

**CALIBRATED COLOUR MAPPING BETWEEN LCD AND CRT
DISPLAYS: A CASE STUDY**

by

William S. Cressman
BS Mechanical Engineering, Rice University, 1989
MBA, University of California, Irvine, 2000

PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

In the School
of
Computing Science

© William S. Cressman 2004

SIMON FRASER UNIVERSITY

January 2004

All rights reserved. This work may not be
reproduced in whole or in part, by photocopy
or other means, without permission of the author.

APPROVAL

Name: William S. Cressman

Degree: Master of Science, Computing Science

Title of Project: Calibrated Colour Mapping Between LCD and CRT Displays: A Case Study

Examining Committee:

Chair: **Dr. Richard Vaughan**
Assistant Professor, School of Computing Science

Dr. Brian Funt
Professor, School of Computing Science

Dr. Mark Drew
Associate Professor, School of Computing Science

Dr. Tim Lee
Adjunct Professor
School of Computing Science
Simon Fraser University

Date Approved:

1/23/04

SIMON FRASER UNIVERSITY

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project and extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/**Project**/Extended Essay

**Calibrated Colour Mapping Between LCD and
CRT Displays: A Case Study**

Author:

(signature)

William Cressman

(name)

JANUARY 23, 2004
(date)

ABSTRACT

The primary goal of a colour characterization model is to establish a mapping from digital input values d_i ($i=R,G,B$) to tristimulus values such as XYZ. A good characterization model should be fast, use a small amount of data, and allow for backward mapping from tristimulus to d_i .

This paper demonstrates implementations of three characterization models tested on seven different display devices. The characterization models implemented in this study are a 3D Look Up Table (LUT), a linear model [2],[4], and the masking model Tamura et al. in 2002 [6]. The devices include two CRT Monitors, three LCD Monitors, and two LCD Projectors.

Several characteristics of the display devices are presented with respect to data collection and characterization modelling. These include the surprisingly long phosphor stabilization time on CRT monitors, which has implications for data collection; and the shifting chromaticity of mixed colours on LCD displays which adversely affects the masking model.

The results of this study indicate that a simple linear model is the most effective and efficient for all devices used in the study, despite the common belief that it is sometimes inappropriate for LCD monitors [6]. A simple extension to the linear model is presented, and it is demonstrated that this extension improves white prediction without causing significant errors for other colours.

ACKNOWLEDGEMENTS

I would like to thank Brian Funt for his guidance in this project, Florian Curiea for his invaluable help with Matlab, and Binay Bhattacharya for his help with some tricky geometric logic critical to the gamut mapping algorithms. Most of all I would like to thank Behnam Bastani for his constant help in every area of the project, and my wife Zhanar for her patience throughout this experience.

TABLE OF CONTENTS



| | |
|---|------|
| Approval | ii |
| Abstract | iii |
| Acknowledgements | iv |
| Table of Contents | v |
| List of Figures..... | vi |
| List of Tables | viii |
| List of Abbreviations and Acronyms | ix |
| Introduction..... | 1 |
| Data Collection | 3 |
| Device characteristics | 7 |
| Implementation Details..... | 11 |
| 3D LUT Model..... | 11 |
| Linear Model | 12 |
| Masking Model..... | 15 |
| Results | 18 |
| Conclusion..... | 23 |
| Contribution Summary..... | 25 |
| Appendix A: The SpectraCOM Software | 26 |
| Getting Started..... | 27 |
| Installing the SpectraCOM DSN | 28 |
| Using the SpectraCOM Interface..... | 29 |
| The SpectraCOM Interface Window | 30 |
| The SpectraCOM Chart Window | 32 |
| The SpectraCOM Gamut Measurement Window | 32 |
| Scripting with SpectraCOM..... | 34 |
| The Database | 38 |
| Accessing the Data..... | 39 |
| Troubleshooting SpectraCOM | 39 |
| Appendix B: The Gamut Viewer Tool | 41 |
| Appendix C: The MS PowerPoint® Sample Interface | 43 |
| Appendix D: A Note on Gamut Mapping..... | 44 |
| Diagonal Transform Scaling | 45 |
| Clipping | 45 |
| Scaling and Rotation..... | 46 |
| Warping..... | 46 |
| Bibliography..... | 48 |

LIST OF FIGURES

| | |
|---|----|
| Figure 1: Percentage of the steady state luminosity for white on the vertical-axis vs. the number of seconds since black was displayed on the horizontal-axis. | 4 |
| Figure 2: Measurement Error (Log scale) vs Integration Time in milliseconds measured on four grayscale colours on CRT1..... | 5 |
| Figure 3: Channel Interaction Interaction. The horizontal axis represents the input value v ranging from 0 to 255 and the vertical axis represents the value of the Channel Interaction metric. $CI_{\text{COLOR}}(v,a,b)$. The black line shows $a=b=255$ and dashed lines show $a=0,b=255$ and $a=255,b=0$ | 8 |
| Figure 4: X response shape differences between channels for PR1..... | 9 |
| Figure 5: Chromaticity Shift Diagrams in xy space, with $x=X/(X+Y+Z)$ on the horizontal axis and $y=Y/(X+Y+Z)$ on the vertical axis..... | 10 |
| Figure 6: X-Channel Response Curves for PR1, with a close-up of the non-monotonic region and the smoothed results..... | 13 |
| Figure 7: Mapping Error vs Training Data Size..... | 14 |
| Figure 8: Backward Error distribution for each characterization model on each device. ΔE error value is shown on the horizontal axis and histogram counts are shown on the vertical axis..... | 19 |
| Figure 9: Backward Error vs Chromaticity. The horizontal and vertical axes are $X/(X+Y+Z)$ and $Y/(X+Y+Z)$ respectively..... | 20 |
| Figure 10: Linearization Failure for the Black Channel on PR1..... | 22 |
| Figure 11: The SpectraCOM Interface..... | 26 |
| Figure 12: Updating the ODBC DSN..... | 28 |
| Figure 13: SpectraCOM System Tray Right-Click Menu..... | 30 |
| Figure 14: The Setup Tab on the SpectraCOM Interface..... | 31 |
| Figure 15: The Gamut Measurement Tool and Start Test Window..... | 34 |
| Figure 16: The Database Schema..... | 38 |
| Figure 17: The GamutViewer Interface..... | 41 |
| Figure 18: Starting GamutViewer by selecting two files and dropping them on the executable..... | 42 |
| Figure 19: Example of swapping the gray file and showing points as spheres..... | 42 |
| Figure 20: The clmageBuddy object in the project toolbox..... | 43 |
| Figure 21: Sample output from the Gamut Visualization application developed for this project. (a) two gamuts, with one in gray, (b) one gamut with points shown as spheres..... | 44 |

| | |
|--|----|
| Figure 22: Example of a clipping operation that results in another OOG point..... | 46 |
| Figure 23: Example of gray axis curvature after warping without rotation or scaling..... | 47 |

LIST OF TABLES

| | | |
|----------|--|----|
| Table 1: | Device Summary | 7 |
| Table 2: | Forward Error ΔE Mean (μ), standard deviation (σ) and maximum | 18 |
| Table 3: | Backward Error ΔE Mean (μ), standard deviation (σ) and maximum..... | 18 |
| Table 4: | Percent Increase in Forward ΔE Error Due to Monotonicity Correction..... | 21 |
| Table 5: | Experimental Running Time and Storage Space as multiples of linear model Usage | 22 |
| Table 6: | SpectraCOM Object Properties  | 36 |
| Table 7: | SpectraCOM Object Methods  | 37 |

LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|----------------------|---|
| ΔE | Delta E is a measurement of distance in CIELAB ₉₄ space. A value of one indicates a just noticeable difference in colour. |
| CRT | Cathode Ray Tube, a traditional type of display monitor which works by scanning a cathode ray from the back of the monitor across phosphors on the back side of the display screen |
| CMYK | Cyan, Magenta, Yellow, Black. In some cases, K is used to represent the gray axis. |
| CIELAB ₉₄ | International Commission On Illumination L a* b* colour space. This is a perceptually uniform colour space, where a unit of distance anywhere in the space represents the same amount of perceptual difference. |
| GOG | Gain-Offset-Gamma, a term for a linear characterization model that assumes that the shape of the output curves is exponential, or “gamma” shaped. |
| LCD | Liquid Crystal Display. A digital flat-panel monitor. |
| LUT | Look Up Table |
| OOG | Out of Gamut. This term refers to a colour that can be produced on one device, but is outside the gamut of the target device. |
| PCA | Principal Component Analysis |
| RGB | Red, Green and Blue. These are the colours of the phosphors, or primaries, in both LCD and CRT displays |
| UCR | Under-colour removal. A technique used in printing, in which an appropriate amount of black ink is used to replace overlapping quantities of magenta, cyan, and yellow in dark areas. |
| XYZ | Used to refer to the CIE XYZ tristimulus space, where X and Z represent chroma and Y represents luminance. |

INTRODUCTION

Accurate colour management across multiple displays is an important problem, and will become more important in years to come. Users are increasingly relying on digital displays for creating, viewing and presenting colour media. Users with multi-panel displays would like to see colour consistency across the displays, while conference speakers would like an accurate prediction of what their slides will look like before they enter the auditorium.

The act of predicting colours across multiple display devices requires implementation of several concepts, including device characterization, gamut mapping, and perceptual models. This paper is focused on the concept of device characterization – establishing a mapping from digital input values d_i ($i=R,G,B$) to tristimulus values such as XYZ. A good characterization model should be fast, use a small amount of data, and allow for *backward* mapping from tristimulus to d_i .

There are a several well-known characterization models that support both forward and backward mapping, three of which were implemented in this experiment: 3D Lookup Table (LUT), linear model and masking model. The 3D LUT method uses a pair of three-dimensional tables to associate a tristimulus triplet with every RGB combination, and vice versa. This method is simple to understand, but difficult and cumbersome to implement.

The term *linear model* refers to the group of models (GOG, S-Curve, and Polynomial model) that estimate tristimulus response with a linear combination of pure phosphor output. These models each start by linearizing the digital input response curves with the specific nonlinear function from which they draw their names. The linear

model has been widely used for CRT monitors, but has been criticized for its assumption of channel independence, which may not apply on LCD displays.

The third model implemented in this study was the masking model introduced by Tamura et al. in 2002 [6]. This model applies the concept of Under Colour Removal (UCR) to mask inputs from 3-dimensional RGB space to 7-dimensional RGBCMYK space, then linearizes inputs and combines outputs as was done in the linear model.

