result of the minimum hue differences with different saturations after five people were asked to measure the colour patches in figure 6-3. According to the table 6-2, the minimum value of hue difference is rough linear function of the saturation value. Based on result 5, we may conclude:

	Minimum observable value of hue difference (degree)				
Saturation	Person 1	Person 2	Person 3	Person 4	Person 5
10	9	8	9	7	8
14	7	7	7	7	8
18	6	7	6	6	6
22	6	6	6	6	6
26	5	5	5	6	5
30	4	5	4	5	5
34	4	4	4	4	4
38	3	4	4	4	4
42	3	3	4	3	3
46 and more	3	3	3	3	3

Table 6-2 Minimum value of hue difference

Result 10:

If colours have a green or yellow appearance, and if the saturations of colours are more than 42, their minimum perceivable hue difference is 3 degree at the same brightness and saturation; otherwise, as the saturation

decreases from 42 to 10, the minimum hue difference increases linearly from 3 to 8; and if the value of saturation is less than 10, colours can be considered too pale to distinguish their hue difference.

6.4.2 Colour difference between more hue and saturation and less hue and saturation

Since only the value of brightness is constant in this experiment, there are two different conditions to be considered if there are two colours being measured: more hue with more saturation comparing with less hue with less saturation; and more hue with less saturation comparing with less hue with more saturation. According to the result 6 from the chapter 5, for colours with the same brightness and hue, the higher the saturation, the greener the colours. Also, colours with higher hue tend to have greener appearance. Thus we conclude that colours with higher saturation and hue value have a greener appearance than colours with smaller saturation and hue value. For colours which are too similar to be distinguished their difference by human perception, colours with more hue and saturation are greener than the colours with less hue and saturation in theory. To prove this result, six pairs of colours are created, and five people are asked to measure their difference (See the table 6-3). The hue and saturation differences are chosen small enough because large differences are easy to distinguish. Figure 6-4 lists such two pairs of colours (The others are listed in the Appendix B). The results of measuring six pairs of colours are exactly the same. The new result is:

Result 11:

If colours mainly have green or yellow appearance, and if the value of brightness is constant, colours with large value of hue and saturation have greener appearance than the small value of hue and saturation; conversely, colours with small values of hue and saturation have yellower appearance than colours with large values of hue and saturation.

	Pair 1	**Pair 2**	**Pair 3**	**Pair 4**	**Pair 5**	**Pair 6**

L*	70	70	70	70	50	50	60	60	80	80	80	80
C*	65	70	65	70	40	44	55	60	70	74	50	54
H*	126	130	96	100	100	104	140	145	120	124	110	114

Table 6-3 Sample pairs of colours with specific values

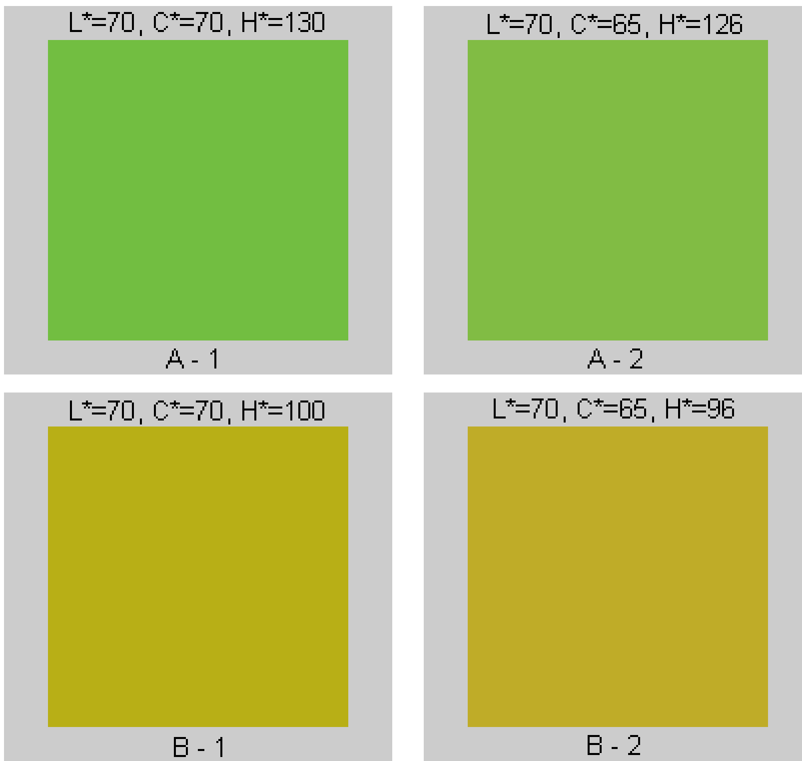
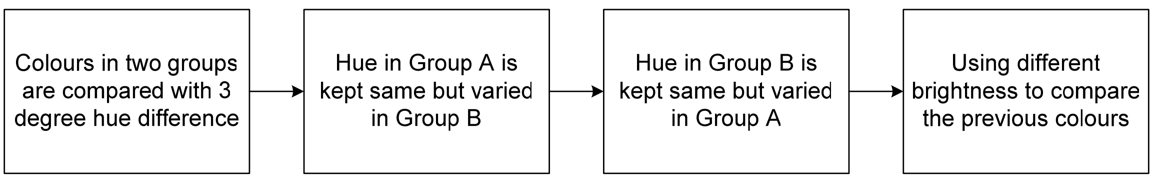


Figure 6-4 Colours with more hue and saturation and less hue and saturation

6.4.3 Colour difference between colour with more hue with less saturation and colours with less hue with more saturation

This experiment is divided into four steps, which is shown in the following flow chart:

Step 1: Colour patches which have same hue in figure 6-3 are chosen as group A. Group B has similar colour patches but their hue value is 3 degrees less than the colours in group A. Each colour patch in group A is only compared with the colour patches of group B having the bigger saturation value, in order to measure which colour is greener or yellower. We created the MATLAB function *dc_lch* to produce two images with different colours. The input parameters are two **CIEL*C*H*** values (See the Appendix A for more details of the function. We mainly use this function to create two images with different colours and measure their difference in this chapter). Five people were asked to judge whether the colour patches with the less saturation in group B are greener than colour patches with small saturation in group A. In this step, the values of hue are chosen in the range from 70 to 150 (yellow and green appearance) and the brightness is constant at 70. Table 6-4 lists parts of the different minimum values of saturation with 125 hue value are greener than colour patches with 128 hue value.



	Hue	Saturation of colour patch with 125 hue is greener than 128 hue value									
Person	128	10	14	18	22	26	30	34	38	42	46
1	125	14	18	26	34	38	46	58	66	82	-
2	125	14	18	26	34	42	46	58	70	78	-
3	125	14	22	26	30	42	50	62	70	82	-
4	125	14	18	36	34	42	50	54	70	82	-
5	125	18	22	26	34	42	54	62	74	82	-
Average	125	15	19	26	33	41	49	59	70	81	-

Table 6-4 Results of measuring threshold value of saturation with 3 degree hue difference

According the results of table 6-4, the colour patches are too pale to distinguish if the saturation is less than 10. If the saturation is more than 46, colour patches with hue of 128 are always greener than colour patches with hue of 125 for any value of saturation. Therefore, if the saturation varies from 10 to 46, then colours with hue of 125 degree or less, which have larger saturation value, have greener appearance than colours with hue of 128 degrees.

Step 2: The hue difference (125 and 128 degrees) in the last step between two groups is 3 degrees. In this step, the colour patches of group A are same as last experiment have 128 hue degrees, but the colour patches of group B are instead of 122, 119 hue degrees. Again, five people were asked to judge their colour difference. The results are listed in the table 6-5. The purpose of this step is to check whether colour difference varies linearly as the values of hue and saturation vary.

	Hue	Saturation of colour patch with small hue is greener than 128 hue value											
	128	10	14	18	22	26	30	34	38	42	46	50	54
Average	125	15	19	26	33	41	49	59	70	81	-	-	-
Average	122	17	30	46	67	81	-	-	-	-	-	-	-
Average	119	24	49	75	-	-	-	-	-	-	-	-	-

Table 6-5 Average results of threshold value of saturation with different hue

Step 3: The value of group A in the previous steps was constant at 128. In this step, the hue values of group A are chosen as 100, 115, and 145, for which such hue values have the yellow and green appearance. The colours of group A with different hue values are measured with the colours of group B with different hue values as in the above two steps.

Step 4: All values of brightness are 70 at the above three steps. In this step, the values of brightness are chosen as 40, 55, and 85 to measure the colour difference using the same methods as the above steps (The colour patches are created same as figure 6-3 with different brightness, and the brightness of colour patches cannot be chosen too small because the maximum value of saturation decreases as the brightness decreases in the **RGB** colour model). The results are very similar as the table 6-5 as the brightness is varied and the errors of each result are very tiny. Since colour perception is very subjective, and even the person may measure colours differently, these errors can be ignored (more results are listed in the survey 3 of Appendix C).

Each result for identical hue is very close to a straight line as the saturation value with both small and big value. According to the standard linear regression function of mathematical statistics, the result in table 6-4 can be found using the formula below: [Hoe71]

$$y' = \bar{y} + b(x - \bar{x}) \tag{6-1}$$

where, $b = \dfrac{\sum (x-\bar{x})y}{\sum (x-\bar{x})^2}$, x is the value of saturation with more hue, y is the value of saturation with small hue, $\bar{x}$ is the average value of x, and $\bar{y}$ is the average value of y. The three results of table 6-4 can be expressed by the following equations:

$$S_{Hue\_125} = 2.1 \times S_{Hue\_128} - 10.4 \tag{6-2}$$

$$S_{Hue\_122} = 4.2 \times S_{Hue\_128} - 26 \tag{6-3}$$

$$S_{Hue\_119} = 6.5 \times S_{Hue\_128} - 42 \tag{6-4}$$

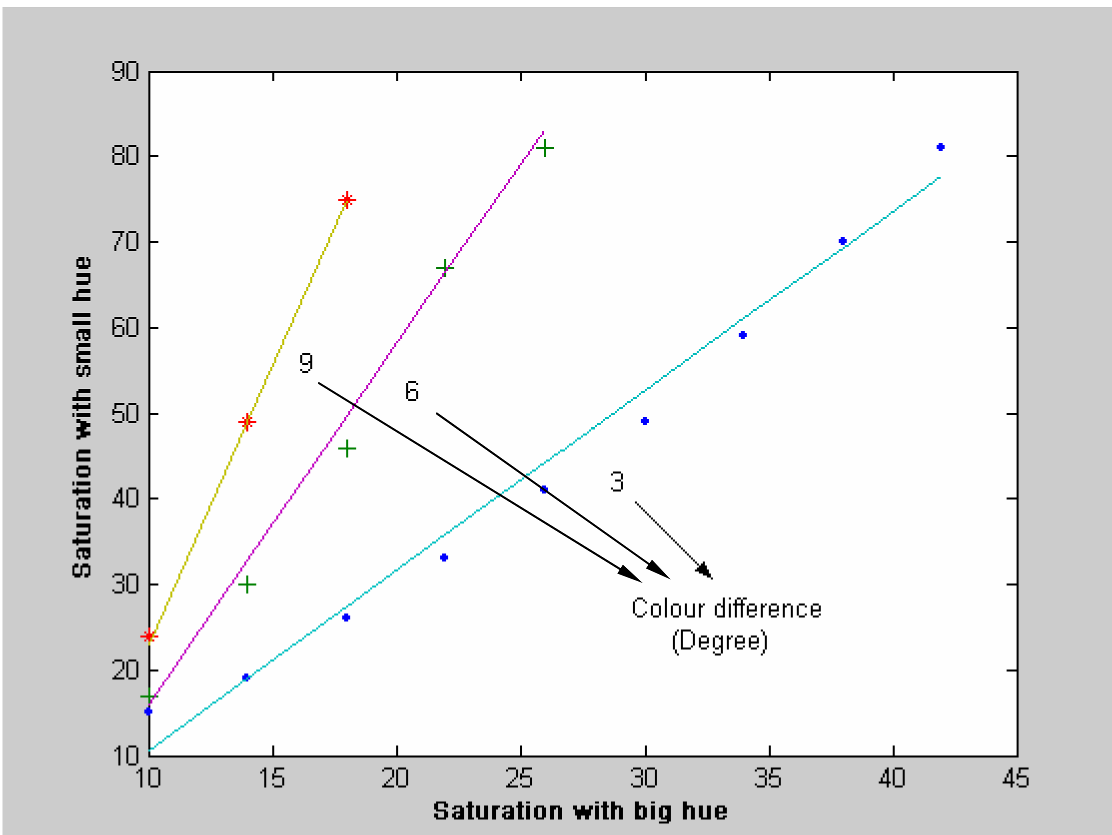


Figure 6-5 Lines of equation 6-5 obtained from the values of experiments

Based on the equations 6-2, 6-3, and 6-4, the linear regression function is used again to obtain an approximation to a linear result as the hue difference varies; the universal equation can be expressed as below:

$$S_{Small\_Hue} = 0.72 \times \Delta_{Hue} \times (S_{Big\_Hue} - 5) \qquad (6\text{-}5)$$

where, S_{Big_Hue} is in the range of [10, 42], and Δ_{Hue} is the hue difference between two compared colours. Colours which have greater saturation than S_{Small_Hue} are greener than the colours with saturation S_{Big_Hue}. The dots in figure 6-5 are the results of the experiment for measuring colour patches, and the lines are drawn by using the equation 6-5 for the corresponding values of colours in MATLAB.

In conclusion, the result is:

Result 12:

If the hue difference is more than 9 degrees, or if the saturation is more than 46, colours with more hue value always are greener than less hue value colours. If the hue difference is less than 9 degrees and the saturation is less than 46, then colours with less hue and more saturation appear greener than colours with more hue and less saturation. The varying of brightness and saturation is approximately linear, their relation can be derived from the linear regression function of mathematical statistics, and its general equation is listed in equation 6-5.

6.5 Colour measurements with different hue and brightness

The effects of hue and saturation to colour appearance have been discussed in the last section. However, we have based our investigation with brightness being constant. According to result 8 and 9, the minimum brightness difference by human perception is only one in one hundred units; the minimum saturation difference by human perception is 4 in one hundred units. Therefore humans are more sensitive to brightness than to saturation. [Fol96] This experiment will explore and analyse how brightness affects the appearance of colours with different hue, but with the same saturation. Similar colour patches in the last experiment are created for people to measure, but this time the saturation is constant. Since there are too many different colour patches, we won't list all the colour patches here. Figure 6-6 lists parts of the all sample colour patches, for which the hue is from 120 to 138 with one degree step, brightness is from 30 to 94 with 2 step, and the saturation is 50. The colour difference in the colour patches of figure 6-6 can be classified into two types, which are more hue and brightness compared with less hue and brightness, and more hue and less brightness compared with less hue and more brightness. According to result 7 in chapter 5, a colour with less brightness has a greener appearance than a colour with big brightness if the hue and saturation are same, and a colour with more hue is greener than a colour with less hue if the brightness and saturation are same. This can be stated as:

Result 13:

> **If colours mainly have green or yellow appearance, and if the value of saturation is constant, colours with more hue and less brightness have greener appearance than colours with less hue and big brightness; conversely, colours with less hue and large brightness have yellower appearance than colours with more hue and less brightness.**

To measure colour differences between more hue and brightness and less hue and brightness, a similar experiment is designed to measure the colour difference between more hue and less saturation and less hue and more saturation. This experiment is divided into four steps:

Step 1: A group colour patches 'A' of identical hue are compared with same group colour patches 'B', but with two degree hue difference.

Step 2: The hue values of colours in the group 'B' are chosen to have 4, 6, 8, and 10 degree difference respectively, for comparing the colours in group 'A' as in step 1.

Step 3: The hue of first group is arbitrarily chosen from yellow to green; then the same experiment is done as above.

Step 4: The previous three steps are repeated to measure the colour difference with saturation other than 50.

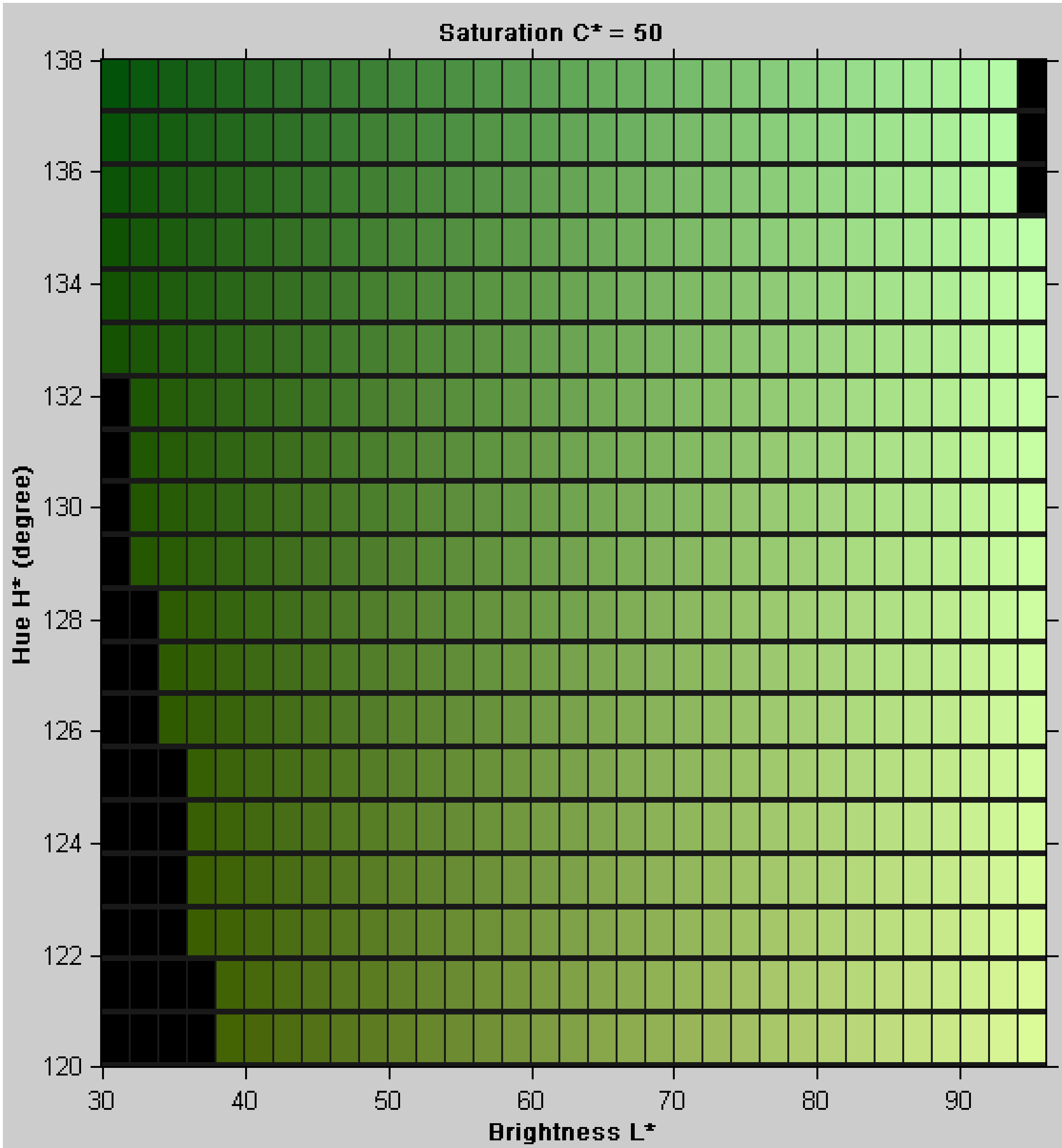


Figure 6-6 Colour patches with different brightness and hue as the saturation is 50

Table 6-6 lists the results of the experiment as five people were asked to measure the colour difference of colour patches in figure 6-6. The other results of colour difference measurement in the step 3 and 4 are almost same (More results are listed in the survey 4 of Appendix C). As the table shows, as the hue difference increases, the brightness decreases thus affecting the appearance of the hue. If the hue difference is more than 9, the brightness only slightly affects the hue appearance. According to the linear regression function 6-2, the three results with different hue of table 6-5 can be formulated as the equation 6-6:

	Hue	Brightness of colour patch with small hue is greener than 138 hue value (Saturation is 50)											
	138	90	86	82	78	74	70	66	62	58	54	50	46
Average	136	84	77	69	62	53	43	33	-	-	-	-	-
Average	134	76	63	51	-	-	-	-	-	-	-	-	-
Average	132	69	54	-	-	-	-	-	-	-	-	-	-
Average	130	55											

Table 6-6 Average testing results of threshold value of brightness with different hue

$$L_{Small\_Hue} = L_{Big\_Hue} - \frac{\Delta_{Hue} \times (96 - L_{Big\_Hue})}{2} \qquad (6\text{-}6)$$

where, L_{Big_Hue} is value of brightness at the range of [66, 90], and Δ_{Hue} is the hue difference between two compared colours. Since all results of this experiment are similar, the equation 6-6 can be generally applied when the saturation is constant. The dots in figure 6-5 are the results of the experiment for measuring colour patches with 2, 4, and 6 hue difference respectively, and the lines are drawn by using the equation 6-5 for the corresponding values of colours in MATLAB. We can state this result:

Result 14:

If the hue difference is more than 9 degrees, or if the brightness is less than 66, colours with more hue value always appear greener than colours with less hue value, independent of their brightness. If the hue difference is less than 9 degrees and the brightness is more than 66, then some colours with less hue and brightness are greener than colours with more hue and saturation. The brightness value of less hue colours increases, as the brightness value of more hue colours increases; and it is increasing as the hue difference increases. Equation 6-6 expresses the threshold brightness values of colours with less hue compared with colours with more hue. Colours which have smaller brightness value than the threshold value, L_{Small_Hue}, are greener than the colour with L_{Big_Hue}.

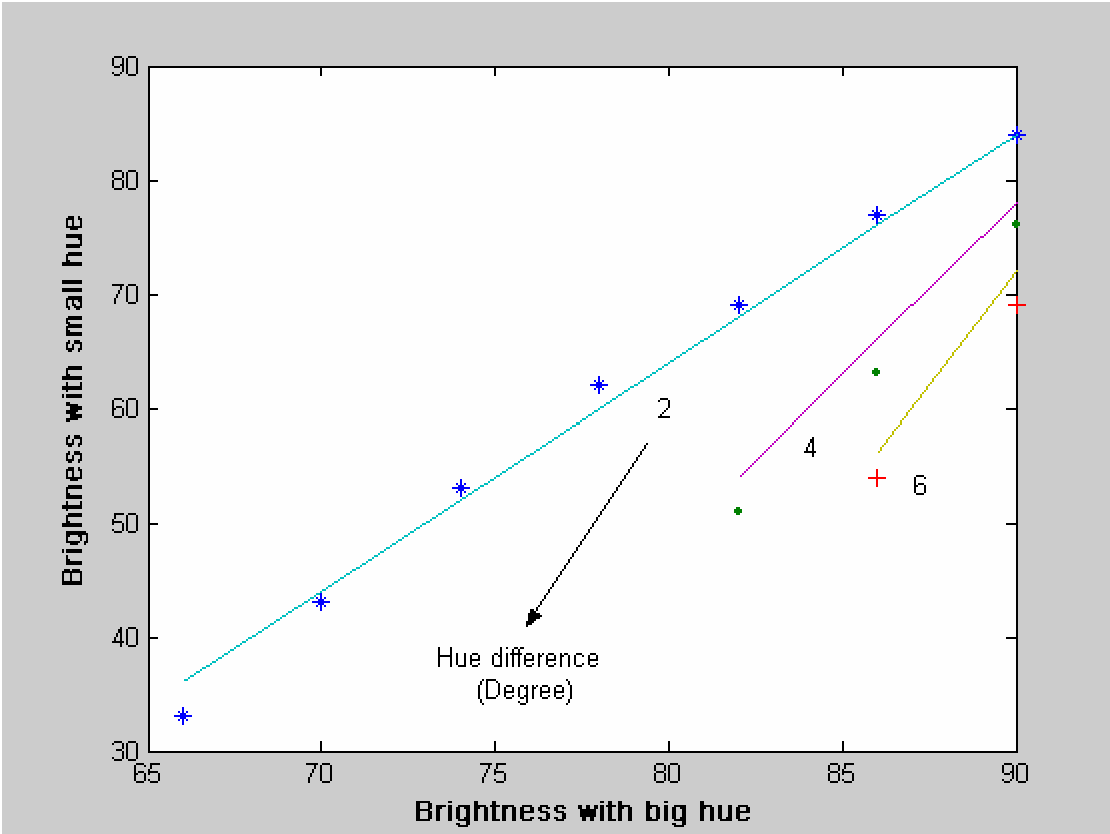


Figure 6-7 Lines of equation 6-6 obtained from the dots of experiment

6.6 Colour differences with varying brightness and saturation

All previous experiments involve varying hue. According to those experiments, the hue is the main factor in determining the colour appearance, but the brightness and saturation also affects the colour appearance if the hue difference is slight. Normally, small brightness or big saturation makes colours have greener appearance if the hue is constant; conversely, big brightness or small saturation makes colours have yellower appearance. Thus:

Result 15:

> **If colours have a green or yellow appearance, and if the value of hue is constant, colours with big saturation and small brightness have greener appearance than colours with small saturation and big brightness;**

conversely, colours with small saturation and big brightness have yellower appearance than colours with big saturation and small brightness.

There remains one condition to measure, in which a colour with big brightness and big saturation is compared with a colour with small brightness and small saturation, with the hue being constant. Figure 6-8 lists such colour patches of 125 degree hue as the brightness and saturation have the four units steps each. There are four parts to this experiment:

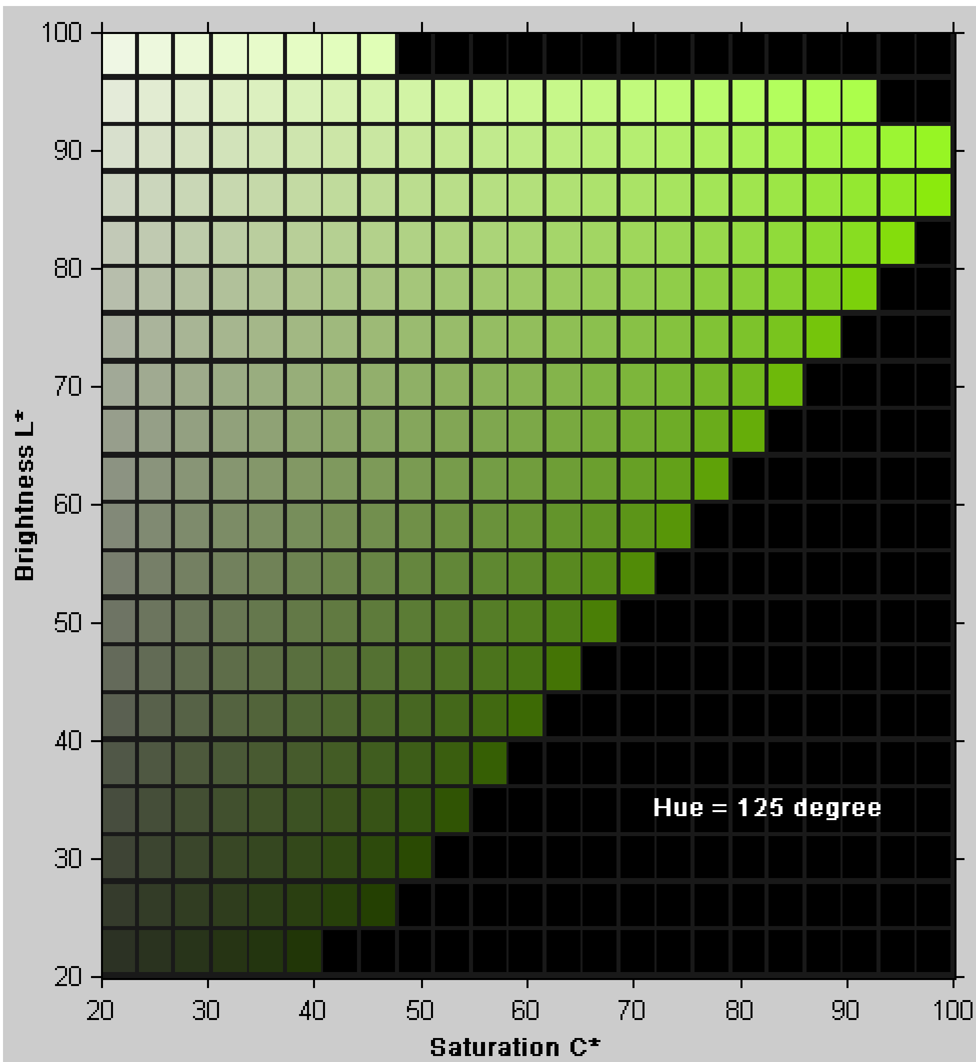


Figure 6-8 Colour patches with different brightness and saturation as the hue is 125 degree

Step 1: Colour patches which have saturation of 88 but different brightness in figure 6-8 are chosen as sample colours. Colour patches whose saturation is 84 are compared to the sample colours for determining their differences.

Step 2: Colours with different saturations which are 4, 8, 12, 16, and 20 units less respectively are chosen for comparison with the sample colour patches.

Step 3: Repeat step 1 and step 2 but using different saturation of the sample colour patches.

Step 4: Repeat parts 1, 2 and 3 for arbitrary hue of green and yellow.