This paper will discuss the implementation, benefits, and pitfalls of each method with respect to use on CRT and LCD display devices. In general, prediction errors will be quantified terms of ΔE , as measured in 1994 CIE $L^*a^*b^*$ colour space.

The first section of the paper deals with data collection. The next section reviews the characteristics of devices used in the study. Section 3 discusses implementation details and considerations for each of the three characterization models, and section 4 reviews the results of the study.

DATA COLLECTION

All data used in this study was collected using a Photo Research SpectraScan 650 Spectrometer in a dark room with the spectrometer at a fixed distance, perpendicular to the center of the display surface. Before beginning each test, the monitor settings were reset to the factory default and the brightness was adjusted using a gray-scale calibration pattern until all shades of gray were visible.

The data collection was performed automatically in large randomized test suites. We found that measurement error was one of the largest contributors to characterization model failure, and as a result took extreme care in data gathering. It is important to test the repeatability of the spectrometer with respect to each monitor and ensure that the test plan is sufficient to smooth out any significant measurement errors. Each RGB sample used in this study was composed of a total of 25 measurements, taken in 5 randomly scheduled bursts of 5 measurements each.

Figure 1 shows the percentage of steady state luminosity for white vs. the number of seconds since a colour change from black. In this paper, luminosity will be defined as the L value in CIELAB₉₄ space. Note that the LCD-based devices often reach steady state within less than the first second, while the CRT devices take longer. The amount of time required for the CRT devices was somewhat surprising – up to 10 seconds in CRT2. The spike that occurs on CRT2 right after the colour change is unexpected as well. However, the implication for testing is straightforward - measurement delay after a colour change must be several seconds longer for CRT devices.

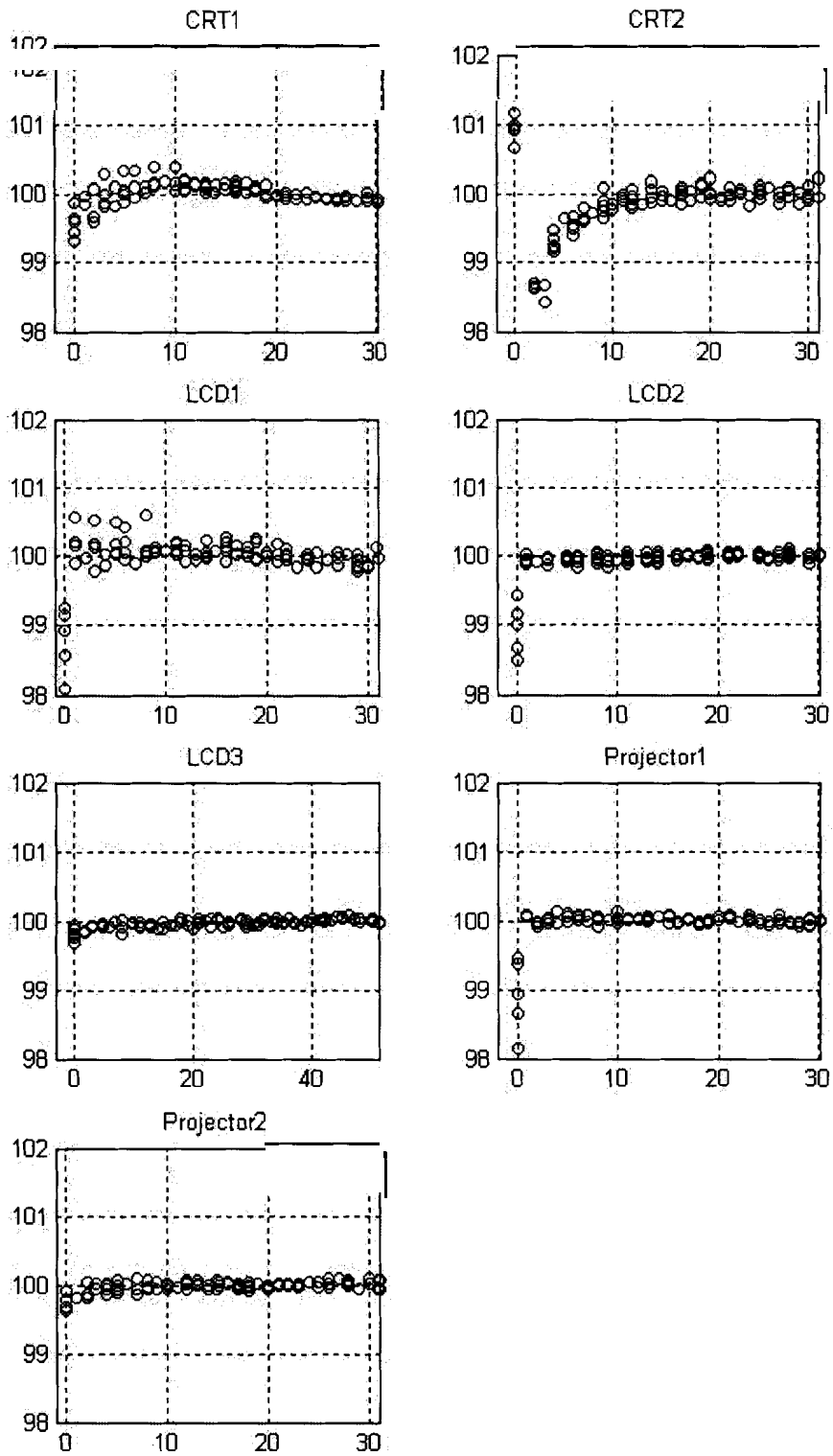


Figure 1: Percentage of the steady state luminosity for white on the vertical-axis vs. the number of seconds since black was displayed on the horizontal-axis.

Another important setting related to data consistency is spectrometer integration time. In general, CRT monitors require a longer integration time because the display flashes with each beam scan. Figure 2 shows the result of an integration time test on CRT1.

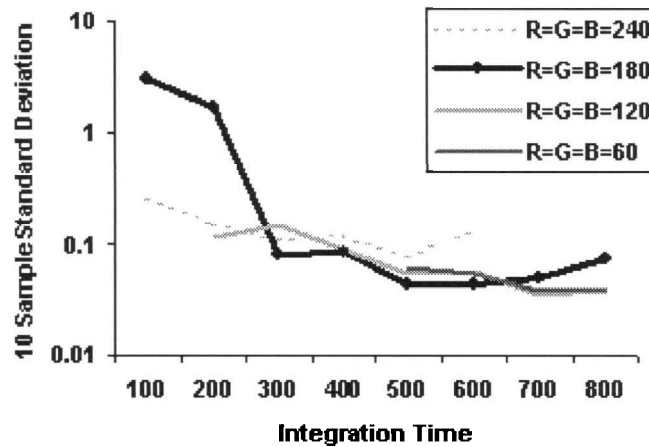


Figure 2: Measurement Error (Log scale) vs Integration Time in milliseconds measured on four grayscale colours on CRT1

Observe that shorter lower integration times result in more unstable measurements. The monitor refresh rate used in this experiment is 75 Hz, which equates to 13.3 ms per scan. Therefore, any integration time t will experience either $\lfloor t/13.3 \rfloor$ or $\lceil t/13.3 \rceil$ scans depending on when the measurement window starts. For example, if the integration time is 100ms, then measurements will either experience seven or eight scans, leading to high variation. Conversely, a time of 400 ms will almost always lead to 30 scans ($400 / 13.33 = 30.00$).

The measurements in this study were taken with a default integration time of 400 ms, which was doubled whenever a “low light” error was detected and halved when a “too much light” error was detected. Although this technique resulted in acceptable error

levels, an improvement would be to use only integration times are exact multiples of 13.3.

Three suites of data were collected for each monitor: a 10x10x10 grid of evenly spaced RGB values covering the entire 3D space, a similar 8x8x8 grid used for testing and verification, and a “101x7” data set made up of 101 evenly spaced measurements for each primary RGB and secondary CMYK channel with the other inputs set to zero.

DEVICE CHARACTERISTICS

Seven devices were tested – two CRT monitors, three LCD monitors, and two LCD projectors. A summary of these devices is given in Table 1.

| Name | Description |
|------|------------------------------------|
| CRT1 | Samsung Syncmaster 900NF |
| CRT2 | NEC Accusync 95F |
| LCD1 | IBM 9495 |
| LCD2 | NEC 1700V |
| LCD3 | Samsung 171N |
| PR1 | Proxima LCD Desktop Projector 9250 |
| PR2 | Proxima LCD Ultralight LX |

Table 1: Device Summary

A common issue in device characterization is channel interaction [5]. In this study, channel interaction is calculated as follows, where v represents the input value for the channel in question, a and b are constant values for the other two channels, and $L(r,g,b)$ represents the measured luminosity for a given digital input.

$$CI_{RED}(v,a,b) = \frac{(L(v,a,b) - L(0,a,b)) - (L(v,0,0) - L(0,0,0))}{L(255,255,255) - L(0,0,0)} \quad (1)$$

This equation returns zero when there is no channel interaction. The equations for CI_{GREEN} and CI_{BLUE} are similar.

It is commonly expected that LCD devices will exhibit channel interaction and CRT devices will not. However, the two CRT monitors exhibited more significant interaction problems than three of the five LCD devices, as shown in Figure 3. The nature of the interaction is surprising as well.