Five people were asked to measure the colour difference of the colour patches. Table 6-7 lists the results of step 1 and step 2 using the colour patches shown in figure 6-8.

As table 6-7 shows, the values of brightness vary almost linearly as the saturation varies. Comparing with other data of this experiment, they almost have the same results. This result can be expressed by equations 6-7 and 6-8.

In conclusion, the results of this experiment can be represented as below:

	S	Brightness of colour patches with small S is greener than sample patches											
Sample	88	88	84	80	76	72	68	64	60	56	52	48	44
Average	84	83	77	71	64	58	53	47	41	35	30	23	-
Average	80	77	70	63	58	52	47	41	36	30	24		-
Average	76	71	65	59	53	48	42	35	29	24		-	-
Average	70	64	59	53	48	41	36	30	25		-	-	-
Average	63	58	52	45	39	34	29	22		-	-	-	-

Table 6-7 Average results of threshold values of brightness with different saturation

Result 16:

If green or yellow colours have constant hue value, and conditions of result 13 do not apply the brightness and saturation affect the colour appearance linearly. Their relationship can be expressed by the equations 6-7 and 6-8. Equation 6-7 expresses the minimum saturation for which colours with small brightness are greener than

colours with big brightness. Equation 6-4 provides the threshold ratio of saturation difference and brightness difference between two colours to distinguish their hue appearance; when the fraction of saturation and brightness is more than 2/3, the colour with big saturation and big brightness is greener than the colour with small saturation and small brightness. It is clear that the saturation is more sensitive than brightness as for human measurement of hue appearance. (This does not contradict with the result 8 and 9; since humans can distinguish a smaller difference of brightness than saturation, but a small change of brightness doesn't change the hue appearance.)

$$S_{Small\_L} = S_{Big\_L} - \frac{2}{3} \times \Delta_{Brightness} \tag{6-7}$$

$$\frac{\Delta_{Saturation}}{\Delta_{Brightness}} = \frac{2}{3} \tag{6-8}$$

where, $\Delta_{Saturation}$ is the saturation difference of two colours, and $\Delta_{Brightness}$ is the brightness difference of two colours.

6.7 Colour difference under any condition

So far, we have obtained 15 results from different experiments. Each result applies only to some special condition, since each is obtained from the condition at least one of the **CIEL*C*H*** values is constant. However, all colours in above experiments are chosen specially to check the relationship in different colours. Normally, we would not expect that different colours have some special distributions. Colours in a digital image are affected by brightness, saturation, and hue at the same time in the **CIEL*C*H*** colour model. If two colours are compared, there are eight possibilities for their three values varying. Table 6-8 lists these eight possibilities if two colours A and colour B are measured. Note that in table 6-8, the states of all the values in condition 2 are opposite to those in condition 1. Similarly, condition 3 and 4 have opposite values, as do condition 5 and 6, and condition 7 and 8. Therefore, if the result of conditions 1, 3, 5, and 7 in table 6-8 are known, the colour difference with any condition can be known. According to previous results, conditions in table 6-8 can be analysed step by step as below (supposing that colours A and B mainly have green and yellow appearance):

In condition 1, firstly, differences of hue, brightness, and saturation are checked individually. Using the previous results, $H*_A < H*_B$ means colour B is greener than colour A; $L*_A > L*_B$ and $C*_A < C*_B$ also mean that colour B has greener appearance than colour A. Since all variables of colour B have the greener appearance than colour A, it is clear that colour B must have a greener appearance than colour A. Conversely, in condition 2, colour A is greener than colour B.

Condition	Hue difference	Brightness difference	Saturation difference
1	$H*_A < H*_B$	$L*_A > L*_B$	$C*_A < C*_B$
2	$H*_A > H*_B$	$L*_A < L*_B$	$C*_A > C*_B$
3	$H*_A < H*_B$	$L*_A > L*_B$	$C*_A > C*_B$
4	$H*_A > H*_B$	$L*_A < L*_B$	$C*_A < C*_B$
5	$H*_A > H*_B$	$L*_A > L*_B$	$C*_A > C*_B$
6	$H*_A < H*_B$	$L*_A < L*_B$	$C*_A < C*_B$
7	$H*_A > H*_B$	$L*_A > L*_B$	$C*_A < C*_B$
8	$H*_A < H*_B$	$L*_A < L*_B$	$C*_A > C*_B$

Table 6-8 Colour measurement with different condition

In condition 3, $H*_A < H*_B$ and $L*_A > L*_B$ means that colour B is greener than colour A, but $C*_A > C*_B$ means colour A is greener than colour B. Firstly, the hue difference between colour A and B is checked to determine whether it is more than 9 degrees or less. If the hue difference is more than 9 degrees, colour B is greener than colour A, whatever the other two values are. If it is less than 9 degrees, the following measurements have to be obtained. Colour B is taken to be a sample colour. According to the result 11 and equation 6-5, a threshold value *C** of saturation of colour with $H*_A$ and $L*_B$ can be obtained comparing with the colour B. A new colour P with saturation C*, $H*_A$ and $L*_B$ has the same colour appearance as colour B. Therefore, the result of measuring the colours A and P is the same as the result of measuring the colours A and B. Since colour

P has the same hue value as the colour A, according to the result 14, if $C*_A < C*$, colour P is greener than colour A because the hue values of two colours are the same and $L*_A > L*_B$; if $C*_A > C*$, then the ratio of the saturation difference and the brightness difference between colours A and P has to be measured to determine their appearance according to the result 15. If the ratio is less than 2/3, colour P is greener than colour A, also colour B is greener than colour A; otherwise colour A is greener than colour B. In condition 4, the opposite situation applies.

In condition 5, $H*_A > H*_B$ and $C*_A > C*_B$ means that colour A is greener than colour B but $L*_A > L*_B$ means that colour B is greener than colour A. If the hue difference is more than 9 degrees, colour A is greener than colour B because the hue difference is too big for brightness and saturation to affect the colour. If the hue difference is less than 9 degrees, then there are two steps to measure their difference. First, the saturation of the two colours is assumed to be the same. According to result 13 and equation 6-6, a threshold value $L*$ of brightness with $H*_B$ and $C*_A$ can be obtained by comparing with colour A. A colour P with the value of such brightness $L*$, $H*_B$, and $C*_A$ has the same colour appearance as the colour A. Therefore, the result of measuring the colours P and B is same as the result of measuring the colours A and B. Since the hue of colour P and B is the same, their colour difference can be measured by the result 15 and 16. According to the result 15, if the $L*$ is less than $L*_B$, colour P is greener than colour B; and so colour A is greener than colour B. If the $L*$ is more than $L*_B$, the ratio of saturation difference and brightness difference between colours P and B must be determined according to result 16 and equation 6-8. If the ratio is more than 2/3, colour P is greener than colour B; and so colour A is greener than colour B too. Otherwise, colour B is greener than colour A. The result of condition 6 is opposite to condition 5.

In condition 7, $H*_A > H*_B$ means that colour A is greener than colour B but $L*_A > L*_B$ and $C*_A < C*_B$ means that colour B is greener than colour A. If the hue difference is more than 9 degrees, colour A is greener than colour B regardless of the value of brightness and saturation. If the hue difference is less than 9 degrees, there are two steps similar to above. First, the colour A is considered as a sample colour. According to result 12, a threshold value C* of saturation can be obtained. If C* is less than $C*_B$, colour B is greener than colour A. Otherwise, suppose that a colour P has the values $C*$, $L*_A$, and $H*_B$. Since the colour P has the same appearance as colour A and colour P and colour B have the same hue value, the ratio of saturation and brightness can be obtained according to the result 15 and equation 6-8. If such ratio is more than 2/3, colour P is greener than colour B; and so colour A is greener than colour B. Otherwise, colour B is greener than colour A. For the condition 8, the results are opposite to those of condition 7.

So far, the measurement of two colours has been investigated for all possible values. According to the previous results and equations, given two colours, it can be determined which one is greener. In conclusion, a colour model which measures colour differences to determine their green or yellow appearance can be summarized as follows:

Colour difference model:

If two colours A and B which have a green or yellow appearance, their brightness, saturation, and hue are expressed by $L*_A$, $L*_B$, $C*_A$, $C*_B$, $H*_A$, and $H*_B$ respectively, the colours can be measured as the following steps:

1. Check the hue difference, $\Delta_{Hue} = |H_A - H_B|$, between these two colours: if the Δ_{Hue} is more than 9 degree, the colour with big hue value is greener than the colour with small hue value.
2. If the Δ_{Hue} is less than 9 degrees, the saturation and brightness difference have to be checked. Supposing $H_A > H_B$, if both $L*_A < L*_B$ and $C*_A > C*_B$, then colour A is greener than colour B. Otherwise, we apply one of the following steps.
3. If $L*_A < L*_B$ but $C*_A < C*_B$, a temporary value of saturation $C*$ can be obtained as below:

   $$C* = 0.72 \times \Delta_{Hue} \times (C*_A - 5)$$

   if $C* > C*_B$, then colour A is greener than colour B.

if $C^* < C^*_B$, check the ratio $R = \frac{C^*_B - C^*}{L^*_B - L^*_A}$. If $R > 2/3$, colour B is greener than colour A; otherwise, colour A is greener than colour B.

4. If $L^*_A > L^*_B$ but $C^*_A > C^*_B$, a temporary value of brightness L^* can be obtained as below:

$$L_{Small\_Hue} = L_{Big\_Hue} - \frac{\Delta_{Hue} \times (96 - L_{Big\_Hue})}{2}$$

if $L^* < L^*_B$, then colour A is greener than colour B

if $L^* > L^*_B$, check the ratio $R = \frac{C^*_A - C^*_B}{L^* - L^*_B}$. If $R > 2/3$, colour A is greener than colour B; otherwise, colour B is greener than colour A.

5. If $L^*_A > L^*_B$ and $C^*_A < C^*_B$, a temporary value of saturation C^* can be obtained as below:

$$C^* = 0.72 \times \Delta_{Hue} \times (C^*_A - 5)$$

if $C^* < C^*_B$, then colour B is greener than colour A.

if $C^* > C^*_B$, check the ratio $R = \frac{C^* - C^*_B}{L^*_A - L^*_B}$. If $R > 2/3$, colour A is greener than colour B; otherwise, colour B is greener than colour A.

6.8 Conclusion

Colour appearance is mainly determined by the value of hue although the brightness and saturation may affect it as well. If the hue difference is more than 9 degrees, the affect of brightness and saturation is too small to affect the colour appearance; colours can be easily distinguished by their hue difference to determine their difference. If the hue difference is less than 9 degrees, brightness and saturation become the main factor in determining the colour appearance. Basically, the values of brightness and saturation inversely affect the colour appearance; the more saturation, the greener the colour, and less brightness, the greener the colour. The final colour difference model can measure the colour difference between any two colours if the hue difference is less than 9 degrees. This model only applies to colours having a green or yellow appearance.

Chapter 7

Windows application of the colour difference model

7.1 Introduction

A green vegetable mainly has green and yellow appearance, and its quality can be determined by the level of the green and yellow. A good quality vegetable has a highly saturated green appearance; a lower quality vegetable will have a low saturated green or a yellow appearance. In this chapter, a Windows application using Visual Basic is designed to measure the colour difference for the digital image of a green vegetable. The colour difference model in the last chapter is the main algorithm and method used to measure the quality.

Visual Basic (VB) is very powerful for the design of GUIs, although its speed of execution is not fast as C++. However, it can be used to easily create buttons, textboxes, pop-up menus, and other functions usually used in Microsoft Windows. Current PCs are fast enough so that the speed difference between VB and other compiled languages can be ignored for this project. Moreover, the functions of C++ can be called by VB if necessary for high speed.

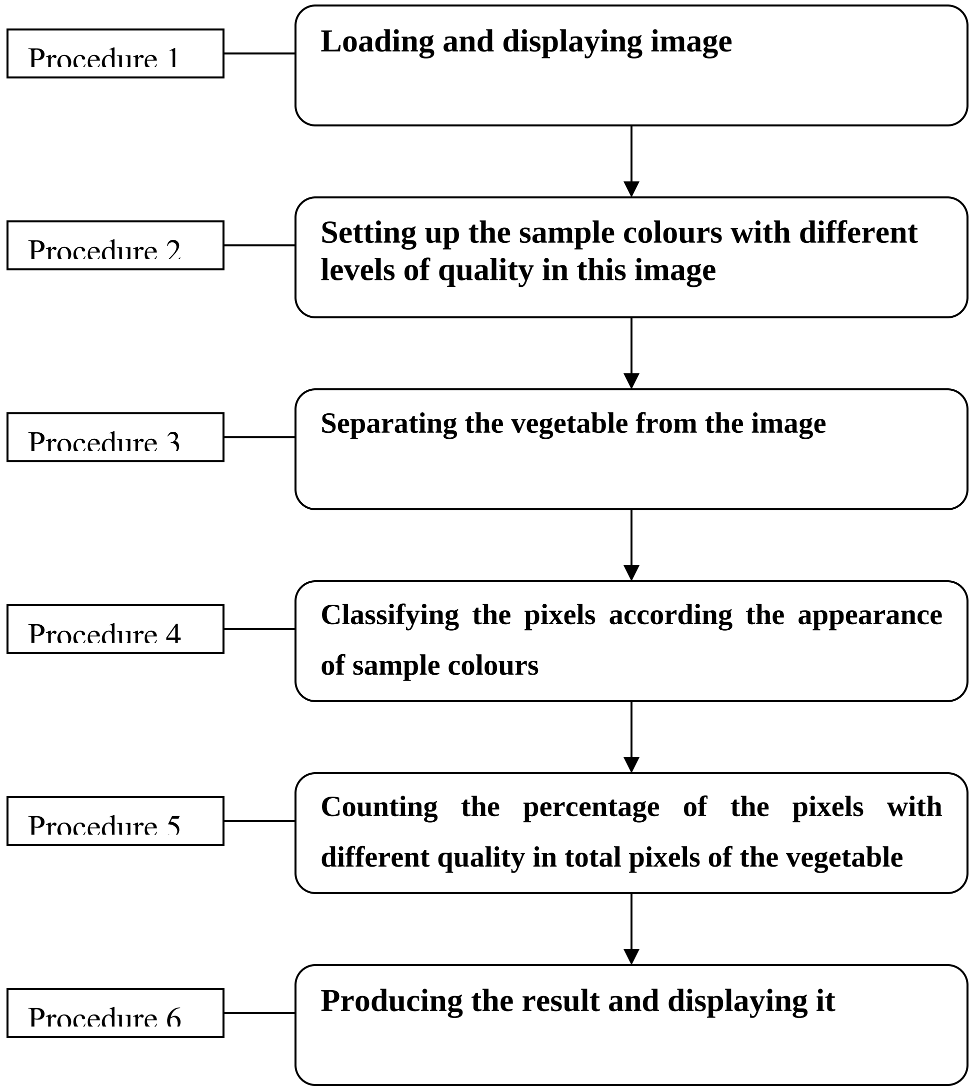


Figure 7-1 The procedures of the application

7.2 Analysis and design

A digital image consists of pixels. The resolution of an image is determined by sampling, and it is expressed by the numbers of pixels in each row and column. An image with high resolution has better quality than an image with low resolution; it also includes more pixels. Image processing involves calculation using the information of each pixel. For a colour image, each pixel has red, green, and blue values to describe the colour appearance of such pixel (see chapter 3). These values in the **RGB** colour model must be converted to the corresponding values in the **CIEL*a*b*** or the **CIEL*C*H*** colour models. According to the colour difference model developed in chapter 6, each pixel with a green or yellow appearance can be classified to different levels of colour appearance. The percentage of pixels with different levels in the whole vegetable can be used to measure the quality of the vegetable in the image. This application has six main procedures and the work flow diagram is given in Figure 7-1.

7.2.1 Procedure 1

Normally, the image file should be 24-bit uncompressed format, such as BMP and TIFF files (See chapter 3). A 24-bit colour image provides enough detail of colours for human perception; and the uncompressed image can avoid losing the details of colours as the image file is converted or loaded. Although the JPEG image format uses a transform compression algorithm where some information is lost to achieve high compression rates, the resulting image is generally indistinguishable from the original. [Gon02] Thus we may allow colour measurements on JPEG image files.

The resolution of the image determines the image size. All images in this application are kept at their original size because details will be lost if the image size is reduced. However, some images are too big to fit in the screen; it is not convenient to observe and measure the colour of these images for the user. Therefore, input images are resized to fit in a fixed area if they are too big and kept at their original size if they can be displayed in the fixed area. However, resizing is for displaying only, and the colour measurement is still obtained from the image of its original size. In Visual Basic, there are built-in functions to read the size of the image, resize the image, and display the image.

7.2.2 Procedure 2

Different kinds of vegetables have difference colour appearance for their quality. For example, the high quality broccoli usually has a dark green appearance, whereas high quality cabbage has a light green appearance. Even for the same kind of vegetable with same quality, their colour appearance could be different because of the influence of the environment and surroundings. In this application, there are five sample colours that are compared to the colour of each pixel of the input image to measure six different quality grades, which are designated ‘Excellent’, ‘Very good’, ‘Good’, ‘Fair’, ‘Poor’, and ‘Very poor’ respectively (see Figure 7-2). These five sample colours have difference colour appearance; their default values range from green to yellow. However, their values are not constant and can be modified by a pop-up submenu. This allows the user to choose suitable colours for the vegetable in the input image. If the default values of the sample colours are not suitable for measuring this vegetable, the user can change these values. For convenience, the values of the sample colours are set up in the **CIEL*a*b*** colour model because it is intuitive for human perception and easy to change.

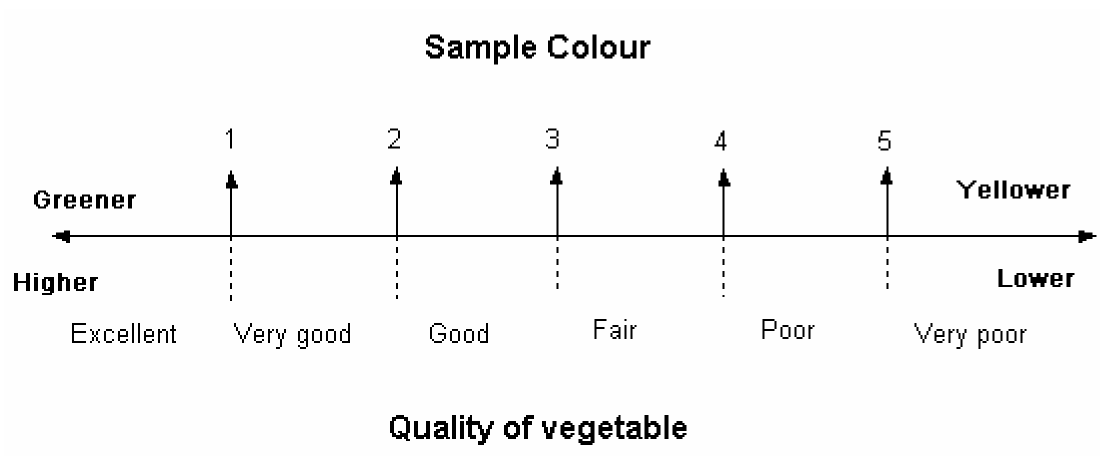


Figure 7-2 Quality of vegetable and the sample colours

The colour gamut of the **CIEL*a*b*** colour model is bigger than the gamut of the **RGB** colour model, so some input values of **CIEL*a*b*** could be out of the **RGB** gamut, which the RGB intensity imcposes a limit on the darkness mesaured. Although the colour difference model uses the **CIEL*C*H*** colour model to measure the colour difference, only the colours with corresponding **RGB** values are measured. Since colours which are out of the **RGB** colour model have not been measured in the previous experiment and these colours can not be displayed on the CRT monitor, the result of the measurement could be incorrect if the sample colour is chosen out of the **RGB** colour model. Basically, the R, G, and B values of 24-bit **RGB** colour model are from 0 to 255. As the values of

CIEL*a*b* or **CIEL*C*H*** are converted to the **RGB** values, some values are out of this range. Values are less than zero are defined as zero, and values are more than 255 are considered as 255. Therefore, different colours in **CIEL*a*b*** could be mapped to the same colours as they are converted to the **RGB** colour model. In this application, to avoid colours which are out of the gamut of **RGB** being chosen as sample colours, the input colours with **CIEL*a*b*** values are checked to determine whether they are located in the **RGB** gamut. If users enter a colour value with **CIEL*a*b***, which is out of the **RGB** colour gamut, the values of this colour are not accepted by this application; and users are requested to enter new values.

7.2.3 Procedure 3

There are many factors to affect the quality of image, such as light source, camera lens, and view angle. In this thesis, the image of vegetable is supposed to be created in ideal condition, i.e. the light source is sunlight; the details of vegetable are described well; the surface of the vegetable doesn't have any shadow. Moreover, the background of the image is predefined as white or light blue because the red colour background will cause the foreground has more red hue and green background will cause the foreground has more green hue. The hue of vegetable is mainly focused on the green and red-yellow area and the background with such colours will affect the correctness of hue of vegetable. Another reason to choose the white or light blue as background is that the vegetable segmentation is easily implemented. If the background of image satisfies this condition, the vegetable can be separated from the image by using the results 1, 3, and 4.

To separate the vegetable from the image, the value of brightness, saturation, and hue can be considered as independent factors. In practice, as humans measure the quality of vegetable, the vegetable must be bright enough; otherwise, the object is too dark to distinguish the colours, and it will be impossible to measure the quality of the vegetable. According to Result 1, colours whose brightness is less than 10 are too dark for their appearance to be measured; such colours can be considered as background. Moreover, if the colour is too pale, the hue appearance can't be determined. Such colours only have different levels of black and white; thus they can't belong to the colour of vegetable. Also, according to Result 3, if the saturation of colour is less than 10, such colours can be considered as the background. Finally, since the background of the image may be assumed to be different from green and yellow, any colour for which the hue is located in the angle value between green and yellow can be considered as the colour of the vegetable; otherwise, the colour forms the background. The hue value of pure green is about 136 degrees and of pure yellow is about 103 degrees. In this application, the hue range of the vegetable is chosen to be between 90 and 150 degrees. The steps of the vegetable separation from the background are as follows:

1. *Loading the image*
2. *Reading the RGB value of pixel*
3. *Converting the RGB value to values of CIEL*a*b* and CIEL*C*H* colour model*
4. *For each pixel: if $L^* < 10$, the pixel is background because it is too dark to distinguish*
5. *else if $C^* < 10$, the pixel is background because it is too pale*
6. *else if $H^* < 90$ degrees or $H^* > 150$ degrees, the pixel is background because the hue is outside the admissible range of hues for a green vegetable.*
7. *Otherwise, the pixel is part of the vegetable.*

7.2.4 Procedures 4 and 5

This part is the core of this application. As the vegetable is successfully separated from the image, only the colours of vegetable are measured. According to the colour difference model in the last chapter, each colour of the pixels of the vegetable is compared with the sample colours to determine which one is greener. For example, suppose that colour A, B, C, D, and E are the five samples of colour from greenest to yellowest appearance. First the vegetable's colours are compared with the colour A. If the colour of the pixel of the vegetable is greener than colour A, then the colour represents 'Excellent' quality. Otherwise, the vegetable's colour is compared with colour B. If it is greener than colour B, it has 'Very good' quality; if not, then measure the difference with colours C, D, and E. There are six counters to record the numbers of pixels for the corresponding grade of quality. As the colour of a pixel is used to determine its grade of quality, its corresponding counter is increased by one. Thus, the sum of the values in the six counters is the total number of pixels of the vegetable. The percentage of numbers of each grade quality in the total pixel number represents the percentage of such colour in the vegetable. The formula (7-1) is the general expression of each quality, where *'n'* means the grade of the quality.

$$Quality_n = \frac{\sum Pixel_n}{\sum_{n=1}^{n=6} \sum Pixel_n} \tag{7-1}$$

In Visual Basic, the measurement of the difference of two colours is defined by a function, *'ColourDifference(L1,C1,H1,L2,C2,H2)'*, where the parameters of the function are the values of **CIEL*C*H*** of two colours. This function is created according the colour difference model in chapter 6; the code is listed in Appendix D. The function returns two values, *'true'* and *'false'*, which *'true'* means the first colour is greener the second one, and *'false'* means the second colour is greener than the first one.

Following the steps in procedure 3, in which the vegetable is successfully separated from the image, the workflow of measuring quality of each pixel is listed as below:

1. *Measure the colour difference between the colour of this pixel and the sample colour 1*
2. *If this pixel has greener appearance than the sample colour 1*
   Increase the counter of Grade1 by one
3. *Otherwise, measure the colour difference between the colour of this pixel and sample colour 2*
4. *If this pixel has greener appearance than the sample colour 2*
   Increase the counter of Grade2 by one
5. *Otherwise, measure the colour difference between the colour of this pixel and sample colour 3*
6. *If this pixel has greener appearance than the sample colour 3*
   Increase the counter of Grade3 by one

7. *Otherwise, measure the colour difference between the colour of this pixel and sample colour 4*
8. *If this pixel has greener appearance than the sample colour 4*
   Increase the counter of Grade4 by one
9. *Otherwise, measure the colour difference between colour of this pixel and sample colour 5*
10. *If this pixel has greener appearance than the sample colour 5*
    Increase the counter of Grade5 by one
11. *Otherwise, increase the counter Grade6 by one*
12. *Add the value of different counters to obtain the total number of pixels of the vegetable*
13. *Obtain the percentage of the value of each counter in the total number pixels of the vegetable*

Since each pixel of the image has to be measured and some pixels will require more than one measurement, especially if they are yellow, it will take long time to process a large size image. For example, if the size of the image is 800 x 600, there are 480,000 pixels to measure.

7.2.5 Procedure 6

After the colour of each pixel has been measured, the result of each grade of quality has to be displayed. In this application, there are two kind results of output, text and graphic results. The text result displays the percentage of six grades of quality. The graphic result redefines the colours of the input image according to the grades of quality and background and creates a new image. In the new image, background is defined as black; the vegetable has the same position as the vegetable of the input image. There are only six colours in the vegetable of the new image, which are from green to yellow to represent the different qualities.

In Visual Basic, as the input image is loaded, a new image with same size is created. According to the result of measuring each pixel in the original image, the pixel at the same position of the new image is set to an identical value to represent the corresponding grade of quality. As the measurement of the original image is finished, the new image has been created. Since the new image has the same size as the original one, some images are too big to fit in the whole screen. As in procedure 1, such new images must be resized to fit in a fixed area to display.

The steps are listed as below:

1. *As the original image is loaded, create a new image with the same size*
2. *If the colour of pixel is background, set the colour of the pixel in the same position of the new image to black*
3. *If the colour of pixel has the 'Excellent' quality, set the RGB value of the pixel in the same position of the new image to (0, 127, 0).*
4. *If the colour of pixel has the 'Very good' quality, setting the RGB value of the pixel in the same position of the new image to (0, 192, 0).*
5. *If the colour of pixel has the 'Good' quality, set the RGB value of the pixel in the same position of the new image to (127, 255, 127).*
6. *If the colour of pixel has the 'Fair' quality, set the RGB value of the pixel in the same position of the new image to (127, 127, 0).*
7. *If the colour of pixel has the 'Poor' quality, set the RGB value of the pixel in the same position of the new image to (255, 255, 127)*
8. *If the colour of pixel has the 'Very poor' quality, setting the RGB value of the pixel in the same position of the new image is (255, 192, 127).*
9. *Check the size of the new image. If it is bigger than the fixed area of the output image, resize the image to fit the fixed area; otherwise, keep its size.*
10. *Display the new resized image.*

7.3 Interface of the application

The main functions and procedures of this application have been described in previous sections. To achieve these functions, the interface of the application is created as in Figure 7-3. The image is loaded from the 'File' menu and is displayed in the 'Input Picture' frame. Five sample colours are initialized as the application is executed. Their values are expressed using the **CIEL*a*b*** colour model and their corresponding colour appearances are displayed beside their definitions. As the 'Define' button is clicked, a pop-up menu appears. The values of sample colours can be modified using the **CIEL*a*b*** colour model and the **RGB** colour model. However, only colours located in the **RGB** gamut are accepted; otherwise, the value has to be entered again. The function of the 'Result' button is to measure the quality of the input image according to the five sample colours. The text results are displayed under the 'Result' button and the graphic result is displayed in the 'Output Picture' frame.