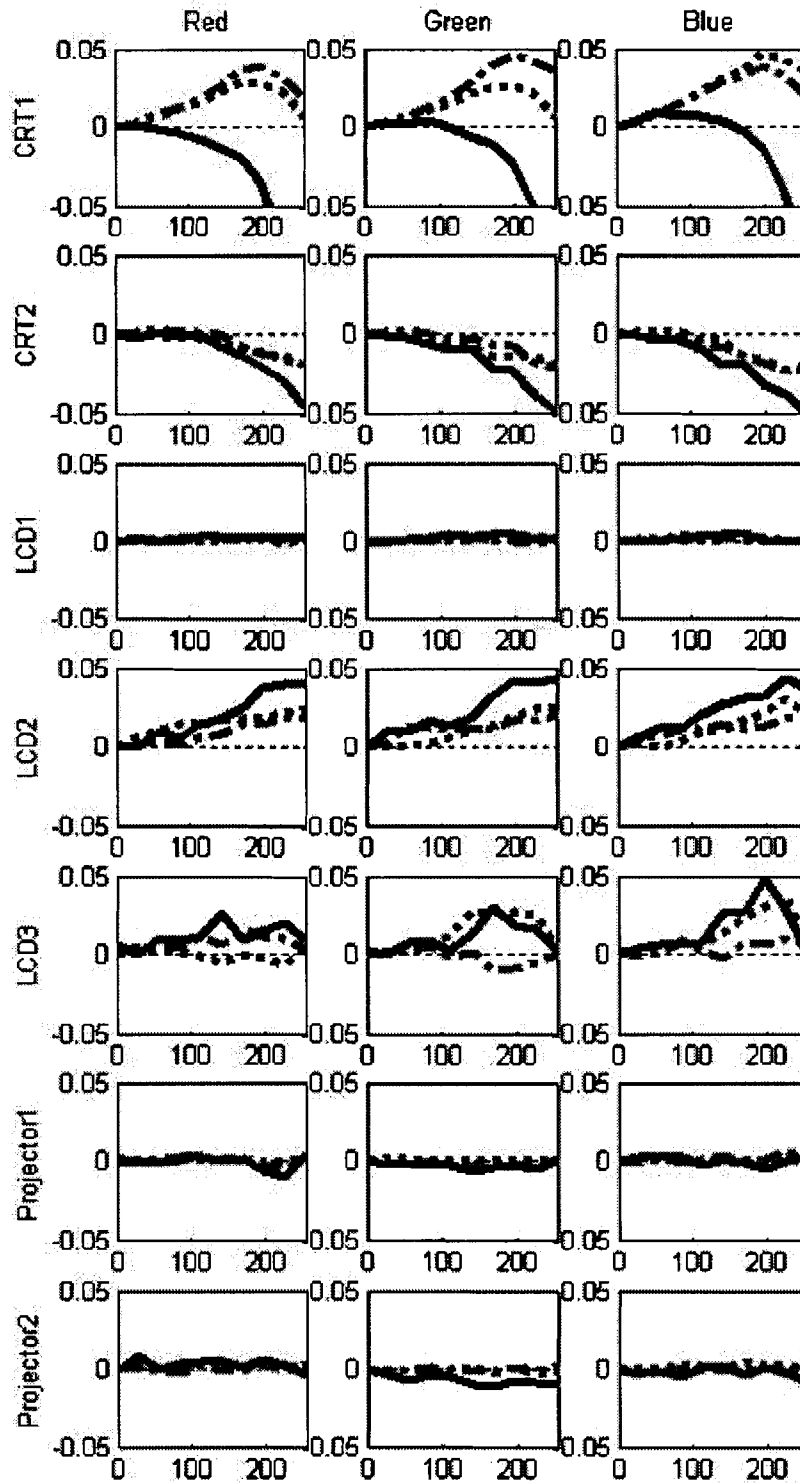


Figure 3: Channel Interaction Interaction. The horizontal axis represents the input value v ranging from 0 to 255 and the vertical axis represents the value of the Channel Interaction metric. $CI_{COLOR}(v,a,b)$. The black line shows $a=b=255$ and dashed lines show $a=0, b=255$ and $a=255, b=0$

Observe that for CRT1, interactions with one other phosphor tend to increase luminosity output while interactions with both other phosphors tend to decrease luminosity output. Interactions on other devices were either consistently additive or subtractive.

Another potential issue with LCD monitors is chromaticity shift [6]. This study found that chromaticity shift of pure phosphor colours was insignificant. However, chromaticity shift of combined colours (CMYK) was notable on all LCD devices as shown in Figure 5 on page 10. This effect is caused by the dissimilarity of shape between the strongly s-shaped B response curves and the more gamma-shaped R and G curves. An example of this shape difference is given in Figure 4.

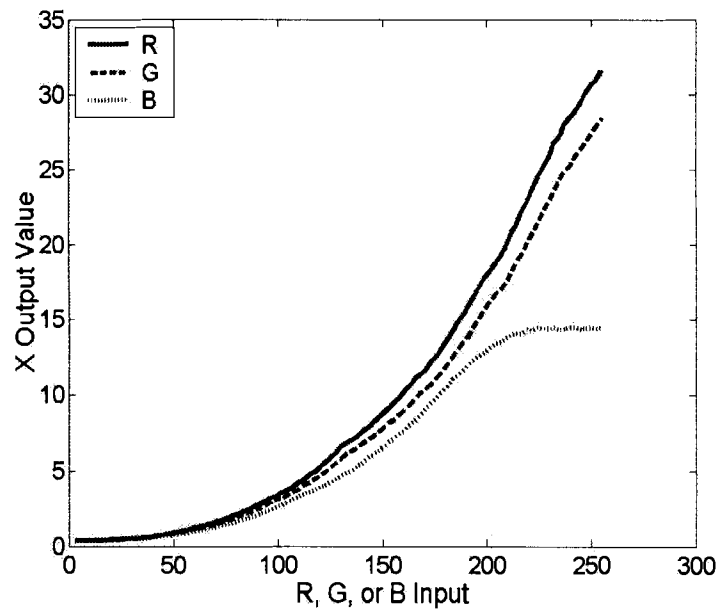


Figure 4: X response shape differences between channels for PR1

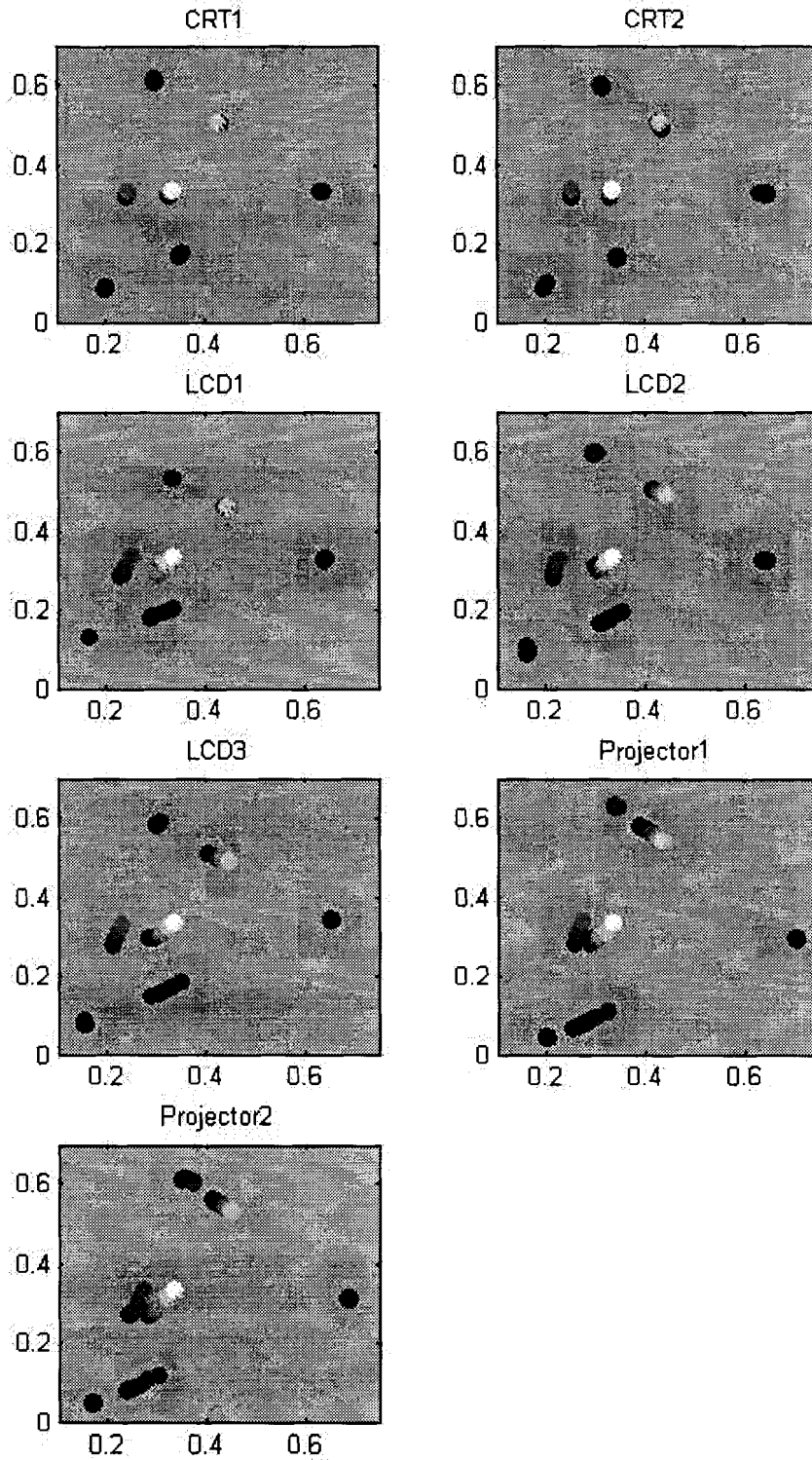


Figure 5: Chromaticity Shift Diagrams in xy space, with $x=X/(X+Y+Z)$ on the horizontal axis and $y=Y/(X+Y+Z)$ on the vertical axis.

IMPLEMENTATION DETAILS

All characterization methods start with black level flare correction, in which the measured XYZ value of black for the device is subtracted from the measured tristimulus value of each colour. This ensures that all devices have a common black point of (0,0,0) in XYZ space [2]. The remaining steps for each characterization are described below.

3D LUT Model

The 3D LUT method was implemented with the intention of providing a golden standard against which to evaluate the other two models, but is expensive both computationally and storage-wise (10 MB for a storage table) and is not well suited for reverse mapping. To create the forward lookup table, the 10x10x10 training data was interpolated using 3D linear interpolation to fill a 52x52x52 lookup table indexed by RGB values spaced 5 units apart. At look-up time, 3D spline interpolation is used to look up intermediate values.

Inverting the lookup to index by XYZ is non trivial – it requires interpolation of a sparse 3D data set; a task that is a field of research in its own right [1]. The reverse lookup was performed via tetrahedral interpolation on the original 10x10x10 data set. Tetrahedral Interpolation was chosen over a number of other methods primarily for its speed and ability to handle sparse, irregularly spaced data. However, any values that fall outside the convex hull of the measured gamut will return errors. This is particularly problematic for the LCD monitors, which have slightly convex gamut faces. In order to prevent edge values from returning invalid data, the entire lookup table was expanded outward by 1% from the gamut centroid.

Linear Model

The linear model is a two-stage characterization process. In the first step, the raw inputs d_i ($i=1, 2, 3$ for R, G, B) are linearized using a fitted function $C_i(d_i)$ for each channel. Linear regression is then used to determine the slope M_{ij} between each linearized input $C_i(d_i)$ and the respective XYZ outputs where $j=(1, 2, 3)$ for (X, Y, Z). The second stage applies matrix M to calculate estimated XYZ values.

$$\begin{bmatrix} X_{est} \\ Y_{est} \\ Z_{est} \end{bmatrix} = M \begin{bmatrix} C_1(d_1) \\ C_2(d_2) \\ C_3(d_3) \end{bmatrix} \quad (2)$$

The linearization functions in this implementation avoid any shape predisposition by using a LUT that is calculated as follows. The 10 measured response values for each input channel i are interpolated to obtain three output vectors $X(d_i)$, $Y(d_i)$ and $Z(d_i)$ in 256-dimensional space. Principal component analysis is then used to find the single vector $C_i(d_i)$ that best approximates all three output vectors. The following equation calculates $C_i(d_i)$ where PCA_i represents the weighting vector obtained from principal component analysis.

$$C_i(d_i) = [X(d_i) \quad Y(d_i) \quad Z(d_i)] * [PCA_i] \quad (3)$$

In order to allow for backward mapping, two conditions are required: the linearization function must be monotonic and the matrix M must be invertible. Inversion is always possible because none of the input channels are linearly dependent. However, the monotonicity requirement is a real risk with LCD displays, where the response curves sometimes level out or even decline for high input values (Figure 6). It is therefore necessary to modify the linearization function to ensure monotonicity. Note

that this modification, although necessary, serves to reduce the accuracy of the linearization and increases the overall error of the characterization.

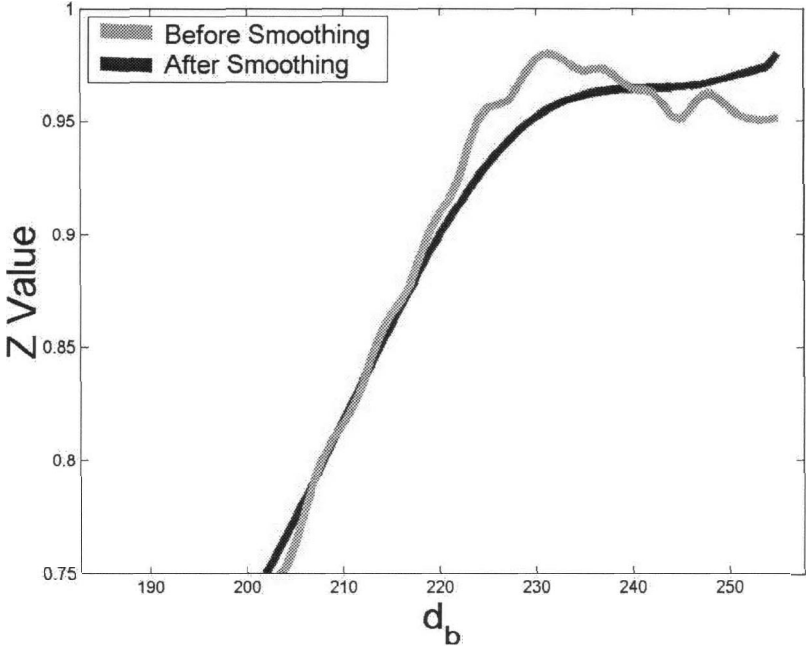


Figure 6: X-Channel Response Curves for PR1, with a close-up of the non-monotonic region and the smoothed results

When creating the lookup table, a decision must be made regarding the size of the training data set. Figure 7 shows the relationship between training data size and forward mapping error, measured in ΔE . In general, a larger training set is better, but the benefit tapers off after about 10 data points. For the results section of this paper, a training data set with 101 points was used to ensure minimal error introduced by training data size.

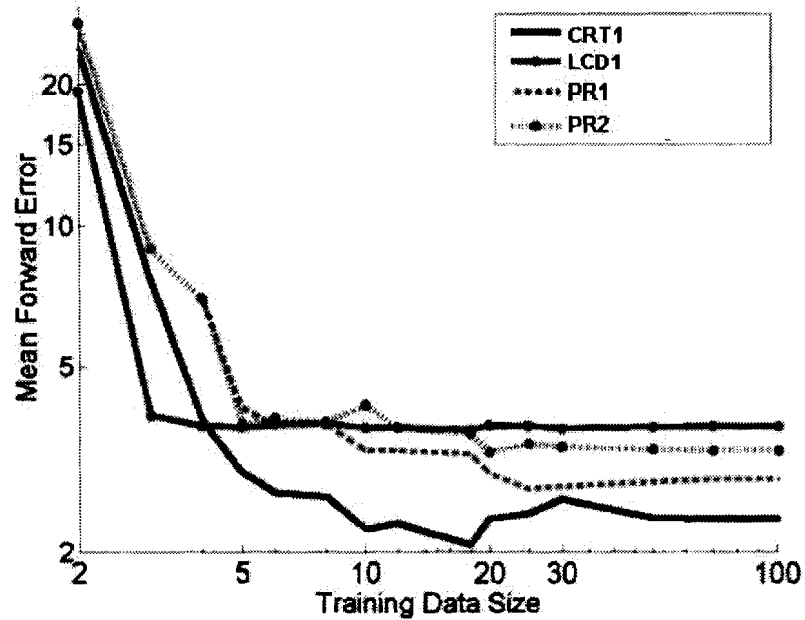


Figure 7: Mapping Error vs Training Data Size

The primary criticism of the linear model is that it assumes channel independence. As we have seen above, this is not always a valid assumption – even for CRT monitors. When there is channel interaction, the predicted output for colours that use more than one phosphor may not be accurate. This is especially true for white, which uses the maximum value of all three phosphors. Our observations suggest that this problem is not very noticeable on natural images where the eye is accustomed to correcting for scene lighting. However it becomes significant on computer-generated images such as presentation slides or charts where there are large regions of pure white with no expected ambient lighting. In this case, the eye is less forgiving.

One solution is to perform a white-point correction to ensure that the predicted white exactly matches the measured white. A simple approach is to apply a diagonal transform to the slope matrix M based on the measured and predicted values of pure white. The following formula shows the conversion, where X_{MEASURED} is the measured X

value for white and $X_{PREDICTED}$ is the predicted X value for white using the original slope matrix.

$$M_{LINEAR+} = M * \begin{bmatrix} \frac{X_{MEASURED}}{X_{PREDICTED}} & 0 & 0 \\ 0 & \frac{Y_{MEASURED}}{Y_{PREDICTED}} & 0 \\ 0 & 0 & \frac{Z_{MEASURED}}{Z_{PREDICTED}} \end{bmatrix} \quad (4)$$

This modification to the slope matrix ensures that predicted white is correct, but slightly shifts all of the other colours in a non-uniform manner, which could potentially increase the overall error. This model will be referred to as “Linear+” in this paper, and is useful when displaying computer-generated images where white is a major colour. Note that a similar correction can be performed using an alternate tristimulus space, such as LMS. In our study, we found that using either XYZ or LMS intermediate space returns the same average increase in forward error ($\pm 0.05 \Delta E$).

Further improvement may be possible using a technique similar to that presented by Finlayson and Drew in [3], where a modified least-squares procedure is used to determine the matrix M. By constraining the prediction error for white to zero, a matrix can be selected that reduces overall error while ensuring an accurate white value. It is interesting to note that their approach achieved good results even without first linearizing the inputs.

Masking Model

The masking model [6] attempts to avoid problems related to channel interaction with a technique similar to UCR in printing. The original digital input d_i is converted to masked input m_i ($i=1,2,3,4,5,6,7$ for RGBCMYK), and the masked values are combined

in a manner similar to what was done in the linear model. The masking operation assigns values to three elements of m – the primary colour (index p), the secondary colour (index s), and the gray colour (index 7), and sets all of the remaining elements of m to zero, as follows.

$$\begin{aligned}
&\text{Primary index } p \text{ such that } d_p = \max(d_1, d_2, d_3) \\
&\text{Gray index } k \text{ such that } d_k = \min(d_1, d_2, d_3) \ \& \ k \neq p \\
&\text{Secondary color index } s = k + 3 \\
&\text{Primary color value } m_p = d_p \\
&\text{Secondary color value } m_s = d_{6-p-k} \\
&\text{Gray (Under) color value } m_7 = d_k \\
&\text{Unused color values } m_q = 0 : q \notin \{p, s, 7\}
\end{aligned} \tag{5}$$

The result of these formulas is to set p to the index of the maximum primary colour (R, G, or B), and m_p to the input value for that colour. It assigns s to the index of the mixed colour (C, M, or Y) that does not contain the minimum colour, and assigns m_s to the median of the original values. Finally, it sets the gray value m_7 to the minimum of the three original inputs. For example, if the original inputs are RGB=(200,100,50), the primary colour will be red, with a value of 200. The secondary colour will be yellow (which does not contain blue) with a value of 100, and the gray (under) colour will have a value of 50. The masked input array becomes $m=[200,0,0,100,0,0,50]$.