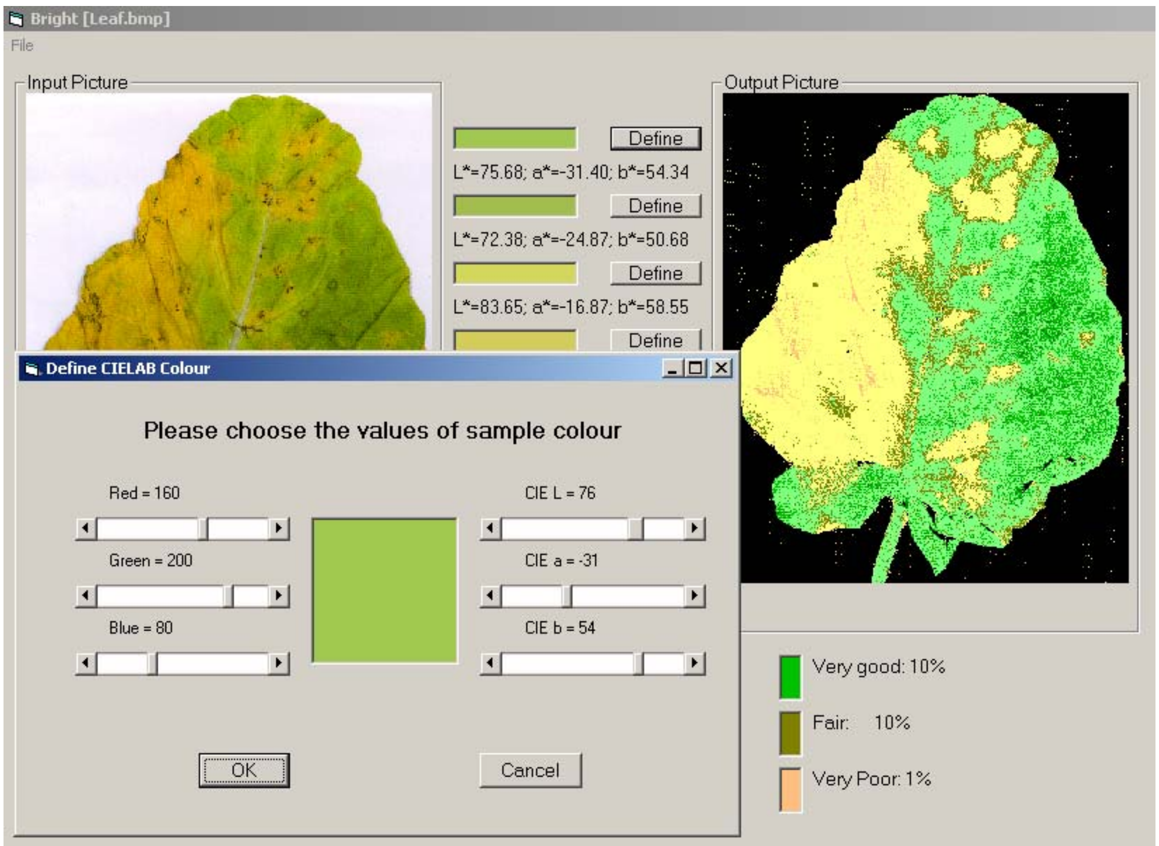


Figure 7-3 Interface of the application

7.4 The correctness of the result

As we have discussed in chapters 2 and 3, although the appearance of colours is very subjective, colours only exist in the brain. The best way to check the correctness of the result is to compare with the measurement by the human eye. In the figure 7-4, the top-left image is the vegetable which is to be measured, and with the default sample colours.

The top-right image is the graphic result and the bottom figure is the text result. This computer result conforms to the measurement by human eye, asking five people to measure the original image. More green-yellow vegetable images are measured to compare the results between humans and computer. This demonstrates that the colour difference model compares favorably to human perception.

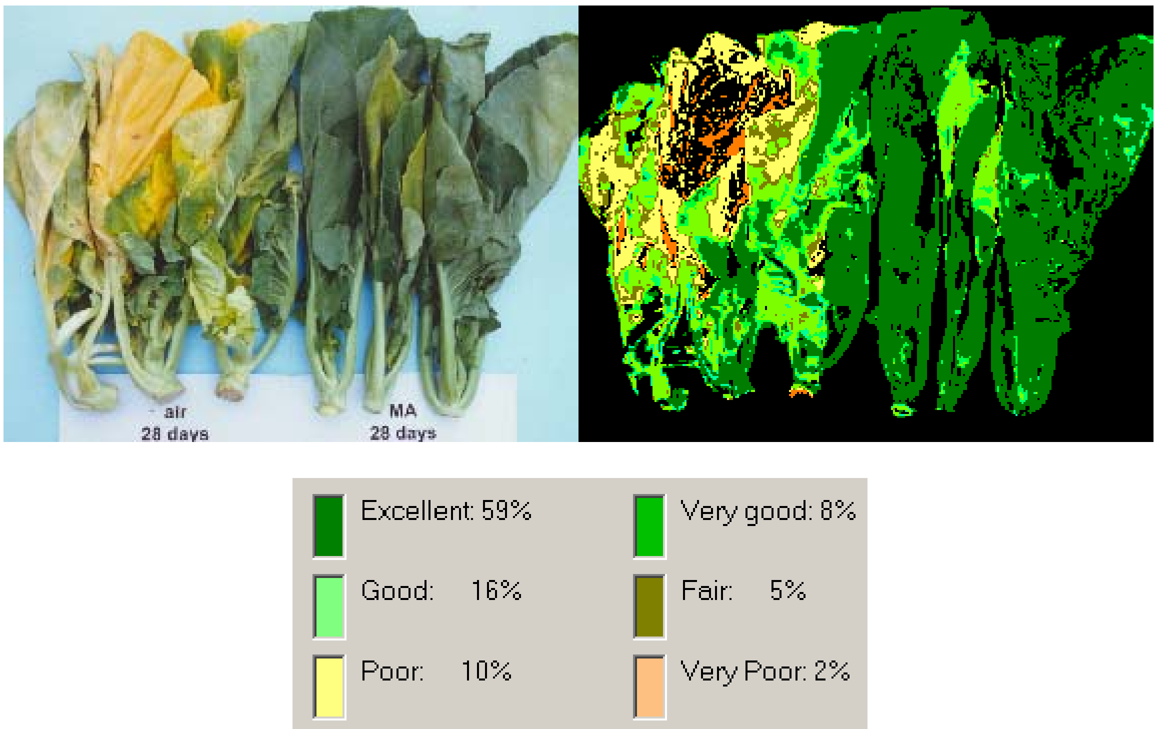


Figure 7-4 Original image and the output of graphic and text

7.5 The range of the application

The colour difference model is an empirical result, which we have concluded from different experiments. It mimics the human eye in measuring the colour difference within the green and yellow range. Normally, the human eye can measure the colour very well under complicated conditions. However, the measurement of human eye provides only an approximate result. In some situations, this is not accurate enough to detect the difference. For example, to check the quality of vegetable in a fixed environment as the time and temperature vary, this application will give more accurate details than the measurement of the human eye. In fact, the main function of this application is to sort the colours within the green and yellow range.

7.6 Conclusion

This windows application is created by Visual Basic. The yellow green grading is based on human (5) selected variables. It mimics the human eye by classifying colours within the green and yellow range according to their appearance. In comparison of other tolerance to shifts in HSV/RGB indices by systematic variation of the boundaryies for the grades, the proposed algorithm is more sensitive for measure green vegetable. Normally, it can be used to measure the quality of green vegetable; it also can be applied to sort the green and yellow colours in other areas. It simply needs a digital camera to get the image of vegetable. So far, this application can only distinguish the colour in green and yellow range very well. However, the functions and the interface still can be advanced and developed if the users have the further demands. The source code of the application is listed at the Appendix D.

Chapter 8

Conclusion and future work

8.1 Conclusions

This thesis is based on a project which is sponsored by the Institute for Horticultural Development, Agriculture Victoria. This project involves computer science, image processing and colour science. The main purpose of the project is to use digital imaging techniques to measure the quality of green vegetables according to their colours. Since interpretations of colours are too subjective to be described precisely, there is no single universal standard to represent them under all possible conditions; for example with varying lighting and background. There are many colour models and colour formulas; however, they are defined only for special conditions. There are also many restrictions and problems as the colours are reproduced: film and colour TV can not reproduce colours to be viewed naturally in bright light, for example, sunlight; and colour printers and digital cameras reproduce the colours less than natural colours because of the colour gamut.

Fortunately, colours can be represented correctly under some special conditions. Firstly, perception of colour is generally independent of the observer. [Wys67] This means that the colour appearance can be described correctly if enough people and experiments are undertaken. Secondly, the ability of human eye to distinguish colours is limited. If the difference between two colours is too small, the human eye cannot distinguish them. The **CIE XYZ** colour model is the basis of current colour science, this model is based on the results of human observation. [Wys67] Many colour models and colour formulae have been created based on **CIE XYZ** for different applications. From **CIEXYZ**, **CIEL*u*v***, **CIEL*a*b***, **CIEL*C*H***, **CIE94**, **CIECAM97s**, to the recent colour difference equation **CIEDE2000**, CIE have attemped to create a complete and universal formula or model to describe any colour for any situation. Unfortunately, all these colour models and formulae are limited to specific applications. [Joh03] [Kue03]

In this project, the computer is used to measure the quality of green vegetables according to human perception, which classifies the green and yellow colours according to their colour appearance. None of the current colour models and colour difference formulas can reproduce or match a given naturally occurring colour very well. In order to create a model to mimic human perception, experiments have been designed to analyse the colour difference according to the variations of brightness, saturation, and hue. With different conditions, brightness, saturation, and hue affect human perception together. A final colour difference model was developed based on these experiments. Since measurement of vegetable quality is subjective and empirical, based on this model (for green and yellow), vegetables can be measured to determine the quality very well. Comparing two colours within the green and yellow range, this colour difference model can determine whether a colour is mostly green or mostly yellow in human perception with different enviroments. A Windows application has been designed based on **RGB**, and **CIEL*a*b*,** and this colour difference model is used to measure the quality of vegetable in colour images; there are two kinds of output results; text and graphical. So far, this application has achieved the aim of the project to evaluate the quality of vegetable and the sponsor satisfies the result based on this model.

8.2 Future work

The sensitivity of the human eye to different colours varies according to the colour's wavelength. Normally, if the colours only differ in hue, the human eye can distinguish 10 nm wavelength hue differences at 480 nm (blue) and 2 nm wavelength hue differences at 580 nm (yellow). [Fol96] In this project, the focus is on the measurement of green vegetables; the experiments are only designed to measure the colours within the green and yellow range. Hence, the colour difference model is mainly applicable to green and yellow colours. For measurement with purple, blue, and cyan colours, further experiments would be required to collect data and analyse it.

Although the experiments have used the **CIEL*C*H*** colour model, only the colours in 24-bit **RGB** colour gamut are measured because standard image formats and display equipment does not support the **CIEL*C*H*** colour model. In fact, 24-bit **RGB** can only represent a limited number of natural colours. Also, some green and yellow colours have not been measured in this project because they can't be displayed on a CRT monitor. To measure all possible colours, display equipment must be chosen according to the values of **CIEL*C*H***.

A colour difference model which can measure all visual colours can be applied in more areas; for example:

- Real-time monitoring of the ripeness of agricultural products,
- Real-time checking the quality of agricultural products as they are stored or transported, using images transferred by a video monitor,
- Analysing aerial images of forest, sea, and land for different purposes,
- Analysing the colour differences in the textile and decorative industries.

References

Note: Much information about modern colour spaces is not yet in print, and exists only in on-line documents.

[Ado00] Adobe, *Color Space*, Adobe Systems Incorporated, 2000
http://www.adobe.com/support/techguides/color/

[Bra98] G. J. Braun, F. Ebner, and M. D. Fairchild, *Color Gamut Mapping in a Hue-Linearized CIELAB Color Space*, IS&T/SID 6th Color Imaging Conference, Scottsdale, 163-168, 1998

[Bri02] Michael H. Brill, *Chromaticity Contour Map of the RGB Cube: A Simple Algorithm,* Color Research and Application, Volume 27, Number 6, December 2002

[Bro95] C. Wayne Brown and Barry J. Shepherd, *Graphics File Formats Reference and Guide*, Manning, 1995

[Buc03] G. Buchsbaum, *An analystical derivation of visual nonlinearity*, IEEE Electron, Lett, Vol 39, no. 2, pp 431-432, 2003

[Burns] Joe Burns, *Image Formats on the Web,*
http://www.htlmgoodies.com/tutors/image_formats.html

[Cas96] Kenneth R. Castleman, *Digital Image Processing*, Prentice Hall, 1996

[Che89] P. Chen, M.J. McCarthy and R. Kauten, *NMR for internal quality evaluation of fruits and vegetables*. Trans. ASAE **32** (1989), pp. 1747–1753.

[Cui00] Guihua Cui, M. Ronnier Luo, B. Rigg, Wei Li, *Colour-Difference Evaluation Using CRT Colours, Part 1: Data Gathering and Testing Colour Difference Formulae,* Color Research and Application, Volume 26, Number 5, October 2001

[Fai01] M. D. Fairchild, *A Revision of CICAM97s for Practical Applications*, Color Research and Application, Volume 26, Issue 6, Date: December 2001, Pages: 418-427

[Fai98] M. D. Fairchild, *Color Appearance Spaces*, Addison-Wesley, Reading , Mass 1998

[Fairc] Mark D. Fairchild, *Status of CIE Color Appearance Models,* http://www.colour.org/tc8-01/MDF_AICpaper.pdf

[Fol94] J. Foley, A. van Dam, S. Feiner, J. Hughes, and R. Phillips, *Introduction to Computer Graphics*, Addison-Wesley, 1994

[Fol96] J. Foley, A. van Dam, S. Feiner, and J. Hughes, *Computer Graphics: Principles and Practice*, Second Edition, Addision-Wesley, 1996

[For98] A. Ford and A. Roberts, *Colour Space Conversions*, August 11, 1998 http://www.inforamp.net/~poynton/PDFs/coloureq.pdf

[Frequ] *Frequently Asked questions about Colour Physics* http://www.colourware.co.uk/cpfaq

[Gon00] Jay Gondek, *An Extended sRGB for High Quality Consumer Imaging,* Hewlett-Packard, Revision 1.01 6/20/2000

[Gon02] Rafael C. Gonzalez, Richard E. Woods, *Digital Image Processing*, Prentice Hall, 2002

[Gri02] Lewis D. Griffin, Arsalan Sepehri, *Performance of CIE94 for Nonreference Conditions,* Color Research and Application, Volume 27, Number 2, April 2002

[Gur98] K. Gurevicius, *CRT display calibration of psychophysical color measurements*, Master's Sc. Thesis of University of Kuopio, 1998
http://members.tripod.com/~Pupikas/msc_thes/thesis.htm

[Hao97] Hao Shi, *Colour Segmentation for Broccoli Quality Inspection*, Digital Image & Vision Computer, Techniques & Applications, Auckland, NZ 1997, Eds Wyatt H. Page & Bob I. Chaplin, PP545-550

[Hea94] D. Hearn and M. Baker, *Computer Graphics*, Second Edition, Prentice Hall, 1994

[Heggi] David Heggie, *The CIE 1994 Colour Difference Model,*
http://www.colorpro.com/info/data/cie94.html

[Hil90] F. S. Hill, *Computer Graphics*, Macmillan Publishing Company, 1990

[Hoe71] Paul G. Hoel, *Introduction to Mathematical Statistics*, New York, Wiley 1971

[Hof03] Gernot Hoffmann, *CieLab Color Space*, 2003
http://www.fho-emden.de/~hoffmann/cielab03022003.pdf

[Hun95] R.W.G. Hunt, *The Reproduction of Colour*, Fifth Edition, Fountain Press, 1995

[IEC98] IEC, *Colour Measurement and Management in Multimedia Systems and*

*Equipment - Part 2-1: Default RGB Colour Space – sRGB***,** 1998
http://www.colour.org/tc8-05/Docs/colorspace/

[IEC99] IEC, *Multimedia Systems And Equipment Colour Measurement And Management Part 2-2: Colour management Extended RGB Colour Space – sRGB64* 1999
http://www.colour.org/tc8-05/Docs/colorspace/

[Jai89] Anil K. Jain, *Fundamental of Digital Image Processing,* Prentice Hall, 1989, Chapter 3

[Joh03] Garrett M. Johnson, Mark D. Fairchild, *A top down description of S-CIELAB and CIEDE2000,* Color Research & Application, Volume 28, Issue 6, Date: December 2003, Pages: 425-435

[Kay92] David C. Kay, John R. Levine, *Graphics file formats,* Blue Ridge Summit, Pa.: Windcrest/McGraw-Hill, c1992

[Kue98] R. G. Kuehni, *Hue uniformity and the CIELAB space and color difference formula*, Color Research & Application, Volume 23, Issue 5, Date: October 1998, Pages: 314-322

[Kue03] Rolf G. Kuehni, *CIEDE2000, milestone or final answer?* Color Research & Application, Volume 27, Issue 2, Date: April 2002, Pages: 126-127

[Lic00] C. J. Li, M. R. Luo, R. W. G. Hunt, *A revision of the CIECAM97s model***,** Color Research & Application, Volume 25, Issue 4, Date: August 2000, Pages: 260-266

[Luo01] M. R. Luo, G. Cui, and B. Rigg, *The development of the CIE 2000 colour-difference formula: CIEDE2000*, Color Research & Application, Volume 26, Issue 5, Date: October 2001, Pages: 340-350

[Luoro] Ronnier Luo and Peter Rhodes, *Colour Science Glossary,* http://colour.derby.ac.uk/colour/info/glossary/

[Mar98] J.S. Marks, J. Schmidt, M.T. Morgan, J.A. Nyenhuis, R.L. Stroshine, *Nuclear magnetic resonance for poultry meat fat analysis and bone chip detection*. Industry summary for US Poultry and Egg Association, 1998.

[Mas78] Robert Massof and Joseph Bird, *A general zone theory of color and brightness vision. I.*, Optical Society of America, Volume 68, Nov 1978

[Mel00] Manuel Melgosa, *Test CIELAB-Based Color-Difference Formulas*, Color Research and Application, Volume 25, Number 1, February 2000

[Mel99] M. Melgosa, M. M. Perez, A. El Moraghi, E. Hita, *Color Discrimination Results from a CRT Device: Influence of Luminance*, Color Research and Application, Volume 24, Number 1, February 1999

[Nav00] Carl R. Nave *HyperPhysics*, 2000 http://hyperphysics.phy-astr.gsu.edu/hbase/vision

[New55] Isaac Newton, *Optics, Great books of the Western World*, v.34, Chicago, Encyclopedia, Britannica, 1955

[Poynt] Charles Poynton, *Frequently Asked Questions About Color*, http://www.inforamp.net/~poynton/

[Qia98] Yue Qiao, Roy S. Berns, Lisa Reniff, Ethan Montag, *Visual Determination of Hue Suprathreshold Color-Difference Tolerances,* Color Research and Application, Volume 23, Number 5, October 1998

[Raj02] A. Rajeev Ramanath, B.W. Snyder, C.D. Hinks, *Image comparison measure for digital still colour cameras***,** Image Processing. Proceedings. 2002 International Conference on, Volume 1, 22-25 Page(s): I-629 - I-632 vol.1, Sept. 2002

[Rober] M. Roberts, *The CIE Colour System*, http://www.cs.bham.ac.uk/~mer/colour

[Rus99] John C. Russ, *The Image Processing Handbook*, Third Edition, CRC, 1999

[Sch95] T.F. Schatzki, R.P. Haff, R. Young, I. Can, L.C. Le and N. Toyofuku, *Defect detection in apples by means of X-ray imaging*. Trans. ASAE 40 (1997), pp. 1407–1415

[Sid95] Maher A. Sid-Ahmed, *Image Processing*, McGraw-Hill, 1995

[Son99] Milan Sonka, Vaclav Hlavac, Roger Boyle, *Image Processing, Analysis, and Machine Vision*, Second Edition, PWS publishing, 1999

[Sut03] S. Suthaharan, S. Kim, K. R. Rao, *A new quality metric based on just-noticeable difference perceptual regions, edge extraction and human vision,* Electrical and Computer Engineering, Vol. 30, Issue 2, pp. 81-88, 2003

[Tho00] Knud Thomsen, *A Euclidean Color Space in High Agreement with the CIE94 Color Difference Formula*, Color Research and Application, Volume 25, Number 1, February 2000.

[Tij01] L. Tijskens, E. Schijvens, and E. Biekman, *Modelling the change in colour of broccoli and green beans during blanching*, Innovative Food Science & Emerging Technologies, no.2, pp.303-313, 2001.

[Woo99] Mason Woo, Jackie Neider, Tom Davis, Dave Shreiner, *OpenGL Programming Guide*, Third Edition, Addison-Wesley, 1999

[Wes00] Stephen Westland, *Frequently asked questions about Colour Physics*, 2000, http://www.colourware.co.uk/cpfaq.htm

[Wys67] G. Wyszecki and W. Stiles, *Color Science*, John Wiley & Sons, 1967

[Xri01] X-rite, A *Guide to Understanding Color Communication*, 2002 http://www.xrite.com/documents/mktg/L10-001.pdf

[Xri02] X-Rite, *A guide to Understanding Color Tolerancing*

http://www.xrite.com/documents/mktg/L10-024.pdf

[Yin99] L. Ying, J. Cai, and D. Cen, *Appraising method and standard for storage durability in broccoli*, Acta Agriculturae Boreali-Sinica. Vol.14, no.4, pp134–136, 1999

Appendix A

Source codes of MATLAB

Function rgb2lab.m

```
function [CIE_L,CIE_a,CIE_b] = rgb2lab(InputRed,InputGreen,InputBlue)

%-------------------------------------------------------------------
%RGB2LAB Convert red-green-blue colors to CIE L-a-b.
%   L = RGB2LAB(M) converts an RGB color map to a CIEL*a*b* color map.
%   Each map is a matrix with any number of rows, exactly three columns.
%   and elements L from 0 to 100, A from -100 to 100 and B from -100
%   to 100. The columns of the input matrix, M,represent intensity of
%   red, blue and green, respectively.  The columns of the resulting
%   output matrix, L, represent CIEL, CIEa and CIEb respectively.
%
%   LAB = RGB2LAB(RGB) converts the RGB image RGB (3-D array) to the
%   equivalent LAB image CIELab (3-D array).
%
% The input RGB values are located in the range of [0,255]
%
%   CLASS SUPPORT
%   -------------
%   If the input is an RGB image, it can be of class uint8, uint16, or
%   double; the output image is of class double.  If the input is a
%   colormap, the input and output colormaps are both of class double.
%
%   See also LAB2RGB, RGB2HSV, HSV2RGB, COLORMAP, RGBPLOT

%   Undocumented syntaxes:
%   [L,A,B] = RGB2LAB(R,G,B) converts the RGB image R,G,B to the
%   equivalent LAB image L,A,B.
%
%   LAB = RGB2LAB(R,G,B) converts the RGB image R,G,B to the
%   equivalent CIEL*a*b* image stored in the 3-D array (LAB).
%
```

```
%   [L,A,B] = RGB2LAB(RGB) converts the RGB image RGB (3-D array) to
%   the equivalent CIEL*a*b* image L,A,B.

%   Formulae are referenced in the chapter 4
%------------------------------------------------------------------------


% Check the numbers of input parameters and transform the RGB in [0,1]

switch nargin
case 1,
InputRed = double(InputRed) / 255;

case 3,
InputRed = double(InputRed) / 255;
InputGreen = double(InputGreen) / 255;
InputBlue = double(InputBlue) / 255;


otherwise,
     error('Wrong number of input arguments.');
end


% Determine whether input includes a 3-D array and check the sizes of the input values

threeD = (ndims(InputRed)==3);

if threeD,
  InputGreen = InputRed(:,:,2); InputBlue = InputRed(:,:,3); InputRed = InputRed(:,:,1);
  siz = size(InputRed);
  InputRed = InputRed(:); InputGreen = InputGreen(:); InputBlue = InputBlue(:);
elseif nargin==1,
  InputGreen = InputRed(:,2); InputBlue = InputRed(:,3); InputRed = InputRed(:,1);
  siz = size(InputRed);
else
  if ~isequal(size(InputRed),size(InputGreen),size(InputBlue)),
    error('R,G,B must all be the same size.');
  end
  siz = size(InputRed);
  InputRed = InputRed(:); InputGreen = InputGreen(:); InputBlue = InputBlue(:);
end


%   Nonlinear sR'G'B' values are transformed to linear R,G,B values

temp_samll = InputRed .* (InputRed <= 0.04045) / 12.92;
temp_big = (((InputRed + 0.055) / 1.055) .^ 2.4) .* (InputRed > 0.04045);
% InputRed is the linear Red value in the interval 0 to 1
InputRed = temp_samll + temp_big;
```

```
temp_samll = InputGreen .* (InputGreen <= 0.04045) / 12.92;
temp_big = (((InputGreen + 0.055) / 1.055) .^ 2.4) .* (InputGreen > 0.04045);
% InputGreen is the linear Green value in the interval 0 to 1
InputGreen = temp_samll + temp_big;

temp_samll = InputBlue .* (InputBlue <= 0.04045) / 12.92;
temp_big = (((InputBlue + 0.055) / 1.055) .^ 2.4) .* (InputBlue > 0.04045);
% Inputblue is the linear Blue value in the interval 0 to 1
InputBlue = temp_samll + temp_big;


%   Linear R, G, B values are transformed to CIE XYZ values

% CIE_X, CIE_Y, and CIE_Z are the CIEXYZ tristimulus values
CIE_X = 0.4124 * InputRed + 0.3576 * InputGreen + 0.1805 * InputBlue;
CIE_Y = 0.2126 * InputRed + 0.7152 * InputGreen + 0.0722 * InputBlue;
CIE_Z = 0.0193 * InputRed + 0.1192 * InputGreen + 0.9505 * InputBlue;


%   CIE XYZ values are transformed to CIE L,a,b values

% xn, yn and zn are the CIEXYZ tristimulus values of the reference white
xn = 0.4124 + 0.3576 + 0.1805;
yn = 0.2126 + 0.7152 + 0.0722;
zn = 0.0193 + 0.1192 + 0.9505;

CIE_X = CIE_X./xn;
CIE_Y = CIE_Y./yn;
CIE_Z = CIE_Z./zn;
temp_big = (CIE_Y .^ (1/3) * 116 - 16) .* (CIE_Y > 0.008856);
temp_small = CIE_Y .* (CIE_Y <= 0.008856) * 903;
CIE_L = temp_big + temp_small;

temp_big = CIE_X .^ (1/3) .* (CIE_X > 0.008858);
temp_small = (7.787 * CIE_X + 16 / 116) .* (CIE_X <= 0.008858);
CIE_Xtemp = temp_big + temp_small;

temp_big = CIE_Y .^ (1/3) .* (CIE_Y > 0.008858);
temp_small = (7.787 * CIE_Y + 16 / 116) .* (CIE_Y <= 0.008858);
CIE_Ytemp = temp_big + temp_small;

temp_big = CIE_Z .^ (1/3) .* (CIE_Z > 0.008858);
temp_small = (7.787 * CIE_Z + 16/116) .* (CIE_Z <= 0.008858);
CIE_Ztemp = temp_big + temp_small;

CIE_a = 500 * (CIE_Xtemp - CIE_Ytemp);
CIE_b = 200 * (CIE_Ytemp - CIE_Ztemp);

% Display the result
if nargout<=1
   if (threeD | nargin ==3),
```

```
      CIE_L = reshape(CIE_L,siz);
      CIE_a = reshape(CIE_a,siz);
      CIE_b = reshape(CIE_b,siz);
      CIE_L = cat(3,CIE_L,CIE_a,CIE_b);
   else
      CIE_L = [CIE_L CIE_a CIE_b];
   end
else
   CIE_L = reshape(CIE_L,siz);
   CIE_a = reshape(CIE_a,siz);
   CIE_b = reshape(CIE_b,siz);
end
```

Function lab2rgb.m

```
function [OutputRed,OutputGreen,OutputBlue] = lab2rgb(CIE_L,CIE_a,CIE_b)

%--------------------------------------------------------------------------------
%LAB2RGB Convert CIE L-a-b colors to red-green-blue
%   M = LAB2RGB(L) converts a CIEL*a*b* color map to an RGB color map.
%   Each map is a matrix with any number of rows, exactly three columns.
%   and elements L from 0 to 100, A from -100 to 100 and B from -100
%   to 100. The columns of the input matrix, L, represent intensity of
%   CIE L, A, B, respectively.  The columns of the resulting output
%   matrix, M, represent intensity of red, blue and green, respectively.
%
%   RGB = LAB2RGB(LAB) converts the LAB image LAB (3-D array) to the
%   equivalent RGB image RGB (3-D array).
%
%   See also LAB2RGB, RGB2HSV, HSV2RGB, COLORMAP, RGBPLOT

%   Undocumented syntaxes:
%   [R,G,B] = LAB2RGB(L,A,B) converts the LAB image L,A,B to the
%   equivalent RGB image R,G,B.
%
%   RGB = LAB2RGB(L,A,B) converts the LAB image L,A,B to the
%   equivalent RGB image stored in the 3-D array (RGB).
%
%   [R,G,B] = LAB2RGB(RGB) converts the LAB image LAB (3-D array) to
%   the equivalent RGB image R,G,B.

%   Formulae are referenced in chapter 4
%----------------------------------------------------------------------

% Check the numbers of input parameters
```

```
threeD = (ndims(CIE_L)==3);

if threeD,
  CIE_a = CIE_L(:,:,2); CIE_b = CIE_L(:,:,3); CIE_L = CIE_L(:,:,CIE_L);
  siz = size(CIE_L);
  CIE_L = CIE_L(:); CIE_a = CIE_a(:); CIE_b = CIE_b(:);
elseif nargin==1,
  CIE_a = CIE_L(:,2); CIE_b = CIE_L(:,3); CIE_L = CIE_L(:,1);
  siz = size(CIE_L);
else
  if ~isequal(size(CIE_L),size(CIE_a),size(CIE_b)),
    error('L,A,B must all be the same size.');
  end
  siz = size(CIE_L);
  CIE_L = CIE_L(:); CIE_a = CIE_a(:); CIE_b = CIE_b(:);
end


%  CIE L,a,b values are transformed to CIE X,Y,Z values

% xn, yn and zn are the CIEXYZ tristimulus values of the reference white
xn = 0.412453 + 0.35758 + 0.180423;
yn = 0.212671 + 0.71516 + 0.072169;
zn = 0.019334 + 0.11919 + 0.950227;

%  CIE_X, CIE_Y, and CIE_Z are the CIEXYZ tristimulus values
CIE_X = xn *((CIE_L+16)/116 + CIE_a/500).^3;
CIE_Y = yn *((CIE_L+16)/116).^3;
CIE_Z = zn *((CIE_L+16)/116 - CIE_b/200).^3;

%  CIE X,Y,Z are transformed to linear R,G,B values
Red = 3.240479*CIE_X - 1.53715*CIE_Y - 0.498545*CIE_Z;
Green = -0.969256*CIE_X + 1.875992*CIE_Y + 0.041556*CIE_Z;
Blue = 0.055648*CIE_X - 0.204043*CIE_Y + 1.057311*CIE_Z;


%  Linear R,G,B values are transformed to nonlinear sR'G'B'
temp_big = (1.055*Red.^(1/2.4) - 0.055) .* (Red>0.0031308);
temp_small = (12.92*Red) .* (Red<=0.0031308);
Red = temp_big + temp_small;

temp_big = (1.055*Green.^(1/2.4)-0.055) .* (Green>0.0031308);
temp_small = (12.92*Green) .* (Green<=0.0031308);
Green = temp_big + temp_small;

temp_big = (1.055*Blue.^(1/2.4) - 0.055) .* (Blue>0.0031308);
temp_small = (12.92*Blue) .* (Blue<=0.0031308);
Blue = temp_big + temp_small;

%  The output RGB values are truncated in the range [0,255]
OutputRed = double(uint8(Red*256));
```

```
OutputGreen = double(uint8(Green*256));
OutputBlue = double(uint8(Blue*256));

%  Display the result
if nargout<=1
   if (threeD | nargin ==3),
      OutputRed = reshape(OutputRed,siz);
      OutputGreen = reshape(OutputGreen,siz);
      OutputBlue = reshape(OutputBlue,siz);
      OutputRed = cat(3,OutputRed,OutputGreen,OutputBlue);
   else
      OutputRed = [OutputRed OutputGreen OutputBlue];
   end
else
   OutputRed = reshape(OutputRed,siz);
   OutputGreen = reshape(OutputGreen,siz);
   OutputBlue = reshape(OutputBlue,siz);
end
```