Once the inputs have been converted into masked values m_i , a linearization function $C_i(m_i)$ for each input channel i is determined using the method described above for the linear model. The slope matrix M_{ij} for each input channel i and output channel j is calculated as using PCA and linear regression, also as described for the linear model. Finally, let the vector P_i represent the column of matrix M that contains the X, Y, and Z slopes for input channel i . The transformation from masked input to XYZ output can then be written as follows:

$$XYZ_{est} = [P_p \quad P_s \quad P_7] * \begin{bmatrix} C_p(m_p) - C_p(m_s) \\ C_s(m_s) - C_7(m_7) \\ C_7(m_7) \end{bmatrix} \quad (6)$$

The inverse mapping from XYZ to RGB is less obvious, and requires knowledge of the primary and secondary colour indices p and s . There is no way to know these values, so all six possible (p, s) combinations are tested (RM, RY, GC, GY, BC, BM) and any combination that satisfies the following conditions will yield the correct result.

$$255 \geq m_p \geq m_s \geq m_7 \geq 0 \quad (7)$$

RESULTS

This study calculated values of forward error ΔE_{FWD} , round trip error ΔE_{TRIP} , and backward error ΔE_{BWD} for 512 colours in an 8x8x8 evenly spaced grid of RGB inputs. For each colour, we find three vertices in CIE L*a*b* space: the measured value for the colour v_M , the predicted value v_P , and a round-trip value v_{RT} found by mapping backward and forward again from v_P . These points form a triangle with edges representing the forward, round-trip and backward error vectors. ΔE_{FWD} is the distance from v_M to v_P , ΔE_{TRIP} is the distance from v_P to v_{RT} , and ΔE_{BWD} is the distance from v_{RT} back to v_M .

| | LUT | | | Lin | | | Lin+ | | | Mask | | |
|---------|-------|----------|-----|-------|----------|-----|-------|----------|-----|-------|----------|------|
| | μ | σ | max | μ | σ | max | μ | σ | max | μ | σ | max |
| CRT1 | 0.8 | 0.3 | 2.4 | 2.4 | 1.2 | 5.4 | 2.2 | 1.3 | 5.2 | 2.6 | 1.3 | 7.1 |
| CRT2 | 0.5 | 0.3 | 2.3 | 1.7 | 0.9 | 4.5 | 2.7 | 1.1 | 6.1 | 1.5 | 0.9 | 4.7 |
| LCD1 | 0.8 | 0.8 | 3.5 | 0.9 | 0.5 | 3.0 | 0.9 | 0.6 | 2.9 | 3.5 | 2.8 | 11.2 |
| LCD2 | 0.9 | 0.7 | 4.2 | 3.1 | 1.1 | 6.5 | 3.2 | 1.6 | 9.7 | 3.3 | 1.6 | 10.4 |
| LCD3 | 1.0 | 0.6 | 4.0 | 3.7 | 1.7 | 8.9 | 3.7 | 1.8 | 9.0 | 4.2 | 2.3 | 11.6 |
| PR1 | 1.4 | 1.0 | 4.5 | 1.7 | 1.2 | 6.5 | 1.9 | 1.3 | 7.2 | 5.6 | 2.2 | 12.3 |
| PR2 | 0.3 | 0.2 | 1.0 | 2.1 | 1.1 | 6.5 | 2.6 | 1.3 | 7.2 | 7.3 | 3.3 | 15.0 |
| Average | 0.8 | | | 2.2 | 1.1 | 5.9 | 2.5 | 1.3 | 6.8 | 4.0 | 2.1 | 10.3 |

Table 2: Forward Error ΔE Mean (μ), standard deviation (σ) and maximum

| | LUT | | | Lin | | | Lin+ | | | Mask | | |
|---------|-------|----------|------|-------|----------|-----|-------|----------|-----|-------|----------|------|
| | μ | σ | max | μ | σ | max | μ | σ | max | μ | σ | max |
| CRT1 | 1.5 | 0.9 | 6.1 | 2.4 | 1.2 | 5.4 | 2.2 | 1.3 | 5.2 | 2.6 | 1.3 | 7.1 |
| CRT2 | 1.6 | 1.1 | 7.6 | 1.7 | 0.9 | 4.5 | 2.7 | 1.1 | 6.1 | 1.5 | 0.9 | 4.7 |
| LCD1 | 1.8 | 1.6 | 11.6 | 0.9 | 0.5 | 3.0 | 0.9 | 0.6 | 2.9 | 3.5 | 2.8 | 11.2 |
| LCD2 | 2.4 | 2.2 | 13.2 | 3.1 | 1.1 | 6.5 | 3.2 | 1.6 | 9.7 | 3.3 | 1.6 | 10.4 |
| LCD3 | 2.7 | 3.2 | 27.6 | 3.7 | 1.7 | 8.9 | 3.7 | 1.8 | 9.0 | 4.2 | 2.3 | 11.6 |
| PR1 | 2.8 | 1.8 | 10.7 | 1.7 | 1.2 | 6.5 | 1.9 | 1.3 | 7.2 | 5.6 | 2.2 | 12.3 |
| PR2 | 1.9 | 1.5 | 8.9 | 2.1 | 1.1 | 6.5 | 2.6 | 1.3 | 7.2 | 7.3 | 3.3 | 15.0 |
| Average | 2.1 | | | 2.2 | 1.1 | 5.9 | 2.5 | 1.3 | 6.8 | 4.0 | 2.1 | 10.3 |

Table 3: Backward Error ΔE Mean (μ), standard deviation (σ) and maximum

With respect to forward or backward error, we see that the 3D LUT is the most accurate, followed by the linear, Linear+ and Masking models (Table 2, Table 3). A comparison of backward error distributions (Figure 8) shows that the linear model had

tightest distribution for each device, while the distribution for 3D LUT tended to have a number of high-error outliers. The cause of these outliers becomes apparent when the error values are plotted by chromaticity. Figure 9 shows the chromaticity coordinates for points that are greater than half the maximum error for each model/device combination. Observe that the largest errors for the 3D LUT are often on or near the gamut boundary, which is where the tetrahedral interpolation tends to fall apart.

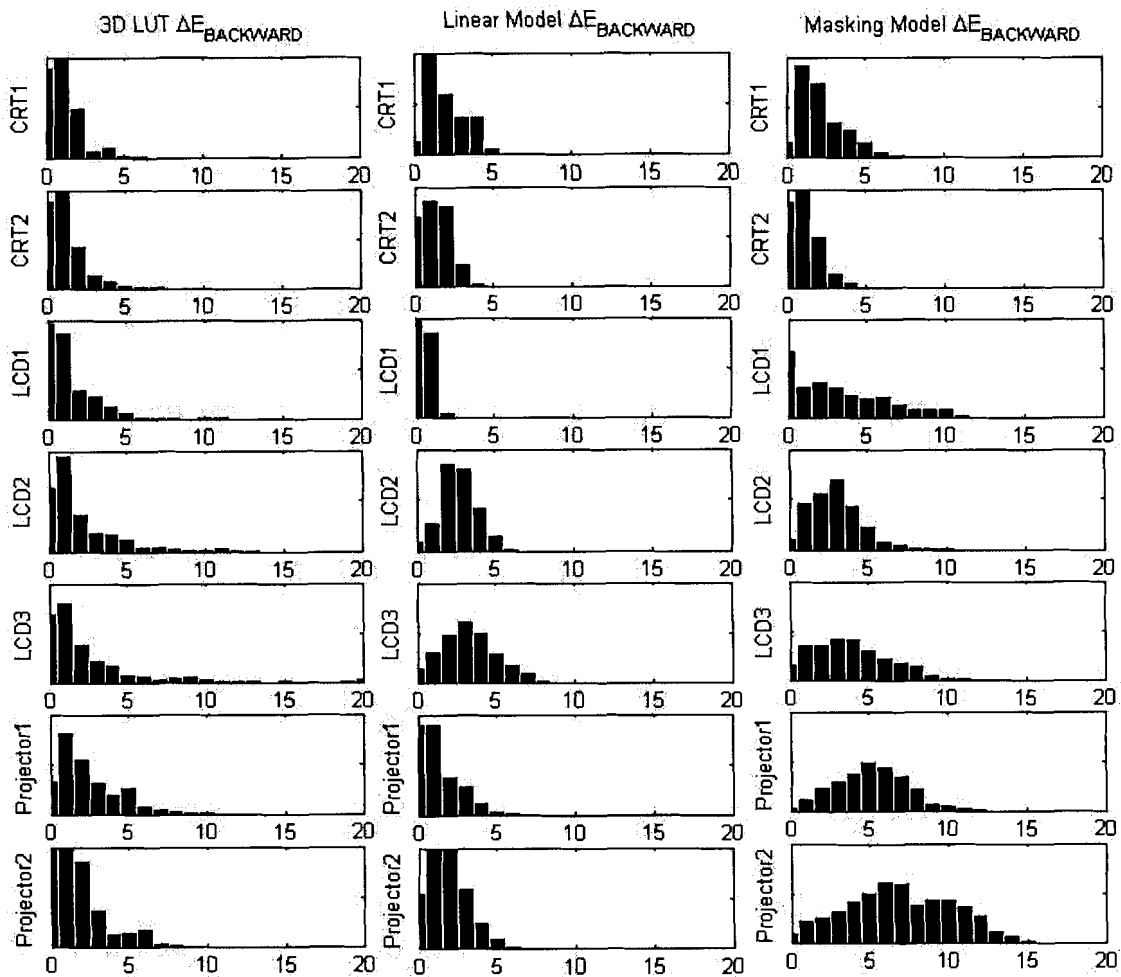


Figure 8: Backward Error distribution for each characterization model on each device. ΔE error value is shown on the horizontal axis and histogram counts are shown on the vertical axis.

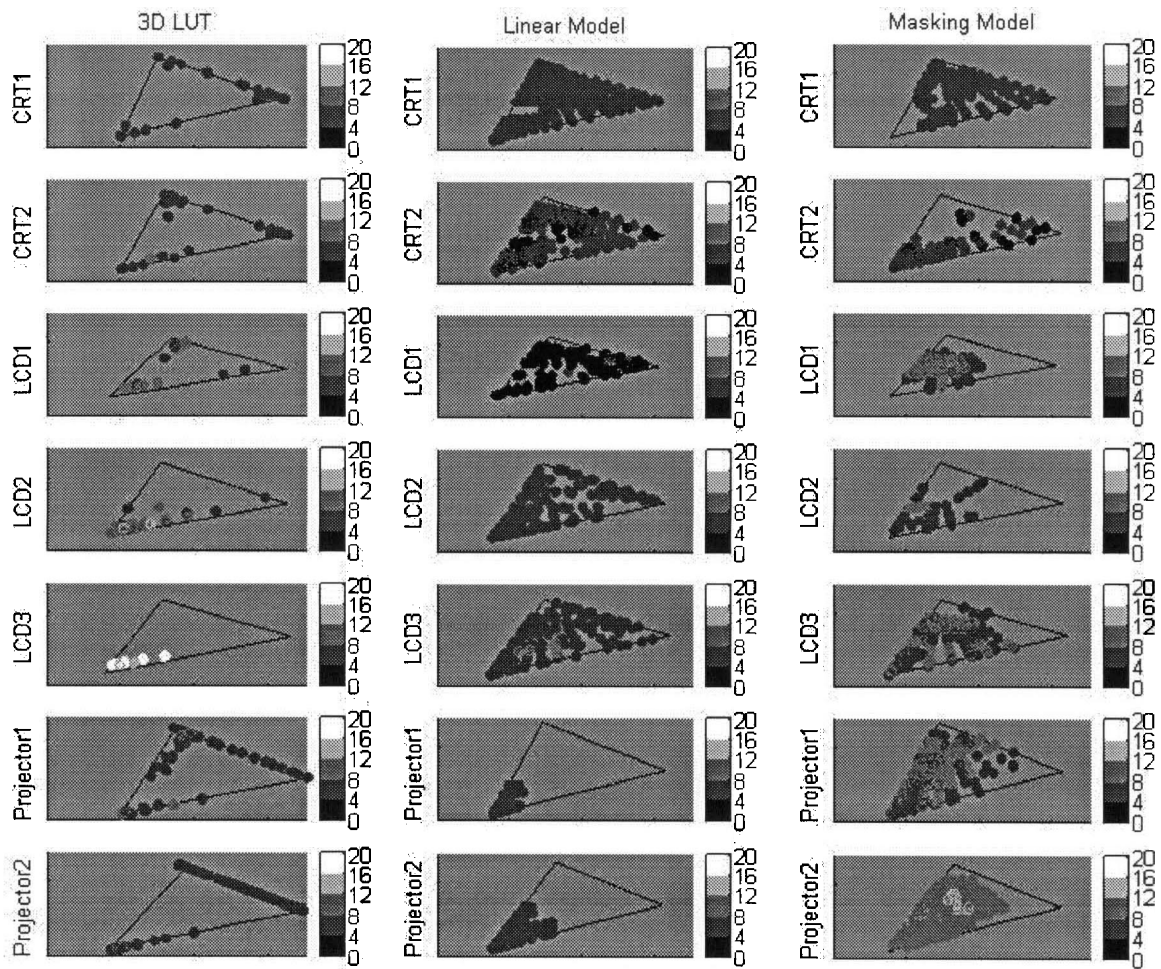


Figure 9: Backward Error vs Chromaticity. The horizontal and vertical axes are $X/(X+Y+Z)$ and $Y/(X+Y+Z)$ respectively

For the linear model, the highest errors are fairly well distributed across the chromaticity space for all devices except the projectors, which have a distinct problem in the blue region. This is most likely due to the non-monotonicity exhibited by the projectors in the blue output curves. As mentioned in the implementation section, the monotonicity correction stage is a potential source of error for all devices. However, it appears to be adding very little error for devices that do not have a monotonicity problem (Table 4). The most notable increase in error was seen with the PR1, which also had the most trouble with non-monotonicity.

| | Uncorrected | Corrected | % Increase |
|----------------|--------------------|------------------|-------------------|
| CRT1 | 2.4 | 2.4 | 0.0% |
| CRT2 | 1.7 | 1.7 | 0.0% |
| LCD1 | 0.9 | 0.9 | -0.7% |
| LCD2 | 3.1 | 3.1 | 0.0% |
| LCD3 | 3.5 | 3.7 | 4.7% |
| PR1 | 1.5 | 1.7 | 12.4% |
| PR2 | 2.1 | 2.1 | -0.8% |
| Average | 2.2 | 2.2 | 2.3% |

Table 4: Percent Increase in Forward ΔE Error Due to Monotonicity Correction

The results in Table 2 and Table 3 on page 18 show that the average error for the Linear+ model was nearly the same as that for the standard linear model. Recall that the goal of Linear+ is to ensure that white is predicted correctly at the possible expense of other colours. This indicates that “perfect” white can be achieved without much degradation in other colours. Informal visual comparisons also indicate that this model is the best one to use for computer-generated media such as presentation slides and charts.

The Masking Model was expected to out-perform the linear model whenever there was an issue with channel interaction. The model’s best performance relative to the linear model was on CRT2, where it is only slightly better than that of the linear model. For all other devices, the Masking Model exhibited higher error than the other models. The primary pitfall of this model is that it depends on constant chromaticity of the “combined primaries” (CMYK). It is clear from Figure 5 on page 10 that this assumption is incorrect for the LCD monitors and projectors.

The chromaticity shift caused by dissimilarity in the shapes of the R, G and B response curves causes the input the linearization step to fail. Figure 10 shows an example of an unsuccessful linearization for the black channel for PR1 in the masking

model – note that none of the lines are straight. This explains why the performance of the masking model was better for CRT monitors than any of the other devices – the CRTs do not have the shifting chromaticity problem.

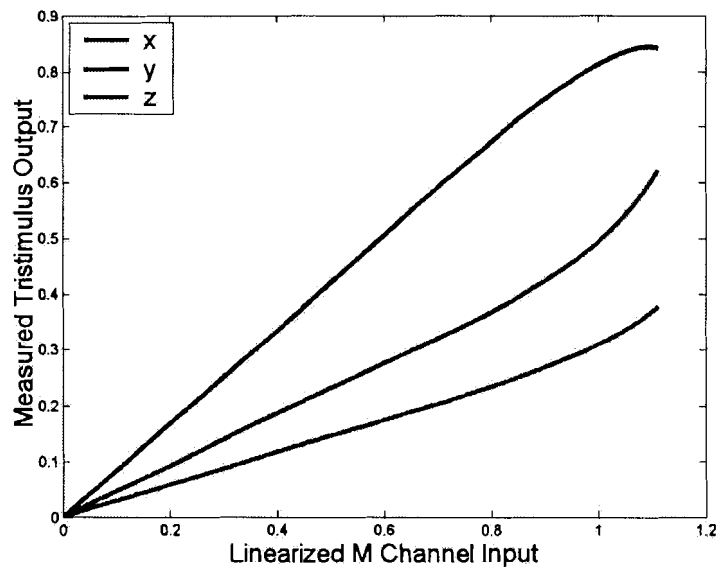


Figure 10: Linearization Failure for the Black Channel on PR1

With respect to efficiency, the linear model is also the top performer. The linear model is slightly faster than the masking model and nearly 20 times faster than the 3D LUT. The linear model also requires less than half the storage space of the masking model, and less than 1/300th the storage space required for 3D LUT (Table 5).

| | Time | Space |
|---------|-------------|--------------|
| Linear | 1.0 | 1.0 |
| Masking | 1.2 | 2.3 |
| 3D LUT | 17.0 | 333.4 |

Table 5: Experimental Running Time and Storage Space as multiples of linear model Usage

CONCLUSION

Several display characterization models were implemented in this paper: a 3D LUT, a linear model, an extension to the linear model, and the Masking Model. These characterization models were each tested on seven devices: two CRT Monitors, three LCD monitors and two LCD projectors.

Several general observations were made with respect to collecting characterization data. We found that the phosphor stabilization time on the CRT monitors was much longer than expected, and can take up to 10 seconds. In practice, a delay time of 2500 ms between a colour change and subsequent measurement resulted in acceptable error levels. With respect to integration time, we showed that longer integration times lead to more stable measurements, and propose that measurements on CRT monitors be taken with integration times that are multiples of the display scan rate. In addition, it was shown that a training set of 10 points data per axis is sufficient for the linear model (Figure 7).

Although recent papers have indicated that the linear model is not applicable to LCD monitors [6], it worked well for the LCD devices tested in this experiment. Furthermore, channel interaction was pronounced on the CRT monitors than on several of the LCD displays. The primary issue with the LCD displays was the fact that the response curves for the three input channels were dissimilar, leading to chromaticity shift of combined colours (CMYK). This problem affected the masking model but not the linear model.

Despite these issues, all three models yielded mapping errors of less than 15 ΔE . The 3D LUT model was slightly more accurate than the other models, but it is too

cumbersome for actual use. The linear model was the most efficient, with accuracy nearly as good as to the 3D LUT.

The primary drawback of the linear model is that it can be adversely affected by channel interaction. A slight modification to the linear model is presented in the Linear+ model that uses a simple white-point correction technique to ensure correct prediction of white. Our results indicate the Linear+ model is able to guarantee white-point accuracy with minimal degradation for other colours.

CONTRIBUTION SUMMARY

The preceding sections of this paper represent a joint effort involving both myself and Behnam Bastani, and will be published in a conference paper at CGIV 2004, where I will be making an oral presentation on the topic. The characterization models were implemented in Matlab, and were largely a joint effort. Module ownership was roughly divided such that I primarily developed the 3D LUT and Masking Model engines, while Mr. Bastani developed the Linear Model and Gamut Mapping engines. With respect to data analysis and model testing, Mr. Bastani focused on two-device gamut mapping results while I focused on analysing measurement error and device characterization errors.

My analysis of measurement error led to some drastic changes in the way we collect and use data. I found that the level of measurement error was considerably higher than expected, and developed a number of techniques to improve our results. These included performing measurements in bursts of 5, averaging multiple bursts for each sample point, randomly scheduling the colours rather than sequential testing, adding a measurement delay between colour display and measurement, and using a self-correcting integration time.

In order to enable this massive data collection effort, I designed, developed and implemented a software tool called SpectraCOM. This application controls the spectrometer and was used to run all of our automated testing, and will be useful for future work beyond this project. Appendix A gives details on the design and use of SpectraCOM. Other individual contributions include the Gamut Viewer (Appendix B), the Microsoft PowerPoint® sample interface (Appendix C), and the initial concept and implementation of the warping technique used for gamut mapping (Appendix D).

APPENDIX A: THE SPECTRACOM SOFTWARE

This project would not have been possible without a fast and flexible tool for collecting spectrometer measurements of screen colours. Towards that end, I developed SpectraCOM – a highly flexible application that controls the spectrometer and manages custom test suites. This program can display screen colours, take measurements, and store results in a database for easy access. It can be used through a friendly GUI interface, or as a scriptable COM object. An example of the interface is given in Figure 11

This section of the paper will provide support and usage information for future users of the SpectraCOM software. All of the files related to this implementation are stored on the project CD available from Brian Funt.