Function lab2rgb_inrgbgamut.m

```
function [OutputRed,OutputGreen,OutputBlue] =
lab2rgb_inrgbgamut(CIE_L,CIE_a,CIE_b)

%-------------------------------------------------------------------------------
%LAB2RGB_INRGBGAMUT is almost same as the function LAB2RGB except that any
colours
%   are out of the RGB gamut are set to zero
%   M = LAB2RGB(L) converts a CIEL*a*b* color map to an RGB color map.
%   Each map is a matrix with any number of rows, exactly three columns.
%   and elements L from 0 to 100, A from -100 to 100 and B from -100
%   to 100. The columns of the input matrix, L, represent intensity of
%   CIE L, A, B, respectively.  The columns of the resulting output
%   matrix, M, represent intensity of red, blue and green, respectively.
%
%   RGB = LAB2RGB(LAB) converts the LAB image LAB (3-D array) to the
%   equivalent RGB image RGB (3-D array).
%
%   See also RGB2LAB, LAB2RGB, RGB2HSV, HSV2RGB, COLORMAP, RGBPLOT

%   Undocumented syntaxes:
%   [R,G,B] = LAB2RGB(L,A,B) converts the LAB image L,A,B to the
%   equivalent RGB image R,G,B.
%
%   RGB = LAB2RGB(L,A,B) converts the LAB image L,A,B to the
%   equivalent RGB image stored in the 3-D array (RGB).
```

```
%
%   [R,G,B] = LAB2RGB(RGB) converts the LAB image LAB (3-D array) to
%   the equivalent RGB image R,G,B.

%   Formulae are referenced in chapter 4
%-------------------------------------------------------------------

% Check the numbers of input parameters

threeD = (ndims(CIE_L)==3);

if threeD,
  CIE_a = CIE_L(:,:,2); CIE_b = CIE_L(:,:,3); CIE_L = CIE_L(:,:,CIE_L);
  siz = size(CIE_L);
  CIE_L = CIE_L(:); CIE_a = CIE_a(:); CIE_b = CIE_b(:);
elseif nargin==1,
  CIE_a = CIE_L(:,2); CIE_b = CIE_L(:,3); CIE_L = CIE_L(:,1);
  siz = size(CIE_L);
else
  if ~isequal(size(CIE_L),size(CIE_a),size(CIE_b)),
    error('L,A,B must all be the same size.');
  end
  siz = size(CIE_L);
  CIE_L = CIE_L(:); CIE_a = CIE_a(:); CIE_b = CIE_b(:);
end


%  CIE L,a,b values are transformed to CIE X,Y,Z values

% xn, yn and zn are the CIEXYZ tristimulus values of the reference white
xn = 0.412453 + 0.35758 + 0.180423;
yn = 0.212671 + 0.71516 + 0.072169;
zn = 0.019334 + 0.11919 + 0.950227;

%  CIE_X, CIE_Y, and CIE_Z are the CIEXYZ tristimulus values
CIE_X = xn *((CIE_L+16)/116 + CIE_a/500).^3;
CIE_Y = yn *((CIE_L+16)/116).^3;
CIE_Z = zn *((CIE_L+16)/116 - CIE_b/200).^3;

%  CIE X,Y,Z are transformed to linear R,G,B values
Red = 3.240479*CIE_X - 1.53715*CIE_Y - 0.498545*CIE_Z;
Green = -0.969256*CIE_X + 1.875992*CIE_Y + 0.041556*CIE_Z;
Blue = 0.055648*CIE_X - 0.204043*CIE_Y + 1.057311*CIE_Z;


%  Linear R,G,B values are transformed to nonlinear sR'G'B'
temp_big = (1.055*Red.^(1/2.4) - 0.055) .* (Red>0.0031308);
temp_small = (12.92*Red) .* (Red<=0.0031308);
Red = temp_big + temp_small;

temp_big = (1.055*Green.^(1/2.4)-0.055) .* (Green>0.0031308);
```

```
temp_small = (12.92*Green) .* (Green<=0.0031308);
Green = temp_big + temp_small;

temp_big = (1.055*Blue.^(1/2.4) - 0.055) .* (Blue>0.0031308);
temp_small = (12.92*Blue) .* (Blue<=0.0031308);
Blue = temp_big + temp_small;

%  Values are out of [0,1] are set to zero
rr = (Red>=0).*(Red<=1);
gg = (Green>=0).*(Green<=1);
bb = (Blue>=0).*(Blue<=1);

factor = rr.*gg.*bb;

%  The output RGB values are transformed in the range [0,255]
OutputRed = double(uint8(Red.*factor*256));
OutputGreen = double(uint8(Green.*factor*256));
OutputBlue = double(uint8(Blue.*factor*256));

%  Display the result
if nargout<=1
   if (threeD | nargin ==3),
      OutputRed = reshape(OutputRed,siz);
      OutputGreen = reshape(OutputGreen,siz);
      OutputBlue = reshape(OutputBlue,siz);
      OutputRed = cat(3,OutputRed,OutputGreen,OutputBlue);
   else
      OutputRed = [OutputRed OutputGreen OutputBlue];
   end
else
   OutputRed = reshape(OutputRed,siz);
   OutputGreen = reshape(OutputGreen,siz);
   OutputBlue = reshape(OutputBlue,siz);
end
```

Function rgbinlab_brightness.m

```
function rgbinlab_brightness(arg1,arg2,arg3)

%--------------------------------------------------------------------
%RGBINLAB_BRIGHTNESS creates a group images with different brightness
%  in CIEL*a*b* colour mode,but only the colours are in the RGB colour
%  model are displayed
%
%  There are three input parameters. Arg1 and arg2 are the range of
%  brightness that the images are created and the arg3 is the step of
```

```
% the brightness varying.
%
% If there is only one parameter, the output only has one image that the
% brightness is this input value.

% Check the input parameters
if nargin==1,
  l1=arg1; l2=arg1; step=1;
elseif nargin==2,
  l1=arg1; l2=arg2; step=1;
elseif nargin==3
  l1=arg1; l2=arg2; step=arg3;
end

% Create two matrices to express all colour with different CIE a* and CIE b* values
[CIE_a,CIE_b] = meshgrid(-100:100,-100:100);

% Create images with different brightness and display them
for temp = l1:step:l2
    CIE_L = ones(201,201)*temp;
    [R,G,B]=lab2rgb_inrgbgamut(CIE_L,CIE_a,CIE_b);
    figure,imshow(uint8(flipud(R)),uint8(flipud(G)),uint8(flipud(B)))
    title(['L* = ', num2str(temp)]);
end
```

Function rgbinlch_hue.m

```
function rgbinlch_hue(arg1,arg2,arg3,arg4)

%----------------------------------------------------------------------
%RGBINLCH_BRIGHTNESS creates a group images with different hue in
% CIEL*C*H* colour mode,but only the colours are in the RGB colour
% model are displayed
%
% There are four input parameters. Arg1 is the size of the output image.
% The value is same as the number of pixels in a squre image. Arg2 and arg3
% are the range of hue that the images are created and the arg4 is the
% step of the hue varying.
%
% If there is only one parameter, the output only has one image that the
% hue is this input value.

% Check the input parameters
if nargin==2,
  siz=arg1; h1=arg2; h2=arg2; step=1;
elseif nargin==3,
  siz=arg1; h1=arg2; h2=arg3; step=1;
```

```
elseif nargin==4
  siz=arg1; h1=arg2; h2=arg3; step=arg4;
end

% Create two matrices to express all colours with different C* and L* values
[CIE_C,CIE_L] = meshgrid(0:100/siz:100,0:100/siz:100);

% Create images with different hue and display them
for temp = h1:step:h2
    CIE_a = cos(temp/180*pi).*CIE_C;
    CIE_b = sin(temp/180*pi).*CIE_C;
    [R,G,B] = lab2rgb_inrgbgamut(CIE_L,CIE_a,CIE_b);
    figure,imshow(uint8(flipud(R)),uint8(flipud(G)),uint8(flipud(B)))
    title(['H* = ', num2str(temp)]);
end
```

Function diffc_inlh.m

```
function diffc_inlh(L,H)

%----------------------------------------------------------------------
%DIFFCINLH create a picture whose colours have same brightness and hue but
%  different saturation in CIEL*C*H* colour model.The saturation of colours
%  are increasing in the horizontal direction from left to right. Colours
%  are same in the vertical direction. Colours which the saturation is less
%  than 10 or more than 80 are not displayed in this image.
%
%  The input parameter L is the value of brightness within [0,100]. Parameter
%  H is the value of hue which is from 0 to 360 degree.


% Create a matrix to represent the varying saturation from 0 to 100
C=meshgrid(0:.5:100);

% Do not display the colour which the saturation is less than 10 or more than
% 80. Convert the CIEL*C*H* value to CIEL*a*b*
CIE_C = (C>10).*(C<80).*C;
CIE_a=cos(H/180*pi).*CIE_C;
CIE_b=sin(H/180*pi).*CIE_C;
CIE_L=ones(201,201).*L.*(C>10).*(C<80);

% Convert CIEL*a*b* value to RGB and display it
[R,G,B]=lab2rgb_inrgbgamut(CIE_L,CIE_a,CIE_b);
figure,imshow(flipud(uint8(R)),flipud(uint8(G)),flipud(uint8(B)))
title([ 'L* = ', num2str(L),', H* = ',num2str(H),' degree']);
```

Function diffl_inch.m

```
function diffl_inch(C,H)

%--------------------------------------------------------------------
%DIFFCINLH create a picture whose colours have same saturation and hue but
%  different brightness in CIEL*C*H* colour model.The brightness of colours
%  are increasing in the horizontal direction from left to right. Colours
%  are same in the vertical direction. Colours which the brihtness is less
%  than 10 are not displayed in this image.
%
%  The input parameter C is the value of brightness within [0,100]. Parameter
%  H is the value of hue which is from 0 to 360 degree.


% Create a matrix to represent the varying brightness from 0 to 100
L=meshgrid(0:.5:100);

% Do not display the colour which the saturation is less than 10
% Convert the CIEL*C*H* value to CIEL*a*b*
CIE_L = (L>10).*L;
CIE_C = ones(201,201).*C.*(L>10);
CIE_a = cos(H/180*pi).*CIE_L;
CIE_b = sin(H/180*pi).*CIE_L;

% Convert CIEL*a*b* value to RGB and display it
[R,G,B]=lab2rgb_inrgbgamut(CIE_L,CIE_a,CIE_b);
figure,imshow(flipud(uint8(R)),flipud(uint8(G)),flipud(uint8(B)))
title([ 'C* = ', num2str(C),', H* = ',num2str(H),' degree']);
```

Function sc_lch.m

```
function sc_lch(L,C,H,N)

%----------------------------------------------------------------------
%SC_LCH creates an N by N image to display the input colour. The input parameters
% are the threes values of CIEL*C*H* colour model, which the value of L* is
% in the range of [0,100], C* is in the range of [0,100], and H* has the
% degree of unit in the range of [0,360]

% Convert to the values in CIEL*a*b* and RGB colour model
CIE_a = cos(H/180*pi)*C;
CIE_b = sin(H/180*pi)*C;

[R,G,B] = lab2rgb_inrgbgamut(L,CIE_a,CIE_b);

% Create and display an N by N image with the same colour appearance
```

```
R = ones(N,N)*R;
G = ones(N,N)*G;
B = ones(N,N)*B;

figure,imshow(uint8(R),uint8(G),uint8(B))
title(['L* = ',num2str(L),', C* = ',num2str(C),', H* = ',num2str(H)])
```

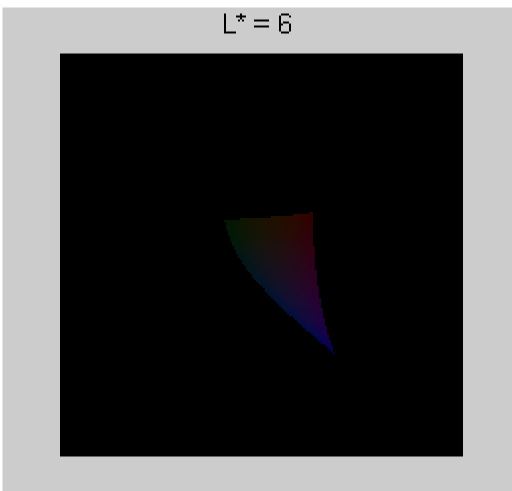
Function dc_lch.m

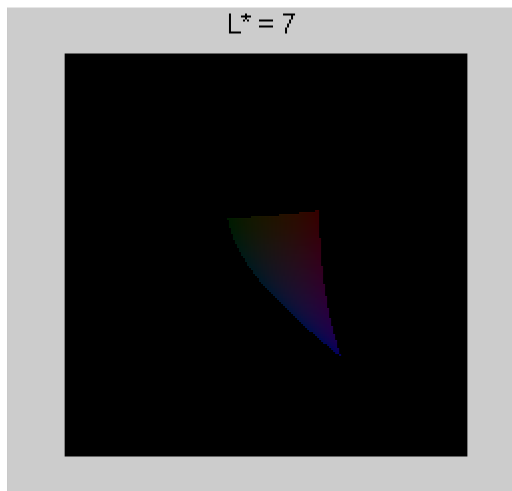
```
function dc_lch(L1,C1,H1,L2,C2,H2)

%------------------------------------------------------------------------
%DC_LCH creates two 150 by 150 images to display two input colours. The
% input parameters are two CIEL*C*H* values, which the value of L* is
% in the range of [0,100], C* is in the range of [0,100], and H* has the
% degree of unit in the range of [0,360].

% Convert to the values in CIEL*a*b* and RGB colour model
CIE_a1 = cos(H1/180*pi)*C1;
CIE_b1 = sin(H1/180*pi)*C1;

CIE_a2 = cos(H2/180*pi)*C2;
CIE_b2 = sin(H2/180*pi)*C2;

[R1,G1,B1] = lab2rgb_inrgbgamut(L1,CIE_a1,CIE_b1);
[R2,G2,B2] = lab2rgb_inrgbgamut(L2,CIE_a2,CIE_b2);

% Create and display two 150 by 150 images with the their colour appearances
R1 = ones(150,150)*R1;
G1 = ones(150,150)*G1;
B1 = ones(150,150)*B1;

R2 = ones(150,150)*R2;
G2 = ones(150,150)*G2;
B2 = ones(150,150)*B2;


figure,subplot(1,2,1),imshow(uint8(R1),uint8(G1),uint8(B1))
title(['L* = ',num2str(L1),', C* = ',num2str(C1),', H* = ',num2str(H1)])
subplot(1,2,2),imshow(uint8(R2),uint8(G2),uint8(B2))
title(['L* = ',num2str(L2),', C* = ',num2str(C2),', H* = ',num2str(H2)])
```

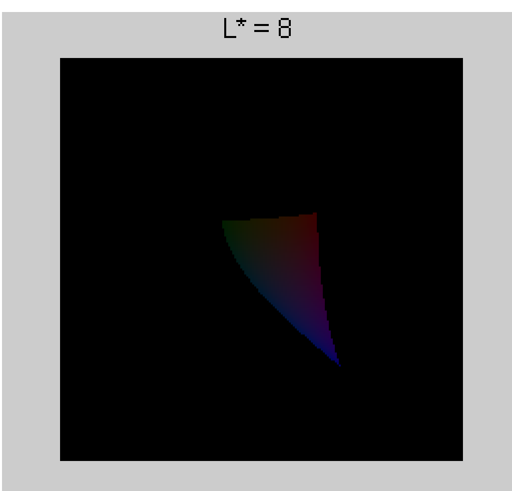
Appendix B

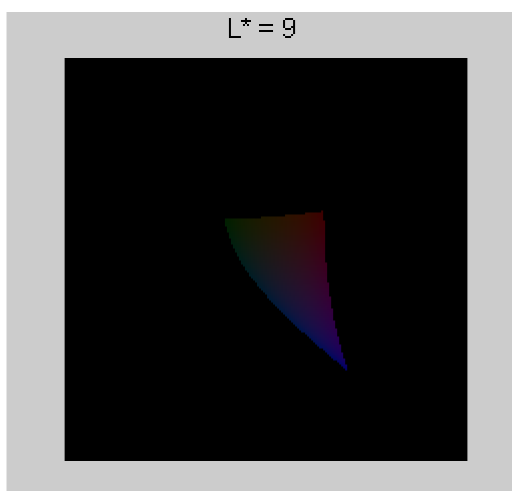
Sample colours

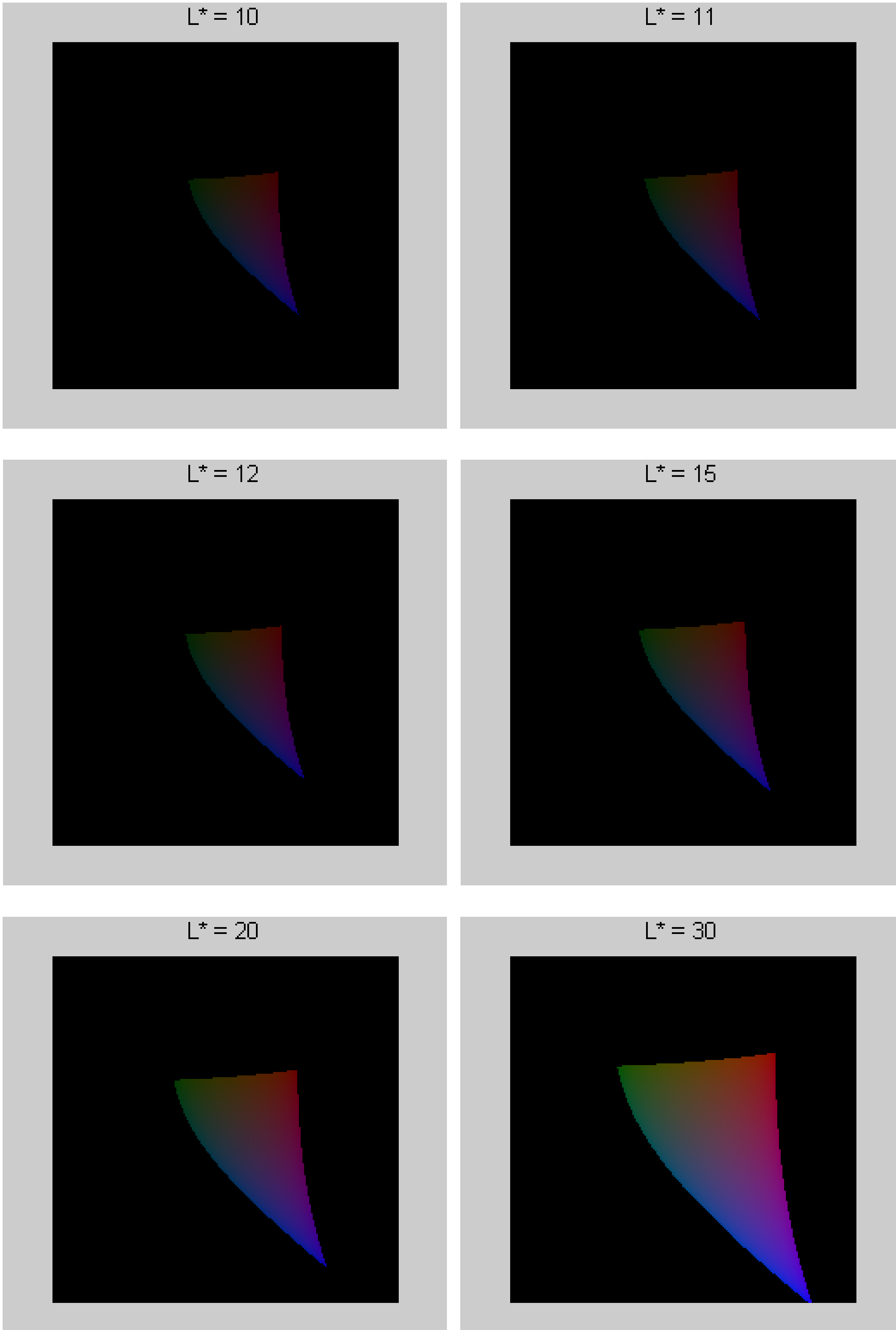
Colour of RGB in CIEL*a*b* colour model with different values of brightness