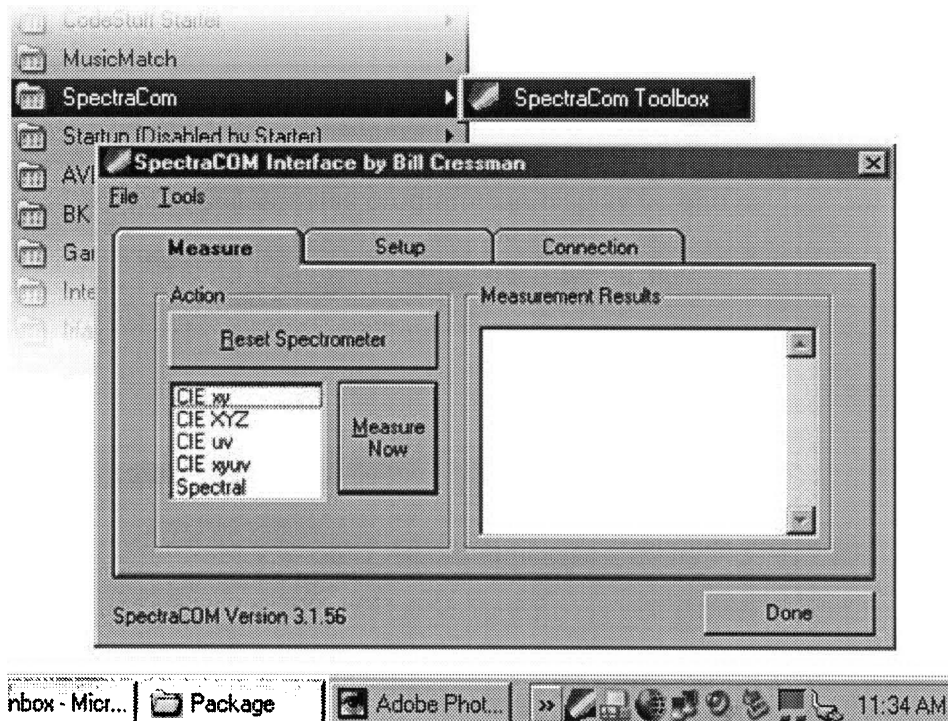


Figure 11: The SpectraCOM Interface

Getting Started

In order to get started using SpectraCOM, you just need to install the package. If you want to use the gamut measurement tool then you also need to define an ODBC Data Source Name (DSN) called SpectraCOM and point it to a valid copy of the database. An outline of the installation procedure is as follows:

1. Run the installation package (setup.exe) provided on the SpectraCOM disk. This will install the software under a default location and add a program group to the Windows start menu.
2. Connect the Spectrometer with a serial cable
3. From the Windows Start Menu, select SpectraCOM Toolbox as shown in Figure 11. This will open the toolbox window, and put the icon in the system tray.
4. Once the spectrometer is connected and turned on, click the "Reset Spectrometer" button in the toolbox window. The spectrometer screen should flash, and then indicate that an "S" command was sent.
5. Select a measurement type and then click "Measure Now". You should hear the spectrometer click, and a response should come back within a couple seconds. If the spectrometer does not work, refer to *Troubleshooting SpectraCOM* on page 39.

The SpectraCOM application is designed to work closely with a custom Microsoft Access Database (MDB file) for storing gamut measurements. The connection information for this database must be defined in an ODBC DSN called "SpectraCOM". ODBC allows a developer to store information about databases in one place that can be shared among applications. Each set of connection information is associated with a simple name (a DSN). Then, if the database information ever changes, we only need to update the DSN information in ODBC rather than updating all the applications that use it

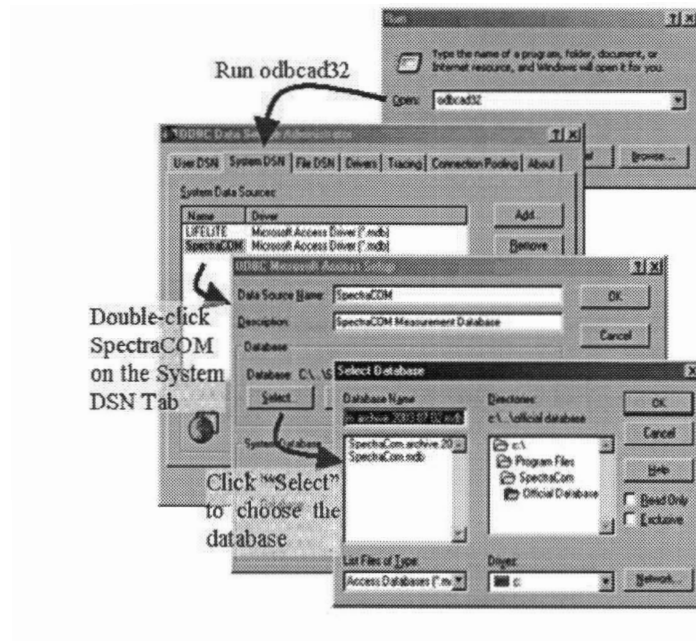


Figure 12: Updating the ODBC DSN

Installing the SpectraCOM DSN

In order to use the database, the computer must have a DSN called SpectraCOM that points to a valid copy of the SpectraCOM database. Although the installation package creates a blank copy of the database in the default installation location, all of the computers in the computational vision lab have DSN's that point to the database on the Gamut computer. These DSN were created manually, using the following steps (See Figure 12):

1. Run the ODBC Administrator (odbcad32.exe)
2. Select the "System DSN" Tab
3. Double-click on the SpectraCOM DSN
4. In the Microsoft Access Driver window, click "Select..." to choose a new database
5. Use the browser to locate the shared MDB file
6. Click OK, OK, OK

This DSN will be used by both the SpectraCOM program and the Matlab programs that access the gamut measurement data.

Using the SpectraCOM Interface

There are two ways to access the functionality of the SpectraCOM object. The SpectraCOM Toolbox wrapper application (available from the Start menu) exposes the functionality through a windows interface and from a persistent toolbar icon in system tray. Alternatively, the SpectraCOM OCX can be included in your own software project. The next few sections will review the functionality available through the COM interface, and the section entitled *Scripting with SpectraCOM* on page 34 will deal with using the SpectraCOM OCX in a scripting language.

As mentioned above the SpectraCOM Toolbox can be launched from the Windows Start menu. Once launched, there are three possible windows that can be displayed:

1. The SpectraCOM Interface window, which has three tabs and a few menu options.
2. The Spectral Chart Window displays the results of spectral measurements.
3. The Gamut Measurement Tool window helps run test suites for measuring a device gamut by displaying and measuring colour patches on the screen.

In addition to the interface windows there is a system tray toolbar icon with a right-click menu (Figure 13). Double-clicking the icon will open the SpectraCOM Interface Window. From the right-click menu, the user can open all three of the windows, reset or disconnect the spectrometer, and close the program.

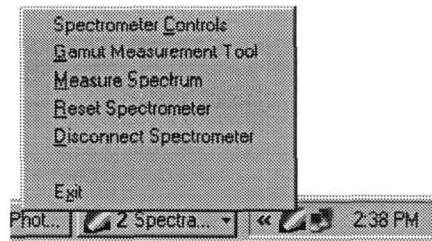


Figure 13: SpectraCOM System Tray Right-Click Menu

- ① **TIP: Closing windows that were opened using the toolbox will not stop the toolbox from running, and may not close the port. In order to ensure that the port is free, you must right-click the "SpectraCOM System Tray icon and choose "Exit"**

The SpectraCOM Interface Window

The **Measure Tab** allows the user to reset the spectrometer, which re-applies the settings from the Setup Tab. It also allows the user to take a number of different measurements. The most notable of these measurements is the Spectral option, which will open the SpectraCOM Chart Window.

The **Setup Tab** on the SpectraCOM Interface (Figure 14) allows the user to control all of the important setup parameters for the SpectraScan 650 Spectrometer. The Sync Frequency allows the user to specify the expected frequency of the light source being measured. It can be set to Auto, in which case the spectrometer is supposed to automatically detect the frequency. This feature was not used in this study, and did not appear to work on CRT monitors. All measurements in this study were taken using "DC Mode".

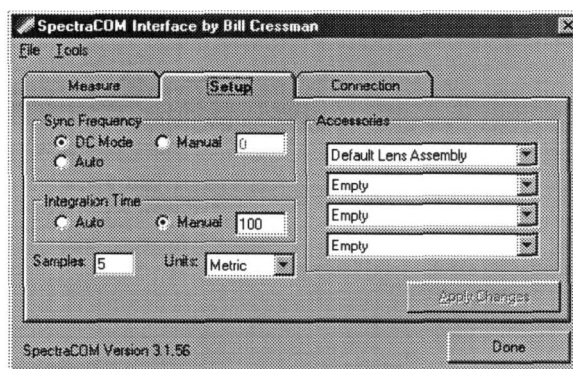


Figure 14: The Setup Tab on the SpectraCOM Interface

- ① **Working with the Setup tab when the spectrometer is offline will cause the application to respond very slowly. If you are experiencing this problem, hit “escape” to cancel the spectrometer request, then toggle to the Measure tab and reset the spectrometer. You may need to hit “escape” again when you change tabs, since the setup is also applied when the Setup tab is de-selected.**

The Integration time option allows the user to specify the shutter speed in milliseconds. A value of 100 indicates that the shutter should remain open 100 milliseconds for each measurement. I recommend setting the integration time to a multiple of the screen scan time. For example, if the scan rate is set to 100 Hz, set the integration time to a multiple of 13.3 ms. It is generally better to err on the side of long integration time in order to ensure repeatable measurements.

The *Samples* option allows the user to specify the number of samples that are taken with each measurement. The average of these samples will be returned from the spectrometer. The remaining options (units and accessories) are set to the default values and were not changed in this study. I don't know what they do and I fear them.

The **Connection Tab** on the SpectraCOM Interface is intended for debugging connection response problems with the spectrometer. It allows the user to select the

COM port that will be used, open and close the port, update the timeout seconds, and send custom commands to the device. For this study, the timeout seconds was set to a large number (90 seconds) to allow for multiple re-test attempts when running in gamut-test batch mode. Sometimes when testing dark colours such as black, the device needs to increase the integration time to 800 ms or more, which can take a considerable amount of time.

The SpectraCOM Chart Window

The SpectraCOM Chart window appears when the user selects Spectral Measurement from the SpectraCOM Interface, or selects the “Measure Spectrum” option from one of the menus. The chart will automatically add subsequent measurements until either the window is closed or the user selects “File→Clear Data”. The user may click on the chart points to see detailed data information.

The Chart also has the ability to export its data to a CSV file from the File→Export menu option. This will create a CSV file with a column of data for each series.

The SpectraCOM Gamut Measurement Window

The Gamut Measurement Window helps the user run test suites that measure a monitor’s gamut. The data from the test is stored in the SpectraCOM MDB using the SpectraCOM DSN. This data can be extracted directly into Matlab, or can be converted to text files using the Excel spreadsheet provided with this project.

The first section of this window defines the **Sample Plan** either using an auto-generated sample plan of equal step size, or by reading a sample plan from a file. The file should be a simple comma separated variable (CSV) file with an R, G, B setting on each row. A sample test file is included in the default installation, entitled “samples.csv”.

The sample plan file should not include repetitions – use the “Test Repetitions” option on the interface instead.

The checkbox for **Record Spectral Data** allows the user to specify that the test will record a full spectral sequence for each sample in the test. This option will make the test run more slowly, and collects 100 records for each sample.