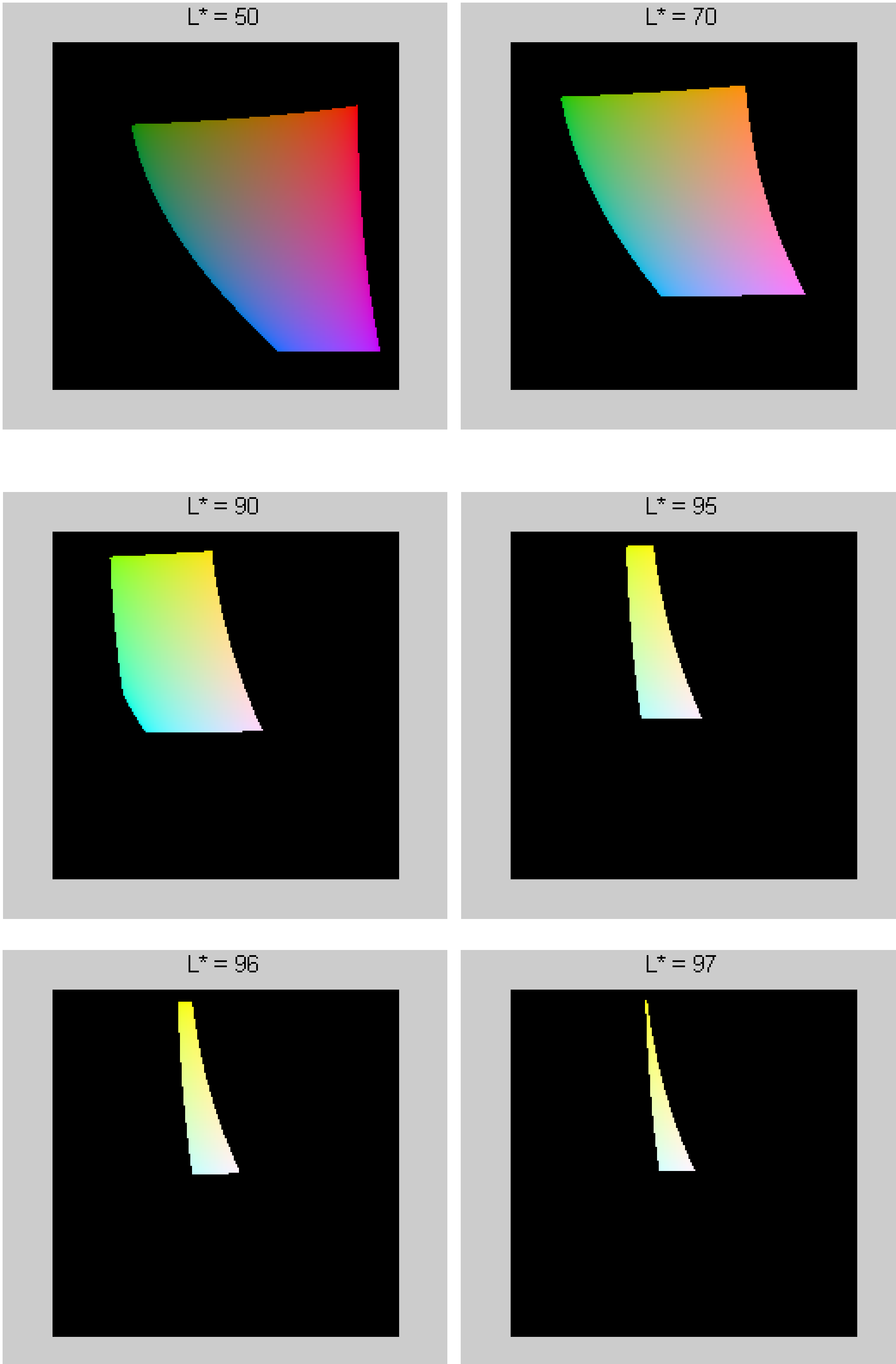


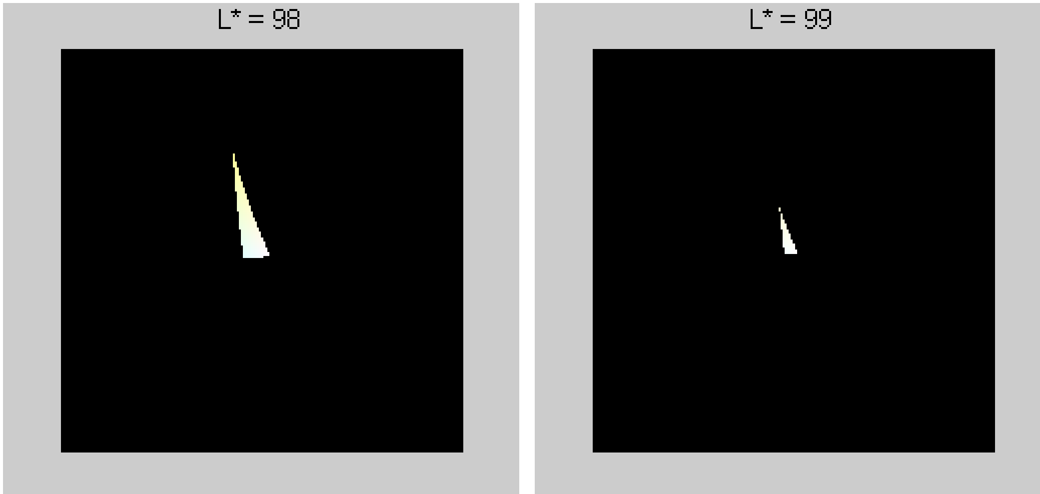




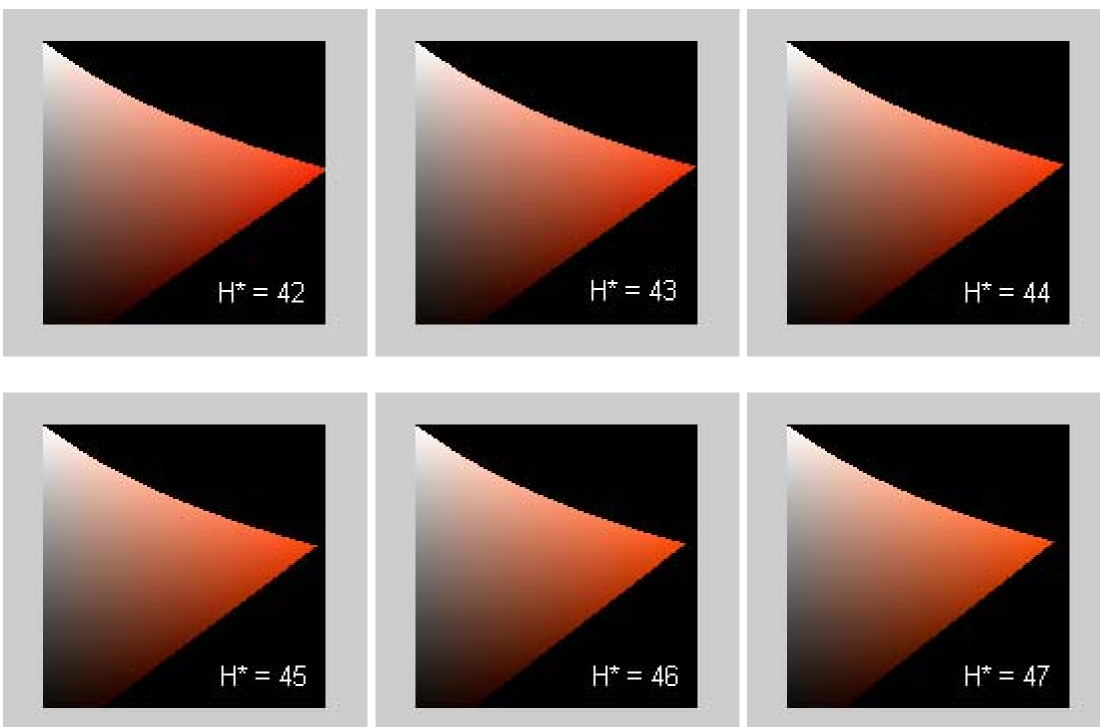


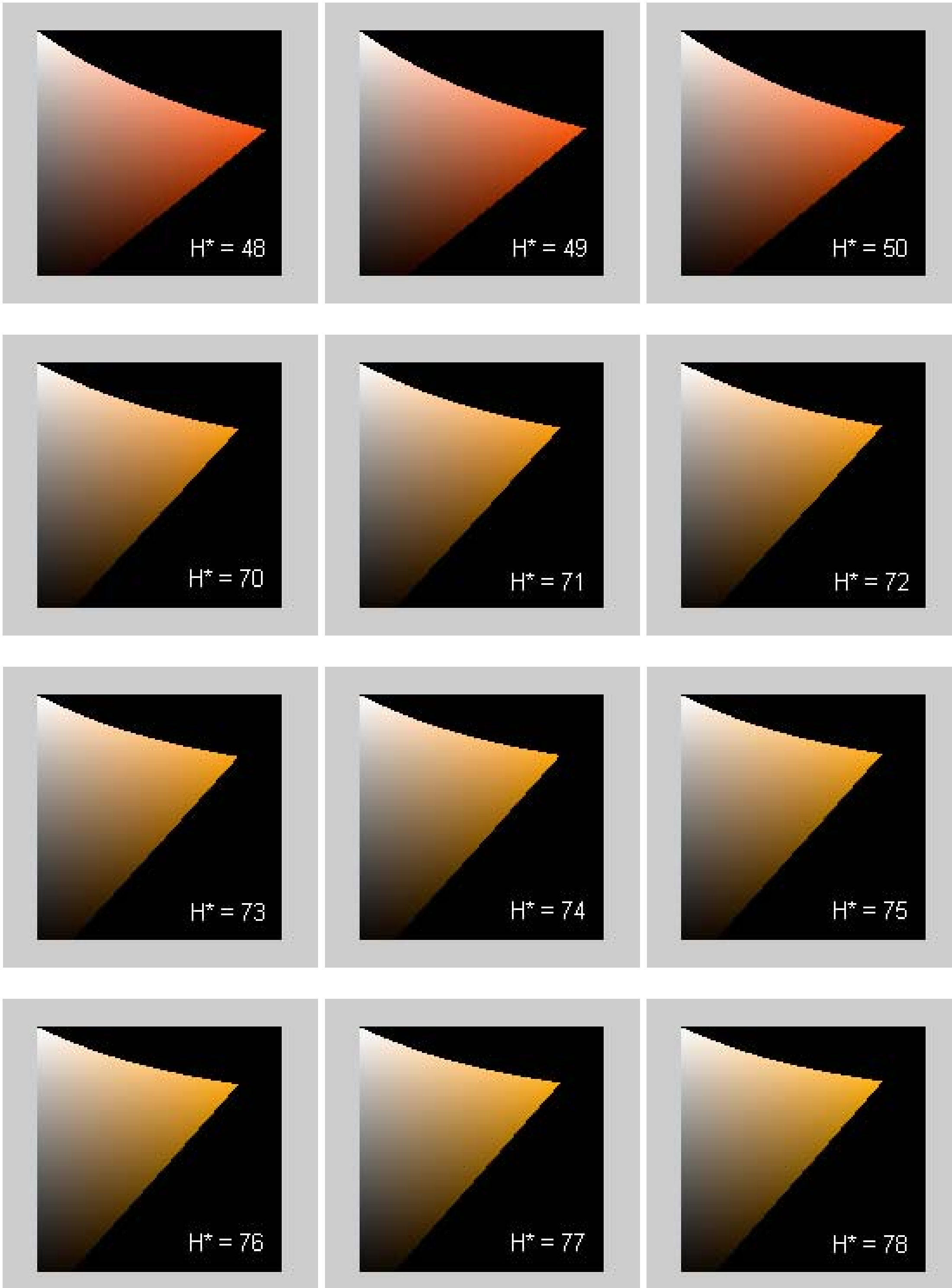

L* = 10
L* = 11
L* = 12
L* = 15
L* = 20
L* = 30

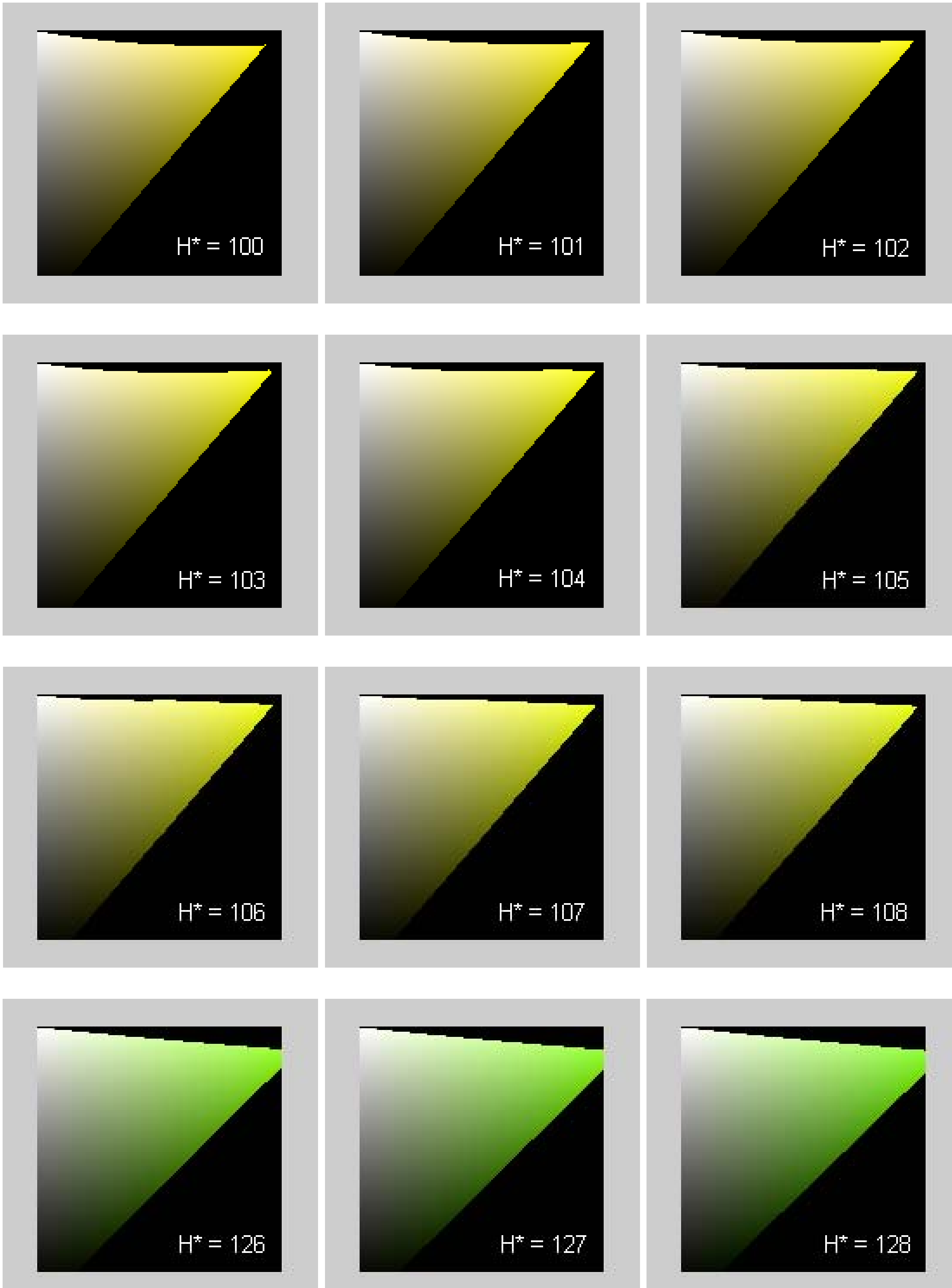

L* = 50
L* = 70
L* = 90
L* = 95
L* = 96
L* = 97

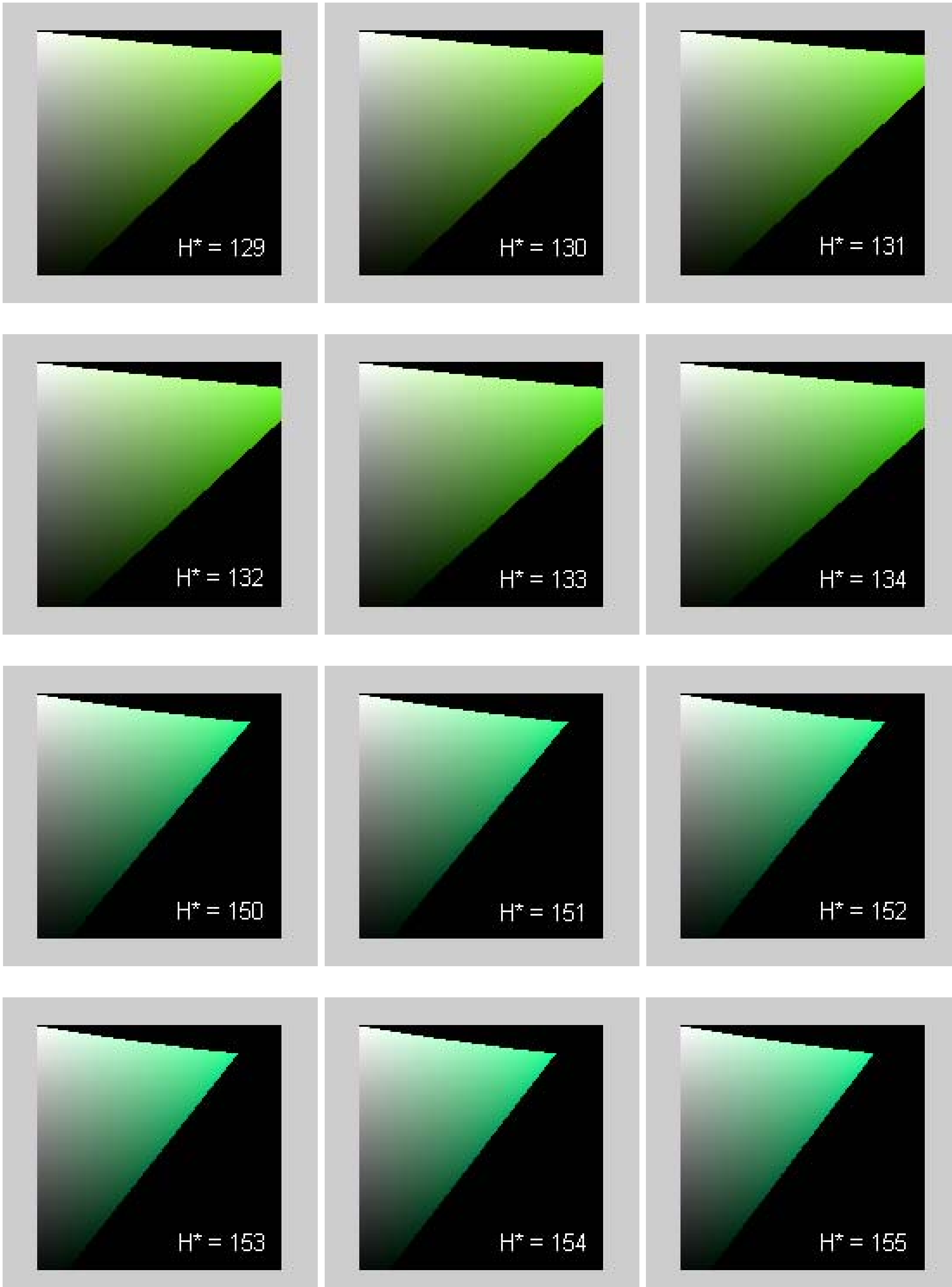


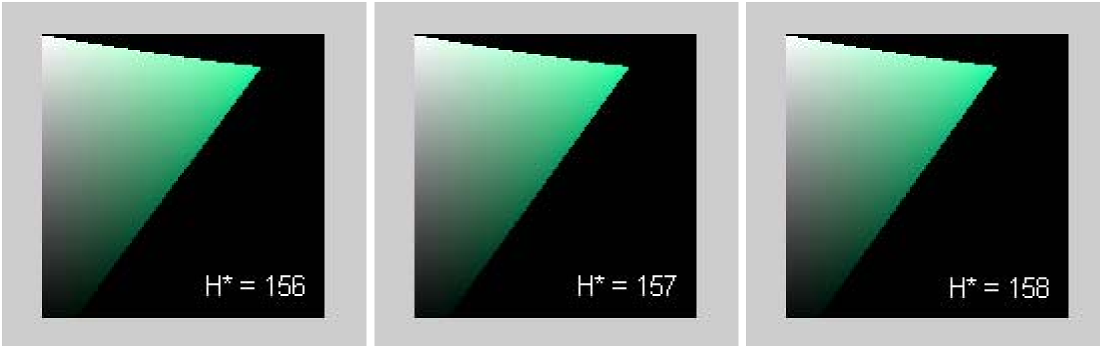
Colour of RGB in CIEL*C*H* colour model with different values of hue




H* = 48
H* = 49
H* = 50
H* = 70
H* = 71
H* = 72
H* = 73
H* = 74
H* = 75
H* = 76
H* = 77
H* = 78

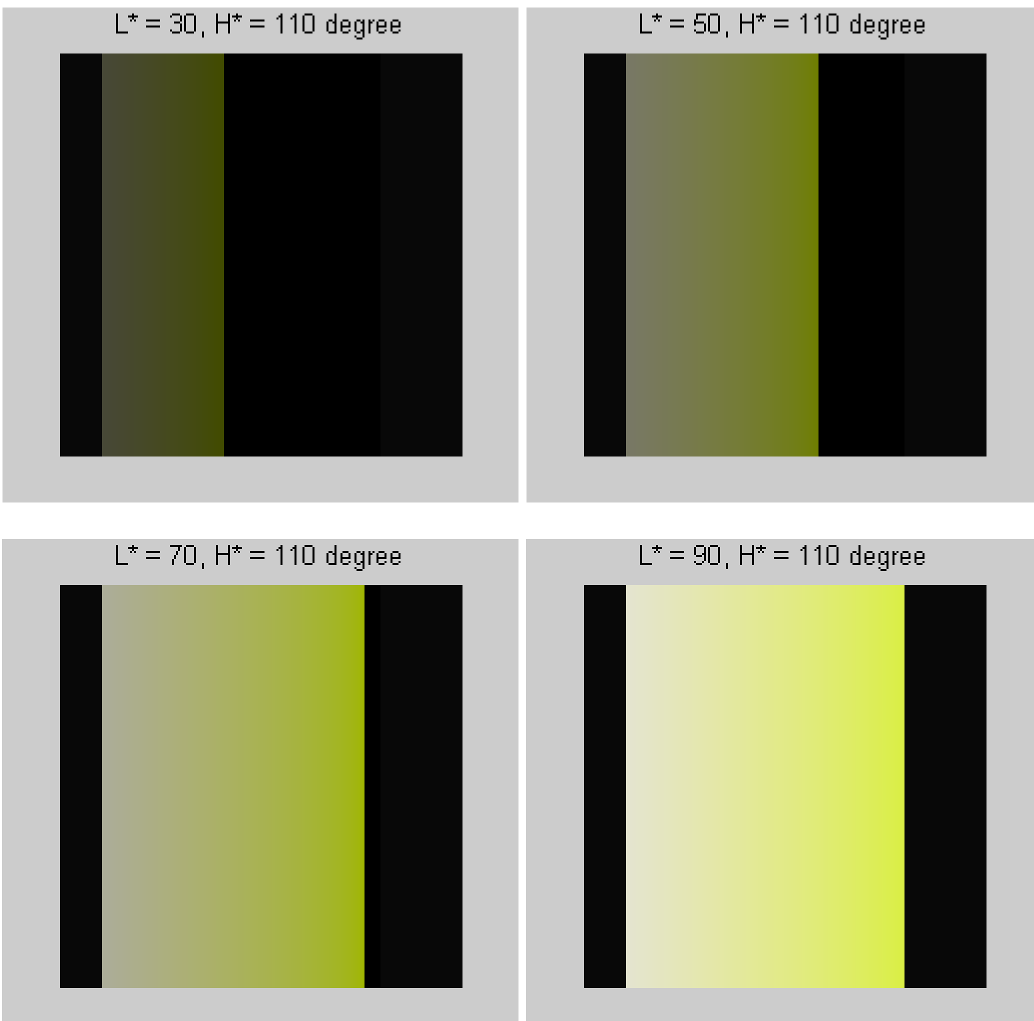

H* = 100
H* = 101
H* = 102
H* = 103
H* = 104
H* = 105
H* = 106
H* = 107
H* = 108
H* = 126
H* = 127
H* = 128

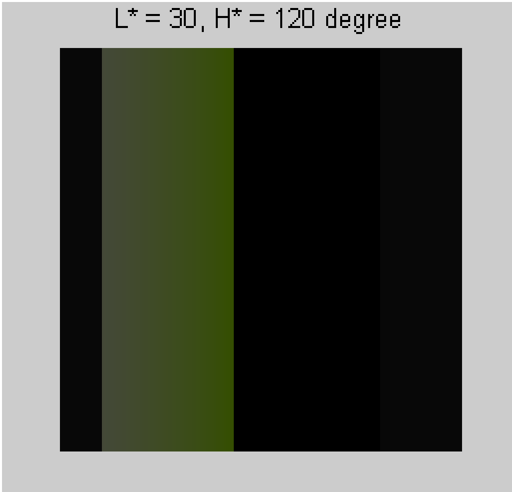

H* = 129
H* = 130
H* = 131
H* = 132
H* = 133
H* = 134
H* = 150
H* = 151
H* = 152
H* = 153
H* = 154
H* = 155

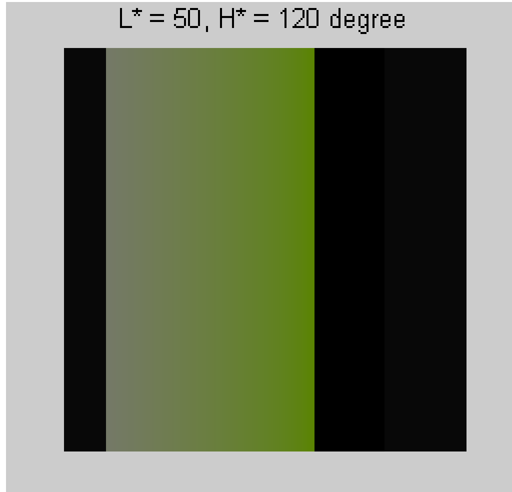


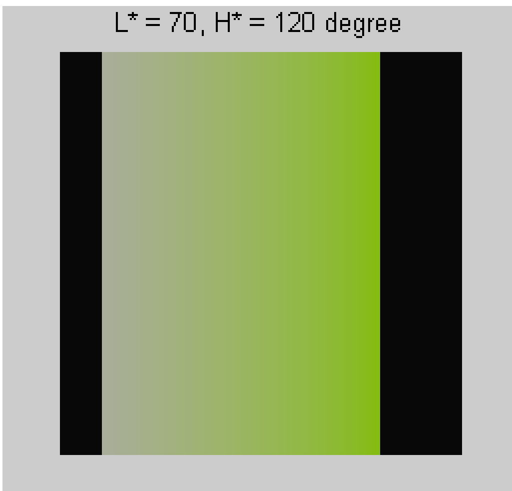
Colours with same hue and brightness but varying saturation

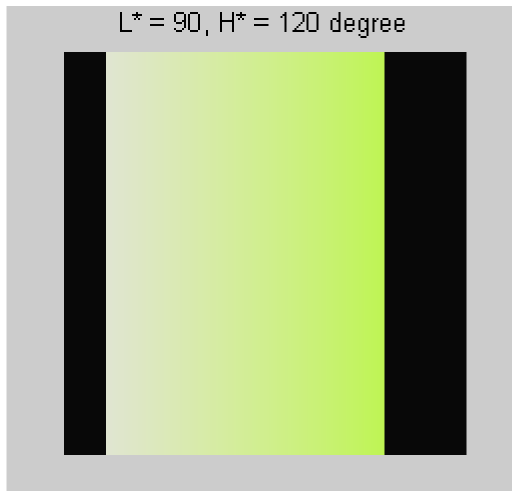
(For survey 1 in the Appendix C)

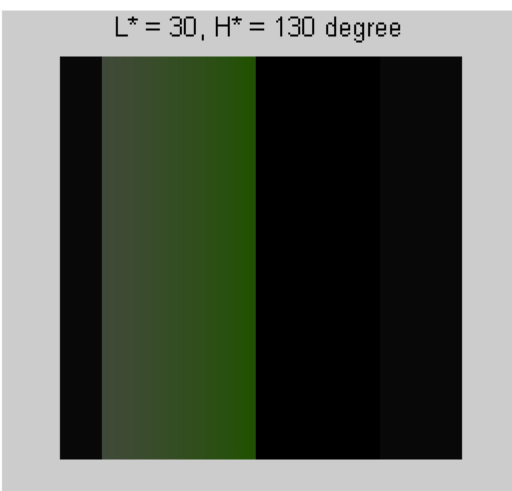


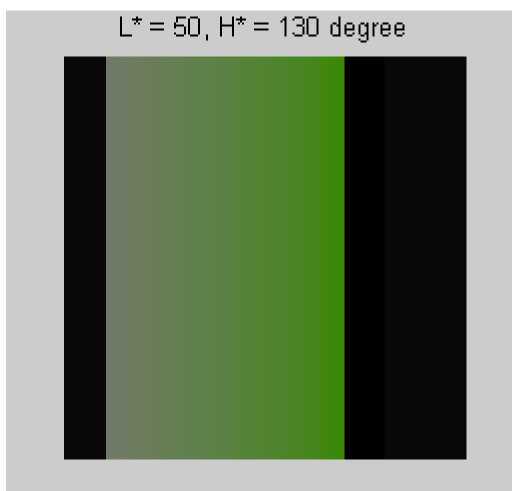

L* = 30, H* = 120 degree

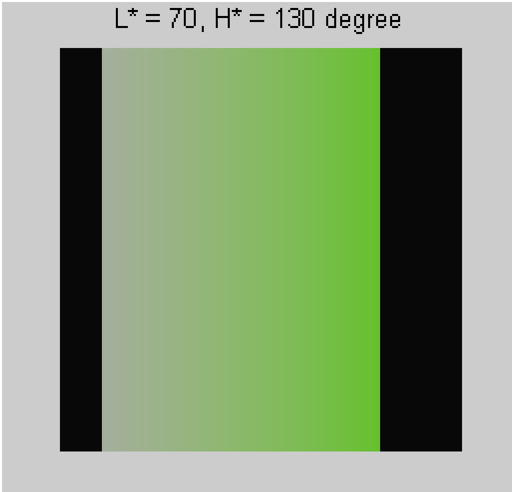

L* = 50, H* = 120 degree

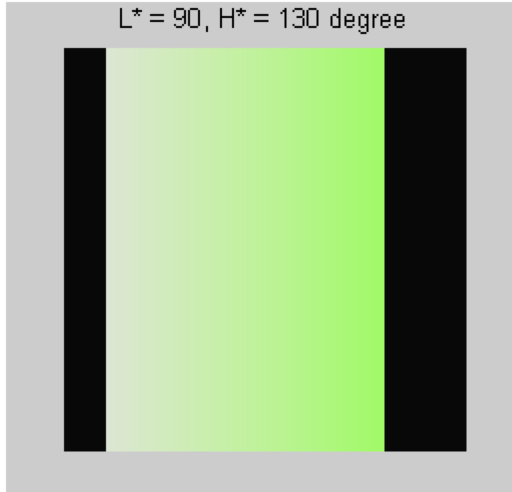

L* = 70, H* = 120 degree

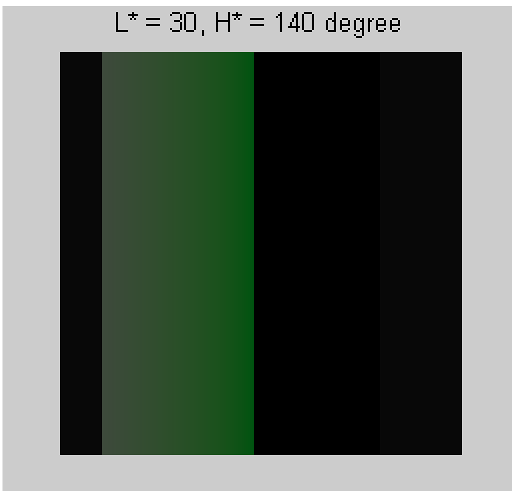

L* = 90, H* = 120 degree

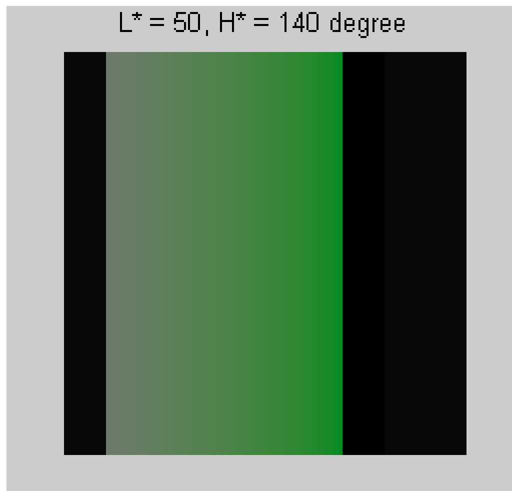

L* = 30, H* = 130 degree

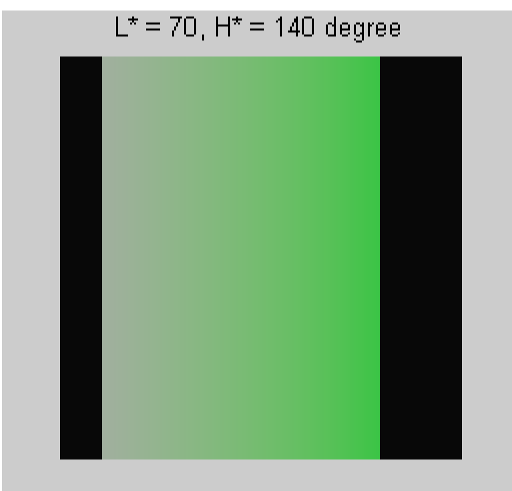

L* = 50, H* = 130 degree

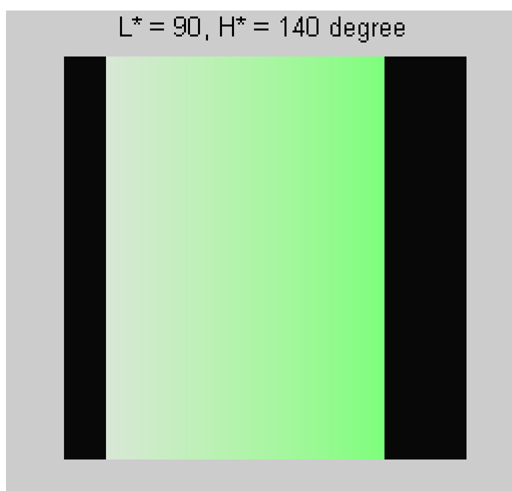

L* = 70, H* = 130 degree

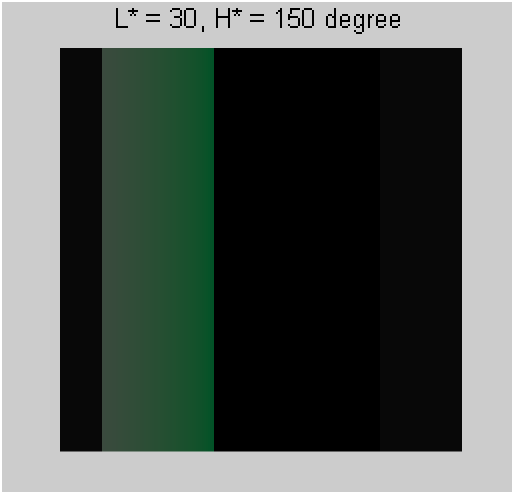

L* = 90, H* = 130 degree

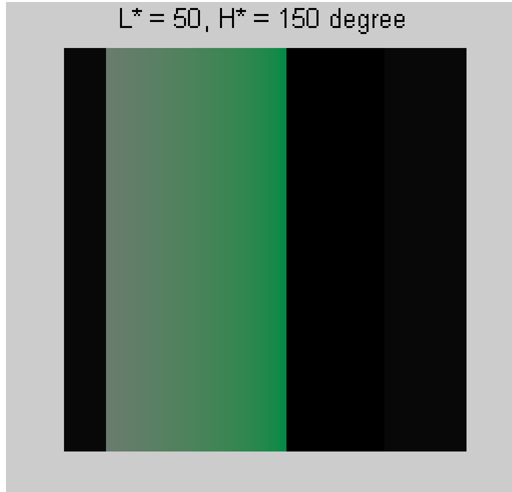

L* = 30, H* = 140 degree

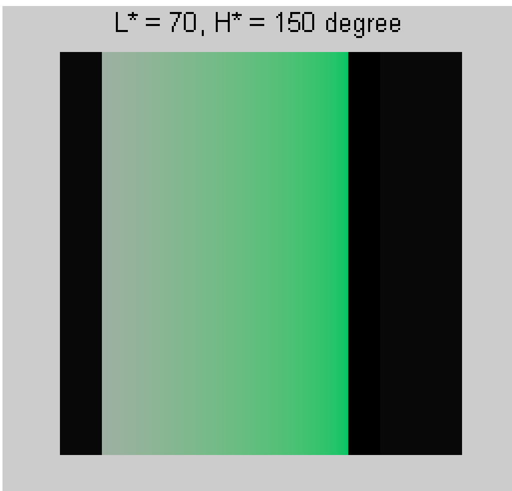

L* = 50, H* = 140 degree

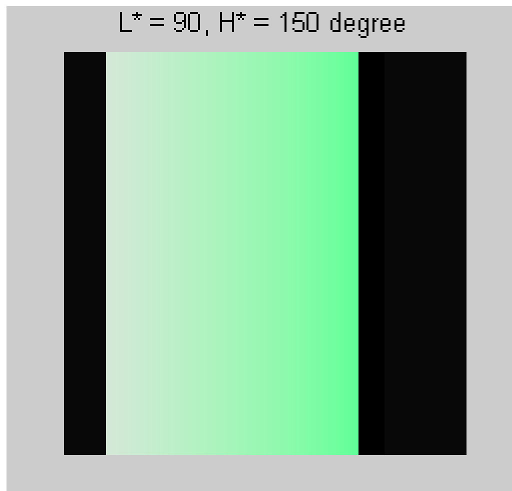

L* = 70, H* = 140 degree


L* = 90, H* = 140 degree



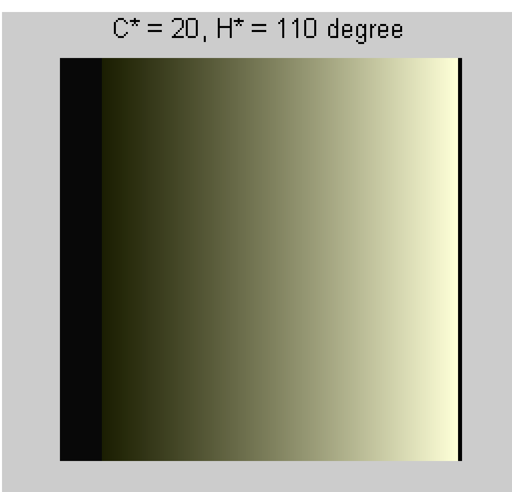


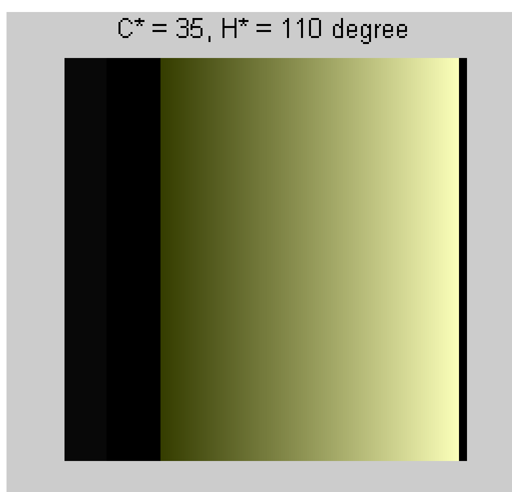


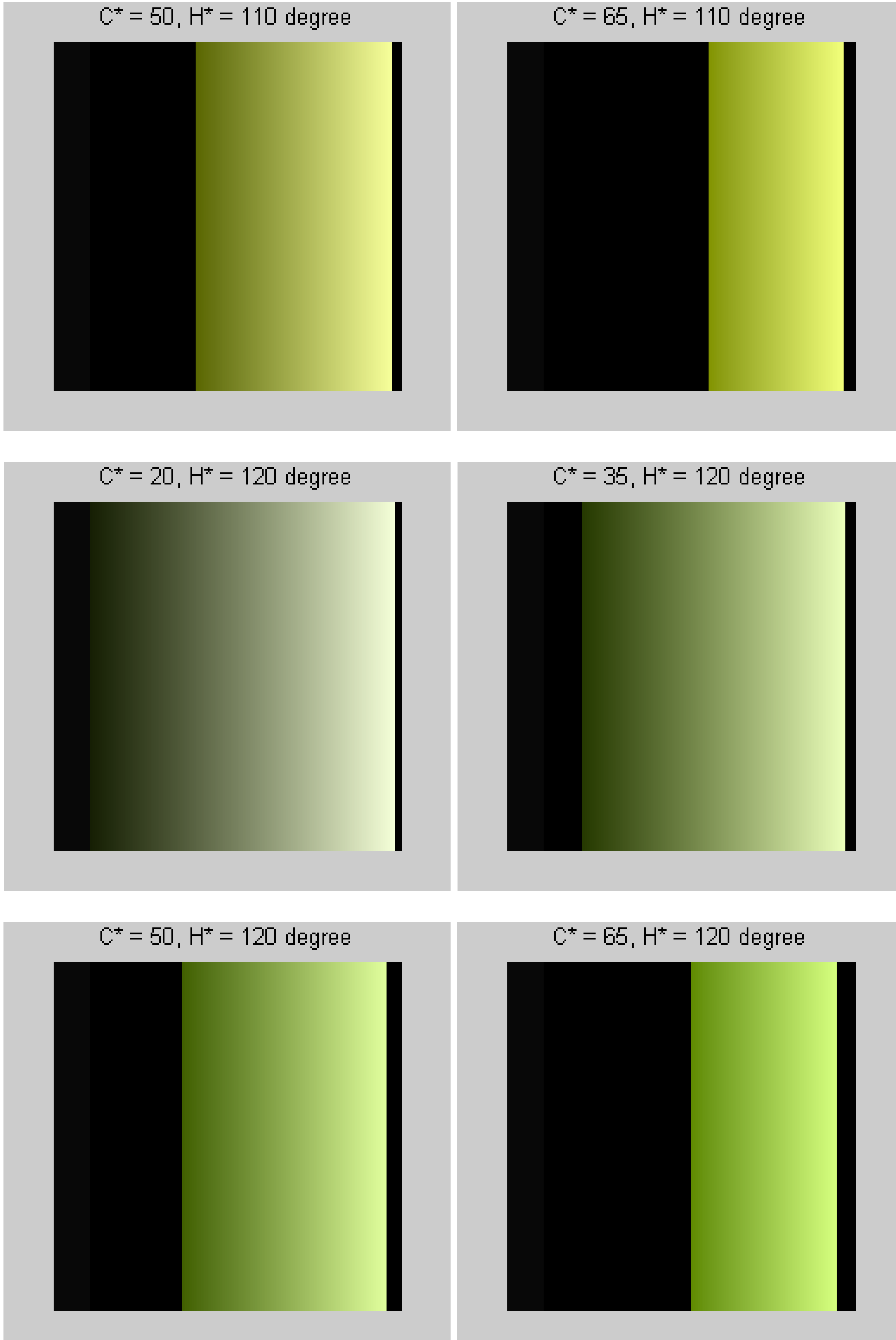


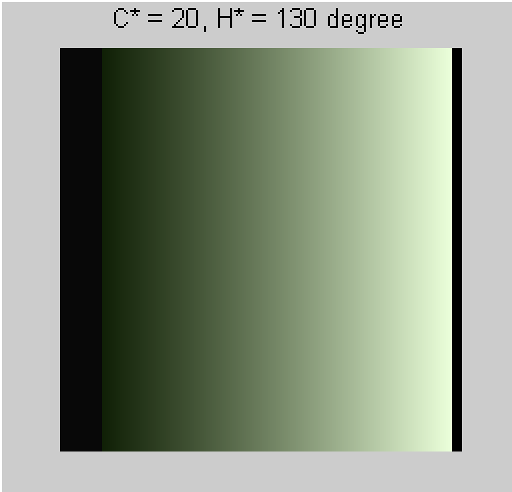
Colours with same hue and saturation but varying brightness

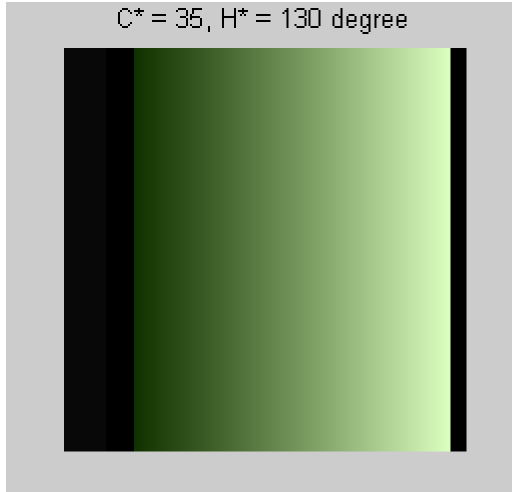
(For survey 1 in the Appendix C)

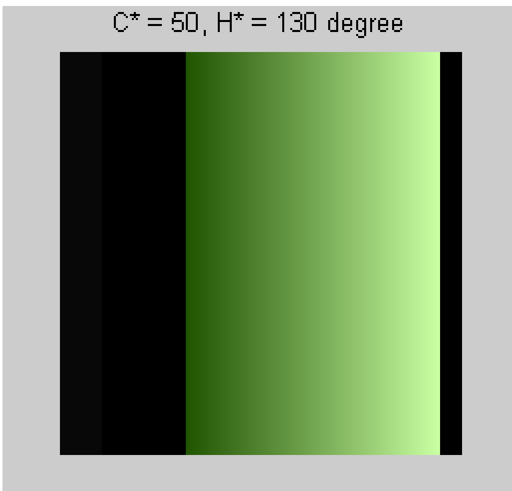


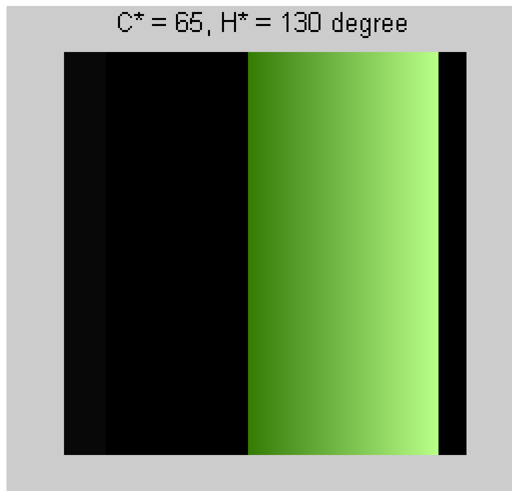


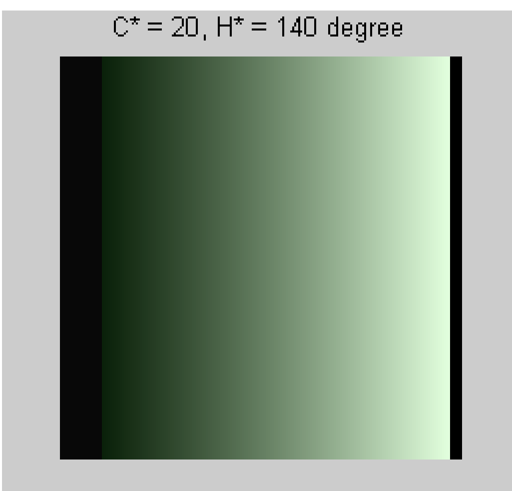

C* = 50, H* = 110 degree
C* = 65, H* = 110 degree
C* = 20, H* = 120 degree
C* = 35, H* = 120 degree
C* = 50, H* = 120 degree
C* = 65, H* = 120 degree

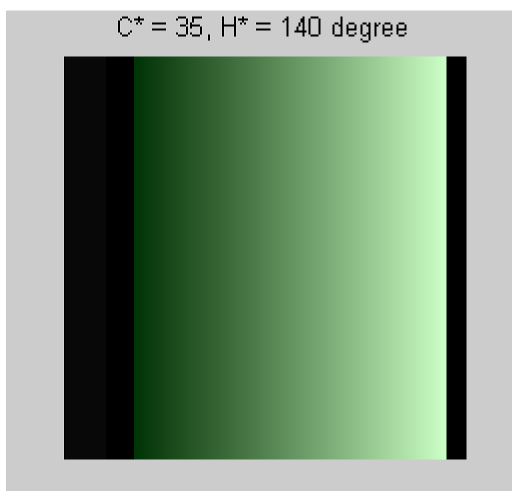

C* = 20, H* = 130 degree

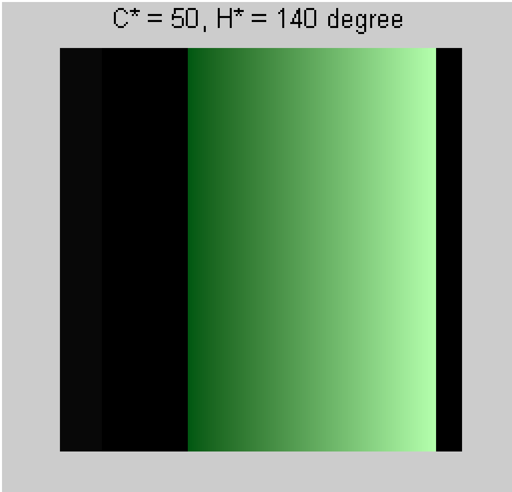

C* = 35, H* = 130 degree

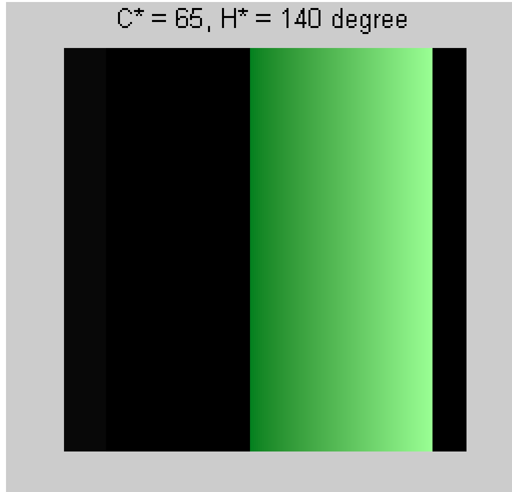

C* = 50, H* = 130 degree

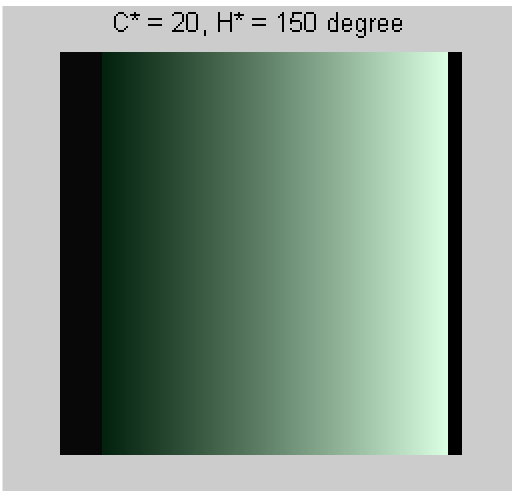

C* = 65, H* = 130 degree

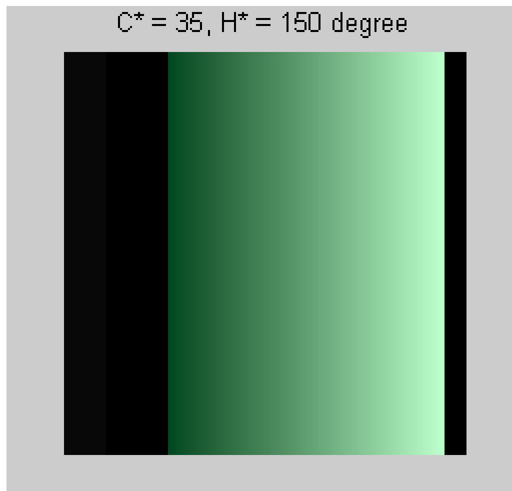

C* = 20, H* = 140 degree

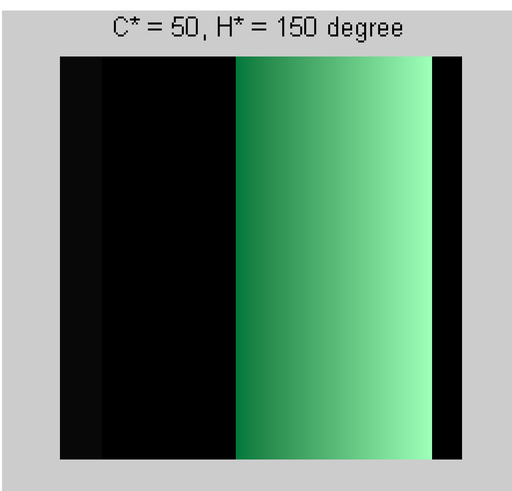

C* = 35, H* = 140 degree

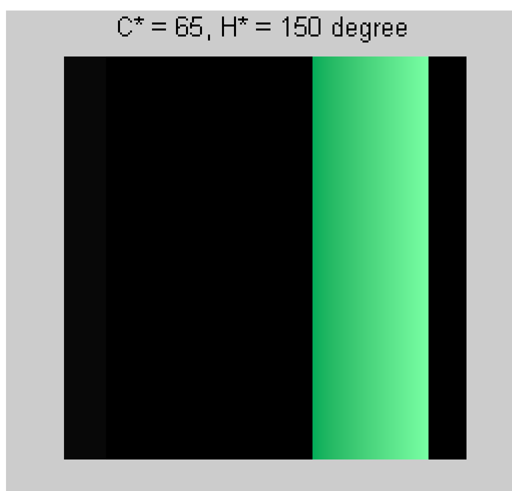

C* = 50, H* = 140 degree


C* = 65, H* = 140 degree

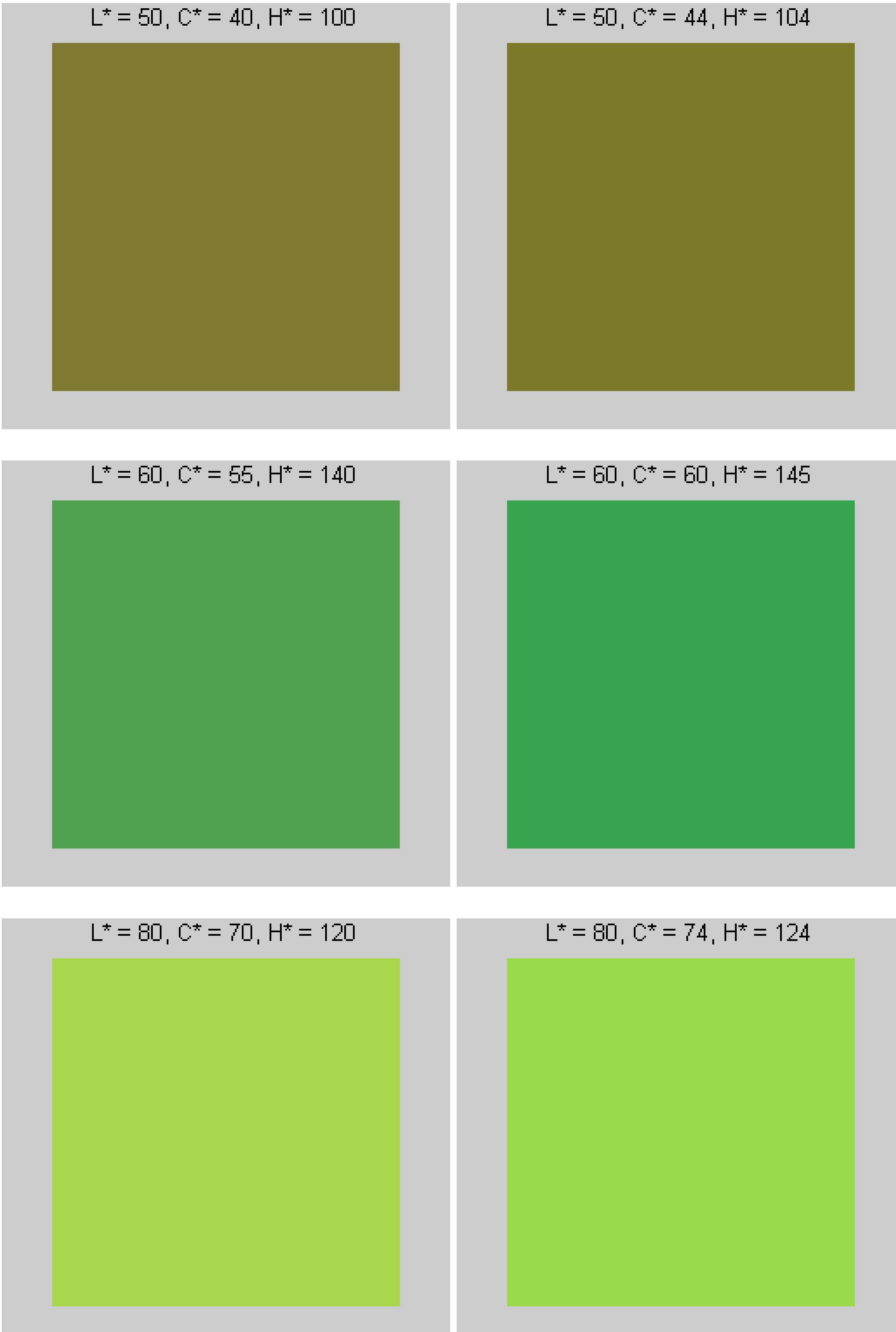

C* = 20, H* = 150 degree

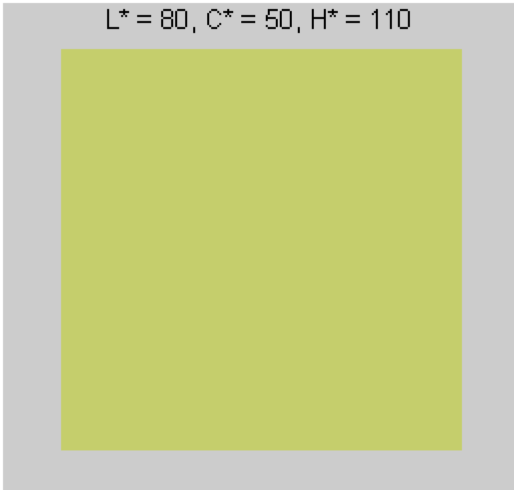

C* = 35, H* = 150 degree

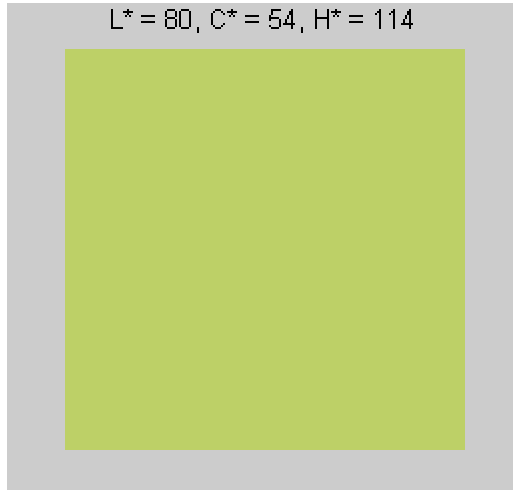

C* = 50, H* = 150 degree


C* = 65, H* = 150 degree

Colour difference between big hue and saturation and small hue and saturation




L* = 80, C* = 50, H* = 110


L* = 80, C* = 54, H* = 114

Appendix C

Surveys and results

Survey 1

The purpose of this survey is to check how the saturation affects colour appearance if the values of hue and brightness are constant.

(Sample colours are listed in the Appendix B - Colours with same hue and brightness but varying saturation)

Question:

Which part of the image is greener or yellower? (Please choose one answer)

1. Left is greener. 2. Left is yellower. 3. Right is greener
4. Right is yellower. 5. They are same. 6. Don't know

Result:

Images	Person 1	Person 2	Person 3	Person 4	Person 5
L*=30, H*=110	3	3	3	3	3
L*=50, H*=110	3	3	3	3	3
L*=70, H*=110	3	3	3	3	3
L*=90, H*=110	3	3	3	3	3

Result: (Continued)

Images	Person 1	Person 2	Person 3	Person 4	Person 5
L*=30, H*=120	3	3	3	3	3
L*=50, H*=120	3	3	3	3	3
L*=70, H*=120	3	3	3	3	3
L*=90, H*=120	3	3	3	3	3
L*=30, H*=130	3	3	3	3	3
L*=50, H*=130	3	3	3	3	3
L*=70, H*=130	3	3	3	3	3
L*=90, H*=130	3	3	3	3	3
L*=30, H*=140	3	3	3	3	3
L*=50, H*=140	3	3	3	3	3
L*=70, H*=140	3	3	3	3	3
L*=90, H*=140	3	3	3	3	3
L*=30, H*=150	3	3	3	3	3
L*=50, H*=150	3	3	3	3	3
L*=70, H*=150	3	3	3	3	3
L*=90, H*=150	3	3	3	3	3

Survey 2

The purpose of this survey is to check how the brightness affects colour appearance if the values of hue and saturation are constant.

(Sample colours are listed in the Appendix B - Colours with same hue and saturation but varying brightness)

Question:

Which part of the image is greener or yellower? (Please choose one answer)

1. Left is greener. 2. Left is yellower. 3. Right is greener
4. Right is yellower. 5. They are same. 6. Don't know

Result:

Images	Person 1	Person 2	Person 3	Person 4	Person 5
L*=20, H*=110	1	1	1	1	1
L*=35, H*=110	1	1	1	1	1
L*=50, H*=110	1	1	1	1	1
L*=65, H*=110	1	1	1	1	1
L*=20, H*=120	1	1	1	1	1
L*=35, H*=120	1	1	1	1	1
L*=50, H*=120	1	1	1	1	1
L*=65, H*=120	1	1	1	1	1
L*=20, H*=130	1	1	1	1	1
L*=35, H*=130	1	1	1	1	1
L*=50, H*=130	1	1	1	1	1
L*=65, H*=130	1	1	1	1	1
L*=20, H*=140	1	1	1	1	1
L*=35, H*=140	1	1	1	1	1
L*=50, H*=140	1	1	1	1	1
L*=65, H*=140	1	1	1	1	1
L*=20, H*=150	1	1	1	1	1
L*=35, H*=150	1	1	1	1	1
L*=50, H*=150	1	1	1	1	1
L*=65, H*=150	1	1	1	1	1

Survey 3

The purpose of this survey is to check how the saturation affects the colour appearance if the value of hue varies within 9 degrees and the value of brightness is constant. People are asked to measure whether colours have small value of hue but big value of saturation are greener than colours have big value of hue but small value of saturation.

	Hue	Saturation of colour patch with small hue is greener than 100 hue value (Brightness is 40)											
	100	10	14	18	22	26	30	34	38	42	46	-	-
Average	97	14	19	25	33	42	-	-	-	-	-	-	-
Average	94	19	31	-	-	-	-	-	-	-	-	-	-
Average	91	24	-	-	-	-	-	-	-	-	-	-	-
	Hue	Saturation of colour patch with small hue is greener than 115 hue value (Brightness is 40)											
	115	10	14	18	22	26	30	34	38	42	46	50	-
Average	112	15	20	25	31	41	50	-	-	-	-	-	-
Average	109	19	32	45	-	-	-	-	-	-	-	-	-
Average	106	26	46	-	-	-	-	-	-	-	-	-	-
	Hue	Saturation of colour patch with small hue is greener than 128 hue value (Brightness is 40)											
	128	10	14	18	22	26	30	34	38	42	46	50	54
Average	125	15	19	26	32	41	49	-	-	-	-	-	-
Average	122	17	30	45	-	-	-	-	-	-	-	-	-
Average	119	25	49	-	-	-	-	-	-	-	-	-	-
	Hue	Saturation of colour patch with small hue is greener than 145 hue value (Brightness is 40)											
	145	10	14	18	22	26	30	34	38	42	46	50	54
Average	142	15	19	26	33	40	49	59	-	-	-	-	-
Average	139	18	29	46	-	-	-	-	-	-	-	-	-

Average	136	24	47	-	-	-	-	-	-	-	-	-	-

	Hue	Saturation of colour patch with small hue is greener than 100 hue value (Brightness is 55)											
	100	10	14	18	22	26	30	34	38	42	46	50	54
Average	97	14	18	24	33	42	48	60	-	-	-	-	-
Average	94	19	32	47	-	-	-	-	-	-	-	-	-
Average	91	24	50	-	-	-	-	-	-	-	-	-	-
	Hue	Saturation of colour patch with small hue is greener than 115 hue value (Brightness is 55)											
	115	10	14	18	22	26	30	34	38	42	46	50	54
Average	112	15	20	25	31	41	49	61	-	-	-	-	-
Average	109	19	32	47	-	-	-	-	-	-	-	-	-
Average	106	26	48	-	-	-	-	-	-	-	-	-	-
	Hue	Saturation of colour patch with small hue is greener than 128 hue value (Brightness is 55)											
	128	10	14	18	22	26	30	34	38	42	46	50	54
Average	125	14	19	26	32	41	49	59	70	-	-	-	-
Average	122	18	30	45	68	-	-	-	-	-	-	-	-
Average	119	25	50	-	-	-	-	-	-	-	-	-	-
	Hue	Saturation of colour patch with small hue is greener than 145 hue value (Brightness is 55)											
	145	10	14	18	22	26	30	34	38	42	46	50	54
Average	142	14	19	26	33	40	49	59	-	-	-	-	-
Average	139	18	28	46	-	-	-	-	-	-	-	-	-
Average	136	24	48	-	-	-	-	-	-	-	-	-	-

	Hue	Saturation of colour patch with small hue is greener than 100 hue value (Brightness is 85)										
	100	10	14	18	22	26	30	34	38	42	46	50	54
Average	97	15	20	25	30	38	49	61	71	82	-	-	-
Average	94	19	32	47	68	81	-	-	-	-	-	-	-
Average	91	24	50	79	-	-	-	-	-	-	-	-	-
	Hue	Saturation of colour patch with small hue is greener than 115 hue value (Brightness is 85)											
	115	10	14	18	22	26	30	34	38	42	46	50	54
Average	112	14	19	27	32	39	47	62	72	83	-	-	-
Average	109	19	31	47	69	82	-	-	-	-	-	-	-
Average	106	25	49	80	-	-	-	-	-	-	-	-	-
	Hue	Saturation of colour patch with small hue is greener than 128 hue value (Brightness is 85)											
	128	10	14	18	22	26	30	34	38	42	46	50	54
Average	125	14	19	25	33	41	49	59	70	81	-	-	-
Average	122	18	30	45	68	83	-	-	-	-	-	-	-
Average	119	24	49	84	-	-	-	-	-	-	-	-	-
	Hue	Saturation of colour patch with small hue is greener than 145 hue value (Brightness is 85)											
	145	10	14	18	22	26	30	34	38	42	46	50	54
Average	142	15	19	26	33	40	49	59	-	-	-	-	-
Average	139	18	28	46	72	-	-	-	-	-	-	-	-
Average	136	26	50	82	-	-	-	-	-	-	-	-	-

	Hue	Saturation of colour patch with small hue is greener than 100 hue value (Brightness is 70)											
	100	10	14	18	22	26	30	34	38	42	46	50	54
Average	97	14	20	26	31	40	49	60	71	84	-	-	-
Average	94	19	32	47	68	83	-	-	-	-	-	-	-
Average	91	24	50	85	-	-	-	-	-	-	-	-	-
	Hue	Saturation of colour patch with small hue is greener than 115 hue value (Brightness is 70)											
	115	10	14	18	22	26	30	34	38	42	46	50	54
Average	112	14	19	27	32	41	48	61	72	83	-	-	-
Average	109	19	32	47	69	82	-	-	-	-	-	-	-
Average	106	26	48	79	-	-	-	-	-	-	-	-	-
	Hue	Saturation of colour patch with small hue is greener than 128 hue value (Brightness is 70)											
	128	10	14	18	22	26	30	34	38	42	46	50	54
Average	125	14	19	25	33	41	49	59	70	85	-	-	-
Average	122	19	31	46	67	85	-	-	-	-	-	-	-
Average	119	24	49	82	-	-	-	-	-	-	-	-	-
	Hue	Saturation of colour patch with small hue is greener than 145 hue value (Brightness is 70)											
	145	10	14	18	22	26	30	34	38	42	46	50	54
Average	142	14	19	26	33	40	49	59	71	84	-	-	-
Average	139	18	28	46	71	83	-	-	-	-	-	-	-
Average	136	24	48	81	-	-	-	-	-	-	-	-	-

Survey 4

The purpose of this survey is to check how the brightness affects the colour appearance if the value of hue varies within 9 degrees and the value of saturation is constant. People are asked to measure whether colours with small values of hue and saturation are greener than colours with big values of hue and saturation.

	Hue	Brightness of colour patch with small hue is greener than 138 hue value (saturation is 40)											
	138	90	86	82	78	74	70	66	62	58	54	50	46
Average	136	84	78	70	62	54	45	33	-	-	-	-	-
Average	134	75	64	55	-	-	-	-	-	-	-	-	-
Average	132	66	53	-	-	-	-	-	-	-	-	-	-
Average	130	59											

	Hue	Brightness of colour patch with small hue is greener than 118 hue value (saturation is 40)											
	138	90	86	82	78	74	70	66	62	58	54	50	46
Average	116	84	76	66	62	53	43	36	-	-	-	-	-
Average	114	75	64	53	-	-	-	-	-	-	-	-	-
Average	112	69	54	-	-	-	-	-	-	-	-	-	-
Average	110	55											

	Hue	Brightness of colour patch with small hue is greener than 98 hue value (saturation is 40)											
	98	90	86	82	78	74	70	66	62	58	54	50	46
Average	96	84	77	70	62	52	44	-	-	-	-	-	-
Average	94	78	67	51	-	-	-	-	-	-	-	-	-
Average	92	68	52	-	-	-	-	-	-	-	-	-	-
Average	90	58											

	Hue	Brightness of colour patch with small hue is greener than 138 hue value (Saturation is 50)										
	138	90	86	82	78	74	70	66	62	58	54	50	46
Average	136	84	77	69	62	53	43	33	-	-	-	-	-
Average	134	76	63	51	-	-	-	-	-	-	-	-	-
Average	132	69	54	-	-	-	-	-	-	-	-	-	-
Average	130	55	-	-	-	-	-	-	-	-	-	-	-

	Hue	Brightness of colour patch with small hue is greener than 118 hue value (Saturation is 50)											
	118	90	86	82	78	74	70	66	62	58	54	50	46
Average	116	85	76	69	62	54	43	35	-	-	-	-	-
Average	114	77	65	53	-	-	-	-	-	-	-	-	-
Average	112	70	56	-	-	-	-	-	-	-	-	-	-
Average	110	59	-	-	-	-	-	-	-	-	-	-	-

	Hue	Brightness of colour patch with small hue is greener than 98 hue value (saturation is 50)											
	98	90	86	82	78	74	70	66	62	58	54	50	46
Average	96	84	78	71	62	55	46	34	-	-	-	-	-
Average	94	78	66	54	-	-	-	-	-	-	-	-	-
Average	92	68	57	-	-	-	-	-	-	-	-	-	-
Average	90	58	-	-	-	-	-	-	-	-	-	-	-

	Hue	Brightness of colour patch with small hue is greener than 138 hue value (saturation is 65)											
	138	90	86	82	78	74	70	66	62	58	54	50	46
Average	136	86	79	72	66	58	47	-	-	-	-	-	-
Average	134	79	65	53	-	-	-	-	-	-	-	-	-
Average	132	71	56	-	-	-	-	-	-	-	-	-	-
Average	130	-	-	-	-	-	-	-	-	-	-	-	-

	Hue	Brightness of colour patch with small hue is greener than 118 hue value (saturation is 65)											
	118	90	86	82	78	74	70	66	62	58	54	50	46
Average	116	85	79	71	65	57	-	-	-	-	-	-	-
Average	114	75	64	54	-	-	-	-	-	-	-	-	-
Average	112	68	57	-	-	-	-	-	-	-	-	-	-
Average	110	-	-	-	-	-	-	-	-	-	-	-	-

	Hue	Brightness of colour patch with small hue is greener than 98 hue value (saturation is 65)											
	98	90	86	82	78	74	70	66	62	58	54	50	46
Average	96	84	78	71	63	-	-	-	-	-	-	-	-
Average	94	76	65	-	-	-	-	-	-	-	-	-	-
Average	92	68	-	-	-	-	-	-	-	-	-	-	-
Average	90	-	-	-	-	-	-	-	-	-	-	-	-

Appendix D

Source codes of VB

Test.frm

```
Option Explicit
'Define the global variables of sample colours
Dim sample_l(4) As Single
Dim sample_a(4) As Single
Dim sample_b(4) As Single

'Function of the 'Result' button
Private Sub Command1_Click()

Dim pixels() As RGBTriplet
Dim bits_per_pixel As Integer
Dim C_lab As CIELAB
Dim CIEL, CIEa, CIEb, CIEC, CIEH As Single
Dim Colour_check As Integer

Dim X, Y As Integer         'Size of the image
Dim i As Integer            'counter
Dim total As Long           'Record the pixels of image
Dim Background As Long      'Record the pixels of background
Dim grade(5) As Long        'Record the pixels of each quality
Dim Colour(5) As Long       'Percentage of each quality
Dim number_colour As Long   'Numbers of pixel of the foreground

'Initialize value
total = 0
Background = 0
number_colour = 0
```

```
PicOriginal.ScaleMode = vbPixels

'Get the pixels from PicOriginal

GetBitmapPixels PicOriginal, pixels, bits_per_pixel
'set the pixel colors
For Y = 0 To PicOriginal.ScaleHeight - 1
   For X = 0 To PicOriginal.ScaleWidth - 1
      With pixels(X, Y)

      'Convert LAB to LCH
      C_lab = ConvertRGBtoLab(.rgbRed, .rgbGreen, .rgbBlue)
      CIEC = Sqr(C_lab.a ^ 2 + C_lab.b ^ 2)
      If C_lab.b <= 0 Then
         CIEH = 0    'Hue is zero if 'b' is less than 0
      Else
         If C_lab.a > 0 Then
            CIEH = (Atn(C_lab.b / C_lab.a)) * 180 / 3.1416
         Else
            CIEH = (3.1416 + Atn(C_lab.b / C_lab.a)) * 180 / 3.1416
         End If
      End If

      CIEL = C_lab.l
      CIEa = C_lab.a
      CIEb = C_lab.b

   If CIEC <= 10 Or CIEL <= 10 Or CIEb <= 0 Then

      'Separate colours from background
      Background = Background + 1
      .rgbRed = 0
      .rgbGreen = 0
      .rgbBlue = 0
   Else
      If CIEH > 150 And CIEH <= 80 Then

         'Colours are not green and yellow are background
         Background = Background + 1
         .rgbRed = 0
         .rgbGreen = 0
         .rgbBlue = 0
      ElseIf ColourDifference(C_lab.l, C_lab.a, _
        C_lab.b, sample_l(4), sample_a(4), sample_b(4)) = False Then

         'Count colours with 'Very Poor' quality
         grade(5) = grade(5) + 1
         number_colour = number_colour + 1
         .rgbRed = 255
         .rgbGreen = 192
         .rgbBlue = 127
```

```
        ElseIf ColourDifference(C_lab.l, C_lab.a, _
          C_lab.b, sample_l(3), sample_a(3), sample_b(3)) = False Then

            'Count colours with 'Poor' quality
            grade(4) = grade(4) + 1
            number_colour = number_colour + 1
            .rgbRed = 255
            .rgbGreen = 255
            .rgbBlue = 127
        ElseIf ColourDifference(C_lab.l, C_lab.a, _
          C_lab.b, sample_l(2), sample_a(2), sample_b(2)) = False Then

            'Count colours with 'Fair' quality
            grade(3) = grade(3) + 1
            number_colour = number_colour + 1
            .rgbRed = 127
            .rgbGreen = 127
            .rgbBlue = 0
        ElseIf ColourDifference(C_lab.l, C_lab.a, _
          C_lab.b, sample_l(1), sample_a(1), sample_b(1)) = False Then

            'Count colours with 'Good' quality
            grade(2) = grade(2) + 1
            number_colour = number_colour + 1
            .rgbRed = 127
            .rgbGreen = 255
            .rgbBlue = 127
        ElseIf ColourDifference(C_lab.l, C_lab.a, _
          C_lab.b, sample_l(0), sample_a(0), sample_b(0)) = False Then
            grade(1) = grade(1) + 1

            'Count colours with 'Very Good' quality
            number_colour = number_colour + 1
            .rgbRed = 0
            .rgbGreen = 192
            .rgbRed = 0
        Else

            'Count colours with 'Excellent' quality
            grade(0) = grade(0) + 1
            number_colour = number_colour + 1
            .rgbRed = 0
            .rgbGreen = 127
            .rgbBlue = 0
        End If
    End If
        End With
    Next X
Next Y

'Set the output image has the same size with the original
```

```
PicResult.AutoSize = True
PicResult.Height = PicOriginal.Height
PicResult.Width = PicOriginal.Width
PicResult.Cls
PicResult.ScaleMode = vbPixels

SetBitmapPixels PicResult, bits_per_pixel, pixels
PicResult.Visible = False
OutputImg.Height = InputImg.Height
OutputImg.Width = InputImg.Width
PicResult.Picture = PicResult.Image

OutputImg.Stretch = True
OutputImg.Picture = PicResult.Picture


'Count the percentage of different qualities and display
total = PicOriginal.ScaleHeight * PicOriginal.ScaleWidth

For i = 0 To 5
   Colour(i) = CInt(grade(i) / (total - Background + 1) * 100)
Next

Picture1.Visible = True
Picture2.Visible = True
Picture3.Visible = True
Picture4.Visible = True
Picture5.Visible = True
Picture6.Visible = True
Grade01.Caption = "Excellent: " & Colour(0) & "%"
Grade02.Caption = "Very good: " & Colour(1) & "%"
Grade03.Caption = "Good:      " & Colour(2) & "%"
Grade04.Caption = "Fair:      " & Colour(3) & "%"
Grade05.Caption = "Poor:      " & Colour(4) & "%"
Grade06.Caption = "Very Poor: " & Colour(5) & "%"
ColourPer.Caption = "Colour percentage of the picture = " _
  & CInt(number_colour * 100 / total) & "%"


End Sub

'Definition of the sample colour with grade 1
Private Sub Command2_Click()
Dim CIEL, CIEa, CIEb As Single
Dim rgbcolour As RGBTriplet

check = 1   'Flag to check whether the 'OK' of sub-menu is clicked
NumberOfSubmenu = 0
DefineValueCIE.Cancel.Cancel = True
Load DefineValueCIE            'Load the sub-menu
DefineValueCIE.Show vbModal    'Pop-up the sub-menu
```

```
If check = 1 Then

   'Display the new defined value of sample colour
   Sampic(0).BackColor = rgb(DefineValueCIE.HScrollRed.value, _
     DefineValueCIE.HScrollGreen.value, DefineValueCIE.HScrollBlue.value)
   Label(0).Caption = "L*=" & DefineValueCIE.HScrollCIEL.value _
     & "; a*=" & DefineValueCIE.HScrollCIEa.value & "; b*=" _
     & DefineValueCIE.HScrollCIEb.value
   sample_l(0) = DefineValueCIE.HScrollCIEL.value
   sample_a(0) = DefineValueCIE.HScrollCIEa.value
   sample_b(0) = DefineValueCIE.HScrollBlue.value
End If

Unload DefineValueCIE

End Sub

'Definition of the sample colour with grade 2
Private Sub Command3_Click()
Dim CIEL, CIEa, CIEb As Single
Dim rgbcolour As RGBTriplet
Dim Colour As HSV

check = 1   'Flag to check whether the 'OK' of sub-menu is clicked
NumberOfSubmenu = 1
DefineValueCIE.Cancel.Cancel = True
Load DefineValueCIE
DefineValueCIE.Show vbModal

If check = 1 Then

   'Display the new defined value of sample colour
   Sampic(1).BackColor = rgb(DefineValueCIE.HScrollRed.value, _
     DefineValueCIE.HScrollGreen.value, DefineValueCIE.HScrollBlue.value)
   Label(1).Caption = "L*=" & DefineValueCIE.HScrollCIEL.value _
     & "; a*=" & DefineValueCIE.HScrollCIEa.value & "; b*=" _
     & DefineValueCIE.HScrollCIEb.value
   sample_l(1) = DefineValueCIE.HScrollCIEL.value
   sample_a(1) = DefineValueCIE.HScrollCIEa.value
   sample_b(1) = DefineValueCIE.HScrollBlue.value
End If

Unload DefineValueCIE

End Sub

'Definition of the sample colour with grade 3
Private Sub Command4_Click()
Dim CIEL, CIEa, CIEb As Single
Dim rgbcolour As RGBTriplet
```

```
Dim Colour As HSV

check = 1   'Flag to check whether the 'OK' of sub-menu is clicked
NumberOfSubmenu = 2
DefineValueCIE.Cancel.Cancel = True
Load DefineValueCIE
DefineValueCIE.Show vbModal

If check = 1 Then

   'Display the new defined value of sample colour
   Sampic(2).BackColor = rgb(DefineValueCIE.HScrollRed.value, _
     DefineValueCIE.HScrollGreen.value, DefineValueCIE.HScrollBlue.value)
   Label(2).Caption = "L*=" & DefineValueCIE.HScrollCIEL.value _
     & "; a*=" & DefineValueCIE.HScrollCIEa.value & "; b*=" _
     & DefineValueCIE.HScrollCIEb.value
   sample_l(2) = DefineValueCIE.HScrollCIEL.value
   sample_a(2) = DefineValueCIE.HScrollCIEa.value
   sample_b(2) = DefineValueCIE.HScrollBlue.value
End If

Unload DefineValueCIE

End Sub

'Definition of the sample colour with grade 4
Private Sub Command5_Click()
Dim CIEL, CIEa, CIEb As Single
Dim rgbcolour As RGBTriplet
Dim Colour As HSV

check = 1   'Flag to check whether the 'OK' of sub-menu is clicked
NumberOfSubmenu = 3
DefineValueCIE.Cancel.Cancel = True
Load DefineValueCIE
DefineValueCIE.Show vbModal

If check = 1 Then

   'Display the new defined value of sample colour
   Sampic(3).BackColor = rgb(DefineValueCIE.HScrollRed.value, _
     DefineValueCIE.HScrollGreen.value, DefineValueCIE.HScrollBlue.value)
   Label(3).Caption = "L*=" & DefineValueCIE.HScrollCIEL.value _
     & "; a*=" & DefineValueCIE.HScrollCIEa.value & "; b*=" _
     & DefineValueCIE.HScrollCIEb.value
   sample_l(3) = DefineValueCIE.HScrollCIEL.value
   sample_a(3) = DefineValueCIE.HScrollCIEa.value
   sample_b(3) = DefineValueCIE.HScrollBlue.value

End If
```

```
Unload DefineValueCIE

End Sub

'Definition of the sample colour with grade 5
Private Sub Command6_Click()
Dim CIEL, CIEa, CIEb As Single
Dim rgbcolour As RGBTriplet
Dim Colour As HSV

check = 1   'Flag to check whether the 'OK' of sub-menu is clicked
NumberOfSubmenu = 4
DefineValueCIE.Cancel.Cancel = True
Load DefineValueCIE
DefineValueCIE.Show vbModal

If check = 1 Then

  'Display the new defined value of sample colour
  Sampic(4).BackColor = rgb(DefineValueCIE.HScrollRed.value, _
    DefineValueCIE.HScrollGreen.value, DefineValueCIE.HScrollBlue.value)
  Label(4).Caption = "L*=" & DefineValueCIE.HScrollCIEL.value & "; a*=" _
    & DefineValueCIE.HScrollCIEa.value & "; b*=" _
    & DefineValueCIE.HScrollCIEb.value
  sample_l(4) = DefineValueCIE.HScrollCIEL.value
  sample_a(4) = DefineValueCIE.HScrollCIEa.value
  sample_b(4) = DefineValueCIE.HScrollBlue.value
End If

Unload DefineValueCIE

End Sub

'Exit the application as the 'Exit' is clicked
Private Sub mnuFileExit_Click()
  End
End Sub

' Load the indicated file.
Private Sub mnuFileOpen_Click()
Dim file_name As String
Dim Width, Height As Integer

  ' Let the user select a file.
  On Error Resume Next
  dlgOpenFile.Flags = cdlOFNFileMustExist + cdlOFNHideReadOnly
  dlgOpenFile.ShowOpen
  If Err.Number = cdlCancel Then
    Exit Sub
  ElseIf Err.Number <> 0 Then
    Beep
```

```
   MsgBox "Error selecting file.", , vbExclamation
   Exit Sub
End If
On Error GoTo 0

Screen.MousePointer = vbHourglass
DoEvents

file_name = Trim$(dlgOpenFile.FileName)
dlgOpenFile.InitDir = Left$(file_name, Len(file_name) _
   - Len(dlgOpenFile.FileTitle) - 1)
Caption = "Bright [" & dlgOpenFile.FileTitle & "]"

' Open the original file.
On Error GoTo LoadError
PicOriginal.Visible = False
PicOriginal.Picture = LoadPicture(file_name)


'Check the Input image has the same size with the PicOriginal

PicOriginal.ScaleMode = vbTwips
Height = PicOriginal.ScaleHeight
Width = PicOriginal.ScaleWidth

If Height > 5652 Then
   If Width > 4332 Then
      If Height / Width > 5652 / 4332 Then
         InputImg.Height = 5652
         InputImg.Width = Width / Height * 5652
      Else
         InputImg.Width = 4332
         InputImg.Height = Height / Width * 4332
      End If
   Else
      InputImg.Height = 5652
      InputImg.Width = Width / Height * 5652

   End If
 Else
   If Width > 4332 Then
      InputImg.Width = 4332
      InputImg.Height = Height / Width * 4332
   Else
      InputImg.Width = Width
      InputImg.Height = Height
   End If
End If

InputImg.Stretch = True
InputImg.Picture = PicOriginal.Picture
```

```
    On Error GoTo 0

    Screen.MousePointer = vbDefault
    Exit Sub

LoadError:
    Screen.MousePointer = vbDefault
    MsgBox "Error " & Format$(Err.Number) & _
        " opening file '" & file_name & "'" & vbCrLf & _
        Err.Description
End Sub

' Start in the current directory.
Private Sub Form_Load()
    Dim i As Integer
    For i = 0 To 4
        Sampic(i).ScaleMode = vbPixels
    Next
    PicOriginal.AutoSize = True
    PicOriginal.ScaleMode = vbPixels
    PicOriginal.AutoRedraw = True

    dlgOpenFile.CancelError = True
    dlgOpenFile.InitDir = App.Path
    dlgOpenFile.Filter = _
        "Bitmaps (*.bmp)|*.bmp|" & _
        "GIFs (*.gif)|*.gif|" & _
        "JPEGs (*.jpg)|*.jpg;*.jpeg|" & _
        "Icons (*.ico)|*.ico|" & _
        "Cursors (*.cur)|*.cur|" & _
        "Run-Length Encoded (*.rle)|*.rle|" & _
        "Metafiles (*.wmf)|*.wmf|" & _
        "Enhanced Metafiles (*.emf)|*.emf|" & _
        "Graphic Files|*.bmp;*.gif;*.jpg;*.jpeg;*.ico;*.cur;*.rle;*.wmf;*.emf|" _
         & "All Files (*.*)|*.*"

        calcColor

End Sub

'calculate the value of R,G,B of the sample picture
Private Sub calcColor()

Dim pixels() As RGBTriplet
Dim bits_per_pixel As Integer
Dim total As Integer
Dim i As Integer
Dim Result_R As Integer
Dim Result_G As Integer
Dim Result_B As Integer
```

```
Dim Colour As HSV
Dim value_r(4) As Integer
Dim value_g(4) As Integer
Dim value_sample_b(4) As Integer
Dim lab As CIELAB

value_r(0) = 160
value_g(0) = 200
value_sample_b(0) = 80

value_r(1) = 162
value_g(1) = 188
value_sample_b(1) = 80

value_r(2) = 212
value_g(2) = 215
value_sample_b(2) = 93

value_r(3) = 212
value_g(3) = 204
value_sample_b(3) = 93

value_r(4) = 231
value_g(4) = 182
value_sample_b(4) = 77


For i = 0 To 4
   Sampic(i).ScaleMode = vbPixels
   Sampic(i).BackColor = rgb(value_r(i), value_g(i), value_sample_b(i))
   lab = ConvertRGBtoLab(value_r(i), value_g(i), value_sample_b(i))
   sample_l(i) = lab.l
   sample_a(i) = lab.a
   sample_b(i) = lab.b
   Label(i).Caption = "L*=" & Format(sample_l(i), "Fixed") & "; a*=" _
     & Format(sample_a(i), "fixed") & "; b*=" & Format(sample_b(i), "fixed")
Next i

End Sub
```

DefineValueCIE.frm

```
Option Explicit

'Function of 'Cancel' button
Private Sub Cancel_Click()
check = 0
Unload DefineValueCIE
End Sub

'Initialize the sub-menu
Private Sub Form_Load()
   Dim red, green, blue As Integer
   Dim lab As CIELAB
   red = Test.Sampic(NumberOfSubmenu).BackColor Mod 256
   green = ((Test.Sampic(NumberOfSubmenu).BackColor And "&HFF00FF00") / 256&)
   blue = (Test.Sampic(NumberOfSubmenu).BackColor And "&HFF0000") / 65536
   SampleColour.BackColor = rgb(red, green, blue)
   lab = ConvertRGBtoLab(red, green, blue)
   valuecheck = 1
   HScrollRed.value = red
   valuecheck = 1
   HScrollGreen.value = green
   valuecheck = 1
   HScrollBlue.value = blue
   valuecheck = 1
   HScrollCIEL.value = lab.l
   valuecheck = 1
   HScrollCIEa.value = lab.a
   valuecheck = 1
   HScrollCIEb.value = lab.b
   valuecheck = 0
   LabelRed.Caption = "Red = " & HScrollRed.value
   LabelGreen.Caption = "Green = " & HScrollGreen.value
   LabelBlue.Caption = "Blue = " & HScrollBlue.value
   LabelCIEL.Caption = "CIE L = " & HScrollCIEL.value
   LabelCIEa.Caption = "CIE a = " & HScrollCIEa.value
   LabelCIEb.Caption = "CIE b = " & HScrollCIEb.value
End Sub

Private Sub HScrollBlue_Change()
   If valuecheck <> 1 Then
      Dim lab As CIELAB
      SampleColour.Picture = LoadPicture()
      LabelBlue.Caption = "Blue = " & HScrollBlue.value
      lab = ConvertRGBtoLab(HScrollRed.value, _
        HScrollGreen.value, HScrollBlue.value)
      SampleColour.BackColor = rgb(HScrollRed.value, _
```

```
        HScrollGreen.value, HScrollBlue.value)
      valuecheck = 1
      HScrollCIEL.value = lab.l
      valuecheck = 1
      HScrollCIEa.value = lab.a
      valuecheck = 1
      HScrollCIEb.value = lab.b
      valuecheck = 0
      LabelCIEL.Caption = "CIE L = " & HScrollCIEL.value
      LabelCIEa.Caption = "CIE a = " & HScrollCIEa.value
      LabelCIEb.Caption = "CIE b = " & HScrollCIEb.value
      OK.Enabled = True
   Else
      valuecheck = 0
   End If
End Sub

Private Sub HScrollCIEa_Change()
   If valuecheck <> 1 Then
      Dim rgbvalue As RGBTriplet
      LabelCIEa.Caption = "CIE a = " & HScrollCIEa.value
      SampleColour.Picture = LoadPicture()
      rgbvalue = ConvertLabtoRGB(HScrollCIEL.value, _
        HScrollCIEa.value, HScrollCIEb.value)
      If rgbvalue.flag = -1 Then
         SampleColour.Picture = LoadPicture("caution.gif")
         OK.Enabled = False
      Else
         SampleColour.BackColor = rgb(rgbvalue.rgbRed, _
           rgbvalue.rgbGreen, rgbvalue.rgbBlue)
         OK.Enabled = True
      End If

      valuecheck = 1
      HScrollRed.value = CInt(rgbvalue.rgbRed)
      valuecheck = 1
      HScrollGreen.value = CInt(rgbvalue.rgbGreen)
      valuecheck = 1
      HScrollBlue.value = CInt(rgbvalue.rgbBlue)
      valuecheck = 0
      LabelRed.Caption = "Red = " & HScrollRed.value
      LabelGreen.Caption = "Green = " & HScrollGreen.value
      LabelBlue.Caption = "Blue = " & HScrollBlue.value
   Else
      valuecheck = 0
   End If
End Sub

Private Sub HScrollCIEb_Change()
If valuecheck <> 1 Then
      Dim rgbvalue As RGBTriplet
```

```
        LabelCIEb.Caption = "CIE b = " & HScrollCIEb.value
        SampleColour.Picture = LoadPicture()
        rgbvalue = ConvertLabtoRGB(HScrollCIEL.value, _
          HScrollCIEa.value, HScrollCIEb.value)
        If rgbvalue.flag = -1 Then
            SampleColour.Picture = LoadPicture("caution.gif")
            OK.Enabled = False
        Else
            SampleColour.BackColor = rgb(rgbvalue.rgbRed, _
              rgbvalue.rgbGreen, rgbvalue.rgbBlue)
            OK.Enabled = True
        End If

        valuecheck = 1
        HScrollRed.value = CInt(rgbvalue.rgbRed)
        valuecheck = 1
        HScrollGreen.value = CInt(rgbvalue.rgbGreen)
        valuecheck = 1
        HScrollBlue.value = CInt(rgbvalue.rgbBlue)
        valuecheck = 0
        LabelRed.Caption = "Red = " & HScrollRed.value
        LabelGreen.Caption = "Green = " & HScrollGreen.value
        LabelBlue.Caption = "Blue = " & HScrollBlue.value
    Else
        valuecheck = 0
    End If

End Sub

Private Sub HScrollCIEL_Change()
    If valuecheck <> 1 Then
        Dim rgbvalue As RGBTriplet
        SampleColour.Picture = LoadPicture()
        LabelCIEL.Caption = "CIE L = " & HScrollCIEL.value
        rgbvalue = ConvertLabtoRGB(HScrollCIEL.value, _
          HScrollCIEa.value, HScrollCIEb.value)
        If rgbvalue.flag = -1 Then
            SampleColour.Picture = LoadPicture("caution.gif")
            OK.Enabled = False
        Else
            SampleColour.BackColor = rgb(rgbvalue.rgbRed, _
              rgbvalue.rgbGreen, rgbvalue.rgbBlue)
            OK.Enabled = True
        End If

        valuecheck = 1
        HScrollRed.value = CInt(rgbvalue.rgbRed)
        valuecheck = 1
        HScrollGreen.value = CInt(rgbvalue.rgbGreen)
        valuecheck = 1
        HScrollBlue.value = CInt(rgbvalue.rgbBlue)
```

```
        valuecheck = 0
        LabelRed.Caption = "Red = " & HScrollRed.value
        LabelGreen.Caption = "Green = " & HScrollGreen.value
        LabelBlue.Caption = "Blue = " & HScrollBlue.value
    Else
        valuecheck = 0
    End If
End Sub

Private Sub HScrollGreen_Change()
    If valuecheck <> 1 Then
        Dim lab As CIELAB
        SampleColour.Picture = LoadPicture()
        LabelGreen.Caption = "Green = " & HScrollGreen.value
        lab = ConvertRGBtoLab(HScrollRed.value, _
          HScrollGreen.value, HScrollBlue.value)
        SampleColour.BackColor = rgb(HScrollRed.value, _
          HScrollGreen.value, HScrollBlue.value)
        valuecheck = 1
        HScrollCIEL.value = lab.l
        valuecheck = 1
        HScrollCIEa.value = lab.a
        valuecheck = 1
        HScrollCIEb.value = lab.b
        valuecheck = 0
        LabelCIEL.Caption = "CIE L = " & HScrollCIEL.value
        LabelCIEa.Caption = "CIE a = " & HScrollCIEa.value
        LabelCIEb.Caption = "CIE b = " & HScrollCIEb.value
        OK.Enabled = True
    Else
        valuecheck = 0
    End If
End Sub

Private Sub HScrollRed_Change()
    If valuecheck <> 1 Then
        Dim lab As CIELAB
        SampleColour.Picture = LoadPicture()
        LabelRed.Caption = "Red = " & HScrollRed.value
        lab = ConvertRGBtoLab(HScrollRed.value, _
          HScrollGreen.value, HScrollBlue.value)
        SampleColour.BackColor = rgb(HScrollRed.value, _
          HScrollGreen.value, HScrollBlue.value)
        valuecheck = 1
        HScrollCIEL.value = lab.l
        valuecheck = 1
        HScrollCIEa.value = lab.a
        valuecheck = 1
        HScrollCIEb.value = lab.b
        valuecheck = 0
        LabelCIEL.Caption = "CIE L = " & HScrollCIEL.value
```

```
      LabelCIEa.Caption = "CIE a = " & HScrollCIEa.value
      LabelCIEb.Caption = "CIE b = " & HScrollCIEb.value
      OK.Enabled = True
   Else
      valuecheck = 0
   End If

End Sub

Private Sub OK_Click()
   check = 1
   DefineValueCIE.Hide
End Sub
```

Module1.bas

```
Option Explicit
Public check As Integer
Public valuecheck As Integer
Public NumberOfSubmenu As Integer
Public Type CIELAB
   l As Single
   a As Single
   b As Single
End Type

Public Type HSV
   H As Single
   S As Single
   V As Single
End Type

' ------------------------
' Bitmap Array Information
' ------------------------


Public Type RGBTriplet
```

```
    rgbBlue As Byte
    rgbGreen As Byte
    rgbRed As Byte
    flag As Integer
End Type

' ------------------
' Bitmap Information
' ------------------
Public Type BITMAP
    bmType As Long
    bmWidth As Long
    bmHeight As Long
    bmWidthBytes As Long
    bmPlanes As Integer
    bmBitsPixel As Integer
    bmBits As Long
End Type
Public Declare Function GetBitmapBits Lib "gdi32" _
  (ByVal hBitmap As Long, ByVal dwCount As Long, _
  lpBits As Any) As Long
Public Declare Function SetBitmapBits Lib "gdi32" _
  (ByVal hBitmap As Long, ByVal dwCount As Long, _
  lpBits As Any) As Long
Public Declare Function GetObject Lib "gdi32" Alias _
  "GetObjectA" (ByVal hObject As Long, _
  ByVal nCount As Long, lpObject As Any) As Long

Public Enum bmphErrors
    bmphInvalidBitmapBits = vbObjectError + 1001
    bmphPaletteError
End Enum

' -------------------
' Palette Information
' -------------------
Private Type PALETTEENTRY
    peRed As Byte
    peGreen As Byte
    peBlue As Byte
    peFlags As Byte
End Type
Private Declare Function GetNearestPaletteIndex Lib "gdi32" _
  (ByVal hPalette As Long, ByVal crColor As Long) As Long
Private Declare Function GetPaletteEntries Lib "gdi32" _
  (ByVal hPalette As Long, ByVal wStartIndex As Long, _
  ByVal wNumEntries As Long, lpPaletteEntries As PALETTEENTRY) As Long
Private Declare Function RealizePalette Lib "gdi32" _
  (ByVal hdc As Long) As Long
Private Declare Function GetSystemPaletteEntries Lib _
  "gdi32" (ByVal hdc As Long, ByVal wStartIndex As Long, _
```

```
  ByVal wNumEntries As Long, lpPaletteEntries As PALETTEENTRY) As Long
Private Declare Function ResizePalette Lib "gdi32" _
  (ByVal hPalette As Long, ByVal nNumEntries As Long) As Long
Private Declare Function SetPaletteEntries Lib "gdi32" _
  (ByVal hPalette As Long, ByVal wStartIndex As Long, _
  ByVal wNumEntries As Long, lpPaletteEntries As PALETTEENTRY) As Long
Private Const MAX_PALETTE_SIZE = 256
Private Const PC_NOCOLLAPSE = &H4    ' Do not match color existing entries.

' -------------------------------
' System Capabilities Information
' -------------------------------
Private Declare Function GetDeviceCaps Lib "gdi32" _
  (ByVal hdc As Long, ByVal nIndex As Long) As Long
Private Const NUMRESERVED = 106  ' Number of reserved entries in system palette.
Private Const SIZEPALETTE = 104  ' Size of system palette.

' Copy memory quickly. Used for 24-bit images.
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" _
  (Destination As Any, Source As Any, ByVal Length As Long)


' Load the control's palette so it matches the
' system palette.
Private Sub MatchColorPalette(ByVal pic As PictureBox)
Dim log_hpal As Long
Dim sys_pal(0 To MAX_PALETTE_SIZE - 1) As PALETTEENTRY
Dim orig_pal(0 To MAX_PALETTE_SIZE - 1) As PALETTEENTRY
Dim i As Integer
Dim sys_pal_size As Long
Dim num_static_colors As Long
Dim static_color_1 As Long
Dim static_color_2 As Long

    ' Make sure pic has the foreground palette.
    pic.ZOrder
    RealizePalette pic.hdc
    DoEvents

    ' Get system palette size and # static colors.
    sys_pal_size = GetDeviceCaps(pic.hdc, SIZEPALETTE)
    num_static_colors = GetDeviceCaps(pic.hdc, NUMRESERVED)
    static_color_1 = num_static_colors \ 2 - 1
    static_color_2 = sys_pal_size - num_static_colors \ 2

    ' Get the system palette entries.
    GetSystemPaletteEntries pic.hdc, 0, _
        sys_pal_size, sys_pal(0)

    ' Make the logical palette as big as possible.
    log_hpal = pic.Picture.hpal
```

```
    If ResizePalette(log_hpal, sys_pal_size) = 0 Then
        Err.Raise bmphPaletteError, _
            "DDBHelper.MatchColorPalette", _
            "Error matching bitmap palette"
    End If

    ' Blank the non-static colors.
    For i = 0 To static_color_1
        orig_pal(i) = sys_pal(i)
    Next i
    For i = static_color_1 + 1 To static_color_2 - 1
        With orig_pal(i)
            .peRed = 0
            .peGreen = 0
            .peBlue = 0
            .peFlags = PC_NOCOLLAPSE
        End With
    Next i
    For i = static_color_2 To 255
        orig_pal(i) = sys_pal(i)
    Next i
    SetPaletteEntries log_hpal, 0, sys_pal_size, orig_pal(0)

    ' Insert the non-static colors.
    For i = static_color_1 + 1 To static_color_2 - 1
        orig_pal(i) = sys_pal(i)
        orig_pal(i).peFlags = PC_NOCOLLAPSE
    Next i
    SetPaletteEntries log_hpal, static_color_1 + 1, _
      static_color_2 - static_color_1 - 1, orig_pal(static_color_1 + 1)

    ' Realize the new palette.
    RealizePalette pic.hdc
End Sub

' Return a binary representation of the byte.
' This helper function is useful for understanding
' byte values.
Public Function BinaryByte(ByVal value As Byte) As String
Dim i As Integer
Dim txt As String

    For i = 1 To 8
        If value And 1 Then
            txt = "1" & txt
        Else
            txt = "0" & txt
        End If
        value = value \ 2
    Next i
```

```
    BinaryByte = txt
End Function

' Load the bits from this PictureBox into a
' two-dimensional array of RGB values. Set
' bits_per_pixel to be the number of bits per pixel.
Public Sub GetBitmapPixels(ByVal pic As PictureBox, _
  ByRef pixels() As RGBTriplet, ByRef bits_per_pixel As Integer)
' Uncomment the following to make the routine
' display information about the bitmap.
' #Const DEBUG_PRINT_BITMAP = True

Dim hbm As Long
Dim bm As BITMAP
Dim l As Single
Dim t As Single
Dim old_color As Long
Dim bytes() As Byte
Dim num_pal_entries As Long
Dim pal_entries(0 To MAX_PALETTE_SIZE - 1) As PALETTEENTRY
Dim pal_index As Integer
Dim wid As Integer
Dim hgt As Integer
Dim X As Integer
Dim Y As Integer
Dim two_bytes As Long

    ' Get the bitmap information.
    hbm = pic.Image
    GetObject hbm, Len(bm), bm
    bits_per_pixel = bm.bmBitsPixel

    ' If bits_per_pixel is 16, see if it's really
    ' 15 or 16 bits per pixel.
    If bits_per_pixel = 16 Then
        ' Make the upper left pixel white.
        l = pic.ScaleLeft
        t = pic.ScaleTop
        old_color = pic.Point(l, t)
        pic.PSet (l, t), vbWhite

        ' See what color was set.
        ReDim bytes(0 To 0, 0 To 0)
        GetBitmapBits hbm, 2, bytes(0, 0)
        If (bytes(0, 0) And &H80) = 0 Then
            ' It's really a 15-bit image.
            bits_per_pixel = 15
        End If

        ' Restore the pixel's original color.
        pic.PSet (l, t), old_color
```

```
End If

#If DEBUG_PRINT_BITMAP Then
    Debug.Print "*** BITMAP Data ***"
    Debug.Print "bmType       "; bm.bmType
    Debug.Print "bmWidth      "; bm.bmWidth
    Debug.Print "bmHeight     "; bm.bmHeight
    Debug.Print "bmWidthBytes "; bm.bmWidthBytes
    Debug.Print "bmPlanes     "; bm.bmPlanes
    Debug.Print "bmBitsPixel  "; bm.bmBitsPixel
    Debug.Print "BitsPerPixel "; bits_per_pixel
#End If

' Get the bits.
If (bits_per_pixel = 8) Or _
   (bits_per_pixel = 15) Or _
   (bits_per_pixel = 16) Or _
   (bits_per_pixel = 24) Or _
   (bits_per_pixel = 32) _
Then
    ' Get the bits.
    ReDim bytes(0 To bm.bmWidthBytes - 1, 0 To bm.bmHeight - 1)
    GetBitmapBits hbm, bm.bmWidthBytes * bm.bmHeight, bytes(0, 0)
Else
    ' We don't know how to read this format.
    Err.Raise bmphInvalidBitmapBits, _
        "DDBHelper.GetBitmapPixels", _
        "Invalid number of bits per pixel: " _
        & Format$(bits_per_pixel)
End If

' Create the pixels array.
wid = bm.bmWidth
hgt = bm.bmHeight
ReDim pixels(0 To wid - 1, 0 To hgt - 1)
Select Case bits_per_pixel
    Case 8
        ' Match pic's palette to the system palette.
        MatchColorPalette pic

        ' Get the image's palette entries.
        num_pal_entries = GetPaletteEntries( _
            pic.Picture.hpal, 0, _
            MAX_PALETTE_SIZE, pal_entries(0))

        ' Get the RGB color components.
        For Y = 0 To hgt - 1
            For X = 0 To wid - 1
                With pixels(X, Y)
                    pal_index = bytes(X, Y)
                    .rgbRed = pal_entries(pal_index).peRed
```

```
            .rgbGreen = pal_entries(pal_index).peGreen
            .rgbBlue = pal_entries(pal_index).peBlue
        End With
    Next X
Next Y

Case 15
    For Y = 0 To hgt - 1
        For X = 0 To wid - 1
            With pixels(X, Y)
                ' Get the combined 2 bytes for this pixel.
                two_bytes = bytes(X * 2, Y) + bytes(X * 2 + 1, Y) * 256&

                ' Separate the pixel's components.
                .rgbBlue = two_bytes Mod 32
                two_bytes = two_bytes \ 32
                .rgbGreen = two_bytes Mod 32
                two_bytes = two_bytes \ 32
                .rgbRed = two_bytes
            End With
        Next X
    Next Y

Case 16
    For Y = 0 To hgt - 1
        For X = 0 To wid - 1
            With pixels(X, Y)
                ' Get the combined 2 bytes for this pixel.
                two_bytes = bytes(X * 2, Y) + bytes(X * 2 + 1, Y) * 256&

                ' Separate the pixel's components.
                .rgbBlue = two_bytes Mod 32
                two_bytes = two_bytes \ 32
                .rgbGreen = two_bytes Mod 64
                two_bytes = two_bytes \ 64
                .rgbRed = two_bytes
            End With
        Next X
    Next Y

Case 24
    ' Blast the data from the pixels array
    ' to the bytes array using CopyMemory.
    For Y = 0 To hgt - 1
        CopyMemory pixels(0, Y), bytes(0, Y), wid * 3
    Next Y

Case 32
    For Y = 0 To hgt - 1
        For X = 0 To wid - 1
            With pixels(X, Y)
```

```
                .rgbBlue = bytes(X * 4, Y)
                .rgbGreen = bytes(X * 4 + 1, Y)
                .rgbRed = bytes(X * 4 + 2, Y)
            End With
        Next X
    Next Y

  End Select
End Sub
' Set the bits in this PictureBox using a 0-based
' two-dimensional array of RGBTriplets. The pixels must
' have the right dimensions to match the picture.
Public Sub SetBitmapPixels(ByVal pic As PictureBox, _
 ByVal bits_per_pixel As Integer, pixels() As RGBTriplet)
Dim wid_bytes As Long
Dim wid As Integer
Dim hgt As Integer
Dim X As Integer
Dim Y As Integer
Dim bytes() As Byte
Dim hpal As Long
Dim two_bytes As Long

  ' See how big the image must be.
  wid = UBound(pixels, 1) + 1
  hgt = UBound(pixels, 2) + 1

  ' See how many bytes per row we need.
  Select Case bits_per_pixel
    Case 8
      wid_bytes = wid
    Case 15, 16
      wid_bytes = wid * 2
    Case 24
      wid_bytes = wid * 3
    Case 32
      wid_bytes = wid * 4
    Case Else
      ' We don't understand this format.
      Err.Raise bmphInvalidBitmapBits, _
        "DDBHelper.GetBitmapPixels", _
        "Invalid number of bits per pixel: " _
        & Format$(bits_per_pixel)
  End Select

  ' Make sure it's even.
  If wid_bytes Mod 2 = 1 Then wid_bytes = wid_bytes + 1

  ' Create the bitmap bytes array.
  ReDim bytes(0 To wid_bytes - 1, 0 To hgt - 1)
```

```
' Set the bitmap byte values.
Select Case bits_per_pixel
    Case 8
        ' Use the nearest palette entries.
        hpal = pic.Picture.hpal

        ' Get the RGB color components.
        For Y = 0 To hgt - 1
            For X = 0 To wid - 1
                With pixels(X, Y)
                    bytes(X, Y) = (&HFF And _
                        GetNearestPaletteIndex(hpal, _
                            rgb(.rgbRed, .rgbGreen, .rgbBlue) _
                        + &H2000000))
                End With
            Next X
        Next Y

    Case 15
        For Y = 0 To hgt - 1
            For X = 0 To wid - 1
                With pixels(X, Y)
                    ' Keep the values in bounds.
                    If .rgbRed > &H1F Then .rgbRed = &H1F
                    If .rgbGreen > &H1F Then .rgbGreen = &H1F
                    If .rgbBlue > &H1F Then .rgbBlue = &H1F

                    ' Combine the values in 2 bytes.
                    two_bytes = .rgbBlue + 32 * (.rgbGreen + CLng(.rgbRed) * 32)

                    ' Set the byte values.
                    bytes(X * 2, Y) = (two_bytes Mod 256) And &HFF
                    bytes(X * 2 + 1, Y) = (two_bytes \ 256) And &HFF
                End With
            Next X
        Next Y

    Case 16
        For Y = 0 To hgt - 1
            For X = 0 To wid - 1
                With pixels(X, Y)
                    ' Keep the values in bounds.
                    If .rgbRed > &H1F Then .rgbRed = &H1F
                    If .rgbGreen > &H3F Then .rgbGreen = &H3F
                    If .rgbBlue > &H1F Then .rgbBlue = &H1F

                    ' Combine the values in 2 bytes.
                    two_bytes = .rgbBlue + 32 * (.rgbGreen + CLng(.rgbRed) * 64)

                    ' Set the byte values.
                    bytes(X * 2, Y) = (two_bytes Mod 256) And &HFF
```

```
                bytes(X * 2 + 1, Y) = (two_bytes \ 256) And &HFF

             End With
          Next X
       Next Y

    Case 24
       ' Blast the data from the bytes array
       ' to the pixels array using CopyMemory.
       For Y = 0 To hgt - 1
          CopyMemory bytes(0, Y), pixels(0, Y), wid * 3
       Next Y

    Case 32
       For Y = 0 To hgt - 1
          For X = 0 To wid - 1
             With pixels(X, Y)
                bytes(X * 4, Y) = .rgbBlue
                bytes(X * 4 + 1, Y) = .rgbGreen
                bytes(X * 4 + 2, Y) = .rgbRed
             End With
          Next X
       Next Y

 End Select

 ' Set the picture's bitmap bits.
 SetBitmapBits pic.Image, wid_bytes * hgt, _
    bytes(0, 0)
 pic.Refresh
End Sub

Function ConvertRGBtoLab(ByVal red As Integer, _
 ByVal green As Integer, ByVal blue As Integer) As CIELAB

Dim CIE As CIELAB
'Dim red As Integer
'Dim green As Integer
'Dim blue As Integer
Dim s_r As Single
Dim s_g As Single
Dim s_b As Single
Dim X As Single
Dim Y As Single
Dim Z As Single
Dim Xn As Single
Dim Yn As Single
Dim Zn As Single

s_r = red / 255
s_g = green / 255
```

```
s_b = blue / 255

If s_r <= 0.04045 Then
   s_r = s_r / 12.92
Else
   s_r = ((s_r + 0.055) / 1.055) ^ 2.4
End If
If s_g <= 0.04045 Then
   s_g = s_g / 12.92
Else
   s_g = ((s_g + 0.055) / 1.055) ^ 2.4
End If
If s_b <= 0.04045 Then
   s_b = s_b / 12.92
Else
   s_b = ((s_b + 0.055) / 1.055) ^ 2.4
End If

Xn = 0.4124 + 0.3576 + 0.1805
Yn = 0.2126 + 0.7152 + 0.0722
Zn = 0.0193 + 0.1192 + 0.9505

X = 0.4124 * s_r + 0.3576 * s_g + 0.1805 * s_b
Y = 0.2126 * s_r + 0.7152 * s_g + 0.0722 * s_b
Z = 0.0193 * s_r + 0.1192 * s_g + 0.9505 * s_b

If Y / Yn > 0.008856 Then
   ConvertRGBtoLab.l = 116 * (Y / Yn) ^ (1 / 3) - 16
Else
   ConvertRGBtoLab.l = 903.3 * (Y / Yn)
End If

ConvertRGBtoLab.a = 500 * (F(X / Xn) - F(Y / Yn))
ConvertRGBtoLab.b = 200 * (F(Y / Yn) - F(Z / Zn))

End Function

Function F(t As Single) As Single
If t > 0.008856 Then
   F = t ^ (1 / 3)
Else
   F = 7.787 * t + 16 / 116
End If
End Function

Function ConvertRGBtoHSV(ByVal red As Single, _
  ByVal green As Single, ByVal blue As Single) As HSV

Dim max As Single
Dim min As Single
Dim r As Single
```

```
Dim g As Single
Dim b As Single
Dim C_HSV As HSV

red = red / 255
green = green / 255
blue = blue / 255

If red <= 0.04045 Then
   red = red / 12.92
Else
   red = ((red + 0.055) / 1.055) ^ 2.4
End If
If green <= 0.04045 Then
   green = green / 12.92
Else
   green = ((green + 0.055) / 1.055) ^ 2.4
End If
If blue <= 0.04045 Then
   blue = blue / 12.92
Else
   blue = ((blue + 0.055) / 1.055) ^ 2.4
End If

max = IIf(red > green, IIf(red > blue, red, blue), IIf(green > blue, green, blue))
min = IIf(red < green, IIf(red < blue, red, blue), IIf(green < blue, green, blue))


If (max = min) Then
   ConvertRGBtoHSV.H = -1
Else
   ConvertRGBtoHSV.S = (max - min) / max
   ConvertRGBtoHSV.V = max
End If

If ConvertRGBtoHSV.S = 0 Then
   ConvertRGBtoHSV.H = -1
Else
   r = (max - red) / (max - min)
   g = (max - green) / (max - min)
   b = (max - blue) / (max - min)
   If (red = max) And (green = min) Then
      ConvertRGBtoHSV.H = 60 * (5 + b)
   ElseIf (red = max) And (blue = min) Then
      ConvertRGBtoHSV.H = (1 - g) * 60
   ElseIf (green = max) And (blue = min) Then
      ConvertRGBtoHSV.H = (r + 1) * 60
   ElseIf (green = max) And (red = min) Then
      ConvertRGBtoHSV.H = (3 - b) * 60
   ElseIf red = min Then
      ConvertRGBtoHSV.H = (3 + g) * 60
```

```
    Else
        ConvertRGBtoHSV.H = (5 - r) * 60
    End If
End If

End Function

Function ConvertLabtoRGB(ByVal CIEL As Single, _
  ByVal CIEa As Single, ByVal CIEb As Single) As RGBTriplet

Dim rgb As RGBTriplet
Dim X, Y, Z, Xn, Yn, Zn As Single
Dim s_r, s_g, s_b As Single

Xn = 0.4124 + 0.3576 + 0.1805
Yn = 0.2126 + 0.7152 + 0.0722
Zn = 0.0193 + 0.1192 + 0.9505

X = Xn * (((CIEL + 16) / 116 + CIEa / 500) ^ 3)
Y = Yn * (((CIEL + 16) / 116) ^ 3)
Z = Zn * (((CIEL + 16) / 116 - CIEb / 200) ^ 3)

s_r = 3.240479 * X - 1.53715 * Y - 0.498545 * Z
s_g = -0.969256 * X + 1.875992 * Y + 0.041556 * Z
s_b = 0.055648 * X - 0.204043 * Y + 1.057311 * Z

If s_r < 0.0031308 Then
    s_r = 12.92 * s_r
Else
    s_r = 1.055 * (s_r ^ (1 / 2.4)) - 0.055
End If

If s_g < 0.0031308 Then
    s_g = 12.92 * s_g
Else
    s_g = 1.055 * (s_g ^ (1 / 2.4)) - 0.055
End If

If s_b < 0.0031308 Then
    s_b = 12.92 * s_b
Else
    s_b = 1.055 * (s_b ^ (1 / 2.4)) - 0.055
End If

s_r = s_r * 256
s_g = s_g * 256
s_b = s_b * 256

If s_r < 0 Then
ConvertLabtoRGB.rgbRed = 0
ConvertLabtoRGB.flag = -1
```

```
ElseIf s_r > 255 Then
ConvertLabtoRGB.rgbRed = 255
ConvertLabtoRGB.flag = -1
Else
ConvertLabtoRGB.rgbRed = s_r
End If

If s_g < 0 Then
ConvertLabtoRGB.rgbGreen = 0
ConvertLabtoRGB.flag = -1
ElseIf s_g > 255 Then
ConvertLabtoRGB.rgbGreen = 255
ConvertLabtoRGB.flag = -1
Else
ConvertLabtoRGB.rgbGreen = s_g
End If

If s_b < 0 Then
ConvertLabtoRGB.rgbBlue = 0
ConvertLabtoRGB.flag = -1
ElseIf s_b > 255 Then
ConvertLabtoRGB.rgbBlue = 255
ConvertLabtoRGB.flag = -1
Else
ConvertLabtoRGB.rgbBlue = s_b
End If

End Function

Function ColourDifference(ByVal l1 As Single, ByVal a1 As Single, _
   ByVal b1 As Single, ByVal l2 As Single, ByVal a2 As Single, _
   ByVal b2 As Single) As Boolean

   Dim C_temp As Single
   Dim L_temp As Single
   Dim ratio As Single
   Dim c1, c2, h1, h2 As Single

   c1 = Sqr(a1 ^ 2 + b1 ^ 2)
   c2 = Sqr(a2 ^ 2 + b2 ^ 2)
   If a1 > 0 Then
         h1 = (Atn(b1 / a1)) * 180 / 3.1416
      Else
         h1 = (3.1416 + Atn(b1 / a1)) * 180 / 3.1416
   End If

   If a2 > 0 Then
         h2 = (Atn(b2 / a2)) * 180 / 3.1416
      Else
         h2 = (3.1416 + Atn(b2 / a2)) * 180 / 3.1416
   End If
```

```
If h1 > h2 Then
   If h1 - h2 > 9 Then
      ColourDifference = True
      Exit Function
   ElseIf (l1 < l2) And (c1 > c2) Then
      ColourDifference = True
      Exit Function
   ElseIf (l1 < l2) And (c1 < c2) Then
      C_temp = 0.72 * (h1 - h2) * (c1 - 5)
      If C_temp > c2 Then
         ColourDifference = True
         Exit Function
      Else
         ratio = (c2 - C_temp) / (l2 - l1)
         If ratio > 2 / 3 Then
            ColourDifference = False
            Exit Function
         Else
            ColourDifference = True
            Exit Function
         End If
      End If
   ElseIf (l1 > l2) And (c1 > c2) Then
      L_temp = l1 - (h1 - h2) * (96 - l1) / 2
      If L_temp < l2 Then
         ColourDifference = True
         Exit Function
      Else
         ratio = (c1 - c2) / (L_temp - l2)
         If ratio > 2 / 3 Then
            ColourDifference = True
            Exit Function
         Else
            ColourDifference = False
            Exit Function
         End If
      End If
   ElseIf (l1 > l2) And (c1 < c2) Then
      C_temp = 0.72 * (h1 - h2) * (c1 - 5)
      If C_temp < c2 Then
         ColourDifference = False
         Exit Function
      Else
         ratio = (C_temp - c2) / (l1 - l2)
         If ratio > 2 / 3 Then
            ColourDifference = True
            Exit Function
         Else
            ColourDifference = False
```

```
            Exit Function
        End If
    End If
End If
Else
    If h2 - h1 > 9 Then
        ColourDifference = False
        Exit Function
    ElseIf (l2 < l1) And (c2 > c1) Then
        ColourDifference = False
        Exit Function
    ElseIf (l2 < l1) And (c2 < c1) Then
        C_temp = 0.72 * (h2 - h1) * (c2 - 5)
        If C_temp > c1 Then
            ColourDifference = False
            Exit Function
        Else
            ratio = (c1 - C_temp) / (l1 - l2)
            If ratio > 2 / 3 Then
                ColourDifference = True
                Exit Function
            Else
                ColourDifference = False
                Exit Function
            End If
        End If
    ElseIf (l2 > l1) And (c2 > c1) Then
        L_temp = l2 - (h2 - h1) * (96 - l2) / 2
        If L_temp < l1 Then
            ColourDifference = False
            Exit Function
        Else
            ratio = (c2 - c1) / (L_temp - l1)
            If ratio > 2 / 3 Then
                ColourDifference = False
                Exit Function
            Else
                ColourDifference = True
                Exit Function
            End If
        End If
    ElseIf (l2 > l1) And (c2 < c1) Then
        C_temp = 0.72 * (h2 - h1) * (c2 - 5)
        If C_temp < c1 Then
            ColourDifference = True
            Exit Function
        Else
            ratio = (C_temp - c1) / (l2 - l1)
            If ratio > 2 / 3 Then
                ColourDifference = False
                Exit Function
```

```
            Else
               ColourDifference = True
               Exit Function
            End If
         End If
      End If

   End If

End Function
```