The **Spectrometer Reset** Interval option in the Sample Plan area allows the user to define how many measurements should be attempted before resetting the spectrometer. This option was added because it was found that the spectrometer encountered some problems after a few thousand test points. Resetting the spectrometer before the problem happens is an effective way to prevent it.

The three options on the right side of the window allow you to control measurement parameters – burst sample size, integration time and measurement delay. The measurement delay controls the number of milliseconds between the display of a colour and the start of measurement.

The remaining options on the Gamut Measurement Window are for recording the test conditions in order to make it easy to pull this data back from the database.

Upon clicking Start, the user will be presented with the “Begin Testing” dialog. This dialog presents a few reminders to help the user prepare for the test, as well as a gray-scale pattern for adjusting the monitor brightness. Upon starting a test, the user should adjust the brightness on the monitor until all of the gray colours are visible, record the settings in the text boxes provided, and click “OK” to start the test.

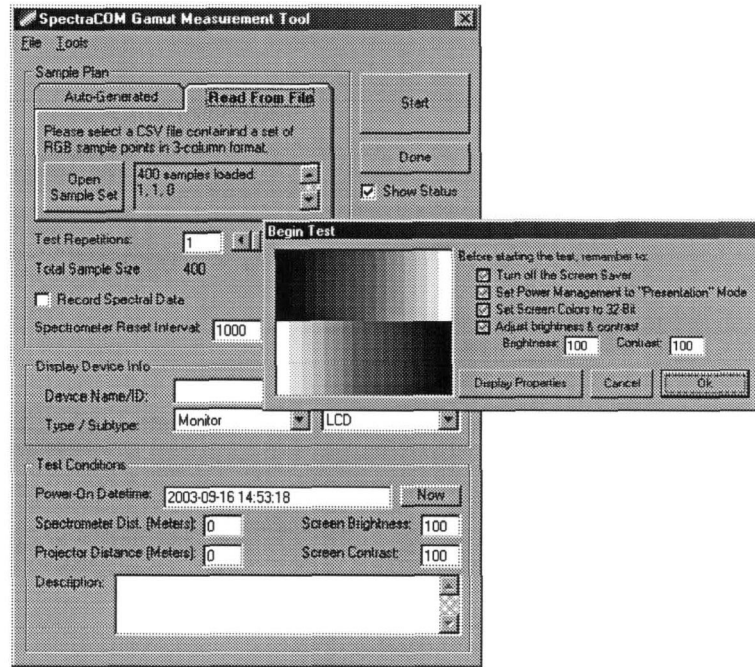


Figure 15: The Gamut Measurement Tool and Start Test Window

- ① **Make sure that you turn off the screen saver and power options before running the gamut measurement test.**
- ① **Double-check the spectrometer settings before running a test. Make sure that the comm port timeout is high (like 120 seconds) in case some measurement takes a long time. You don't want your test to stop halfway because of a timeout.**

Scripting with SpectraCOM

Scripting with the SpectraCOM Object is intended to be an easy alternative to using the interface, but I'll admit it's tough to beat the interface. The COM object can also be used as a reference in any object-oriented programming language such as Java, C++ or Visual Basic. The usage in compiled VB is very similar to the examples shown here, which are for VBScript.

In order to get started in VBScript, simply create a text document and change the extension to “.vbs” and double-click the file to run it. A sample file entitled “Test.vbs” is included in the default installation. The first step in the code is to create an instance of the SpectraCOM Object. The code for doing this in VBScript is as follows:

```
set obj=createobject("SpectraCOM.Spectrometer")
```

The line of code shown above will create a Spectrometer Object and call it obj. Immediately after creating the object, it is a good idea to call the obj.reset method, which will verify that the spectrometer is connected, and re-set it's configuration. The following table the various properties and methods of the SpectraCOM Object.

- ① **TIP: Microsoft Visual InterDev offers code completion for VBScript but getting it started is a little tricky. We need to trick InterDev into thinking that the file is ASP, which uses VBScript. To do this, right-click the file in the InterDev Project Explorer and choose “Open With”, then select HTML Editor. Finally, put the “<%” symbol at the top of the file to tell it that the ASP script is starting. Once you have done that, you will be able to see the correct syntax highlighting and code-completion. You'll need to comment out or delete the “<%” before running the program.**

Table 6 and Table 7 list some important properties and methods of the SpectraCOM.Spectrometer object.

| Property Name | Description |
|--|---|
| AccessoryID1 – 4 (Spectrometer Setting) | These four properties control the Accessory ID values that are set on the spectrometer device. Refer to the spectrometer manual for details. |
| AutoApplySettings | When this property is set to true, changes to Spectrometer Properties will be immediately applied to the spectrometer, and any errors will be thrown. If this value is set to false, the user must remember to call the applySetup() method in order to send setting changes to the spectrometer. |
| CommPort | This property is a pointer to the CommPort visual basic object that is being used for serial communication. |
| CommTimeout | This setting defines the default time in milliseconds for the program to wait for responses from the spectrometer. |
| IntegrationTime (Spectrometer Setting) | This variable reads and sets the integration time used by the spectrometer for taking measurements. Integration time is essentially the shutter speed in milliseconds. |
| ProductName | This value tells the name of the product (i.e. SpectraScan 650) and is reserved for future use. |
| Response | This variable holds the most recent response information from the Spectrometer. In general, a user will use the formatted information that is returned from the measure() method rather than parsing the response directly. |
| SampleSize (Spectrometer Setting) | This setting tells the spectrometer how many samples to take per measurement burst. |
| SyncFrequency (Spectrometer Setting) | This variable tells the spectrometer what sync frequency to use for taking measurements. Valid values can be 0 (DC Mode), 1 (Auto), or 40 to 240 (hz). |
| UnitsType (Spectrometer Setting) | This variable sets the spectrometer Units. See Spectrometer manual for details. |

Table 6: SpectraCOM Object Properties 

| Method Name | Description |
|----------------|---|
| applySetup() | Applies the current public configuration variables to the spectrometer. These variables include <i>AccessoryID1</i> thru 4, <i>SyncFrequency</i> , <i>IntegrationTime</i> , <i>SampleSize</i> and <i>UnitsType</i> . This function will throw an error if the spectrometer does not return a successful response. |
| cancelAction() | This method will tell the object to cancel any pending action. This is important because the spectrometer may have a long timeout (90 seconds is not unusual), so it is nice to be able to kill an action without waiting. |
| configure() | Opens the “Configuration and Control Dialog”, referred to in this paper as the main SpectraCOM Interface dialog box. The dialog box is opened with the “Setup” tab selected. |
| connect(port) | Connects to the serial port # indicated by port |

| | |
|---|--|
| DecodeQualityNumber(<i>qNum</i>) | This method returns a string for any one of the known quality numbers indicated by the argument <i>qNum</i> . |
| disconnect() | Disconnects from the serial port |
| measure(<i>MeasureMode</i> , <i>Quality</i> , <i>Units</i> , <i>aValues</i> , [<i>bGetOldData</i>]) | This function tells the spectrometer to take a measurement. The first argument <i>MeasureMode</i> must be one of the enumerated <i>MeasureMode</i> values (see above). The <i>Quality</i> , <i>Units</i> , and <i>aValues</i> arguments are used for return values, where <i>Quality</i> indicates the quality number that can be decoded using the <i>decodeQualityNumber(qNum)</i> method. The <i>aValues</i> array will contain an array of output values depending on the measurement type. This method also returns the integration time used to make the measurement as a return value. |
| MeasureAndShowSpectrum([<i>bGetOldData</i>]) | This method will measure the spectrum and show the spectral chart window. Setting the optional argument <i>bGetOldData</i> to true tells the spectrometer to use the values from the last measurement rather than measure again. |
| measureScreenColor(<i>MeasureMode</i> , <i>r</i> , <i>g</i> , <i>b</i> , <i>Quality</i> , <i>Units</i> , <i>aVal</i>) | This function is a combination of the <i>Measure</i> and <i>ShowColor</i> methods described in this section. |
| reset() | Attempts to wake up the spectrometer. It sends a setup command to verify that the spectrometer is responding. This function will throw an error if there is a communication error. It returns true on success, but never returns false (it errors instead). |
| sendCommand(<i>strCommand</i> , [<i>strEOR</i>], [<i>intTimeout</i>]) | This function sends the command string indicated by <i>strCommand</i> to the spectrometer. Optional argument <i>strEOR</i> tells the spectrometer the End Of Response identifier, which defaults to vbCrLf. The <i>intTimeout</i> identifier tells the program how long to wait in milliseconds before declaring a timeout. If the timeout value is missing, then it defaults to the spectrometer setting for <i>ConnTimeout</i> . If the spectrometer does not issue a successful response in less than the timeout time, an error is thrown. |
| ShowColor(<i>r</i> , <i>g</i> , <i>b</i> , [<i>strCaption</i>], [<i>Modal</i>]) | This method will show a large colour panel covering the entire screen, using the input variables <i>r</i> , <i>g</i> , and <i>b</i> . The optional string <i>strCaption</i> will appear in the upper left corner of the screen. The " <i>Modal</i> " option allows the user to open the control in modal or non-modal mode. Note that VBScript does not allow non-modal dialog boxes. |
| showControls(<i>Modal</i>) | This function opens the SpectraCOM Interface Dialog Box with the "Measure" tab selected. The " <i>Modal</i> " option allows the user to open the control in modal or non-modal mode. Note that VBScript does not allow non-modal dialog boxes. |
| showGamutTool(<i>Modal</i>) | This method opens the Gamut Measurement Dialogbox. The " <i>Modal</i> " option allows the user to open the control in modal or non-modal mode. Note that VBScript does not allow non-modal dialog boxes. |
| testFrequency() | This function tells the spectrometer to run the frequency test for the "Auto" frequency sync mode. When using the Auto mode, this test must be run before taking any measurements. In that case, this function is called automatically by the <i>measure()</i> method. |

Table 7: SpectraCOM Object Methods

A short sample script using these properties and methods is as follows:

```

set o=createobject("SpectraCOM.Spectrometer")
o.reset
o.syncfrequency=0 'DC Mode
o.integrationtime=200 'Manual, 200 ms
o.samplesize=3 '3 Measurements per request
o.Measure(2,Q, A, aVal, false) 'Measure XYZ
msgbox aVal(1) & "," & aVal(2) & "," & aVal(3)

```

The Database

The database was designed for recording data for large suites on individual display devices. There are a number of tables in the database, as shown in Figure 16.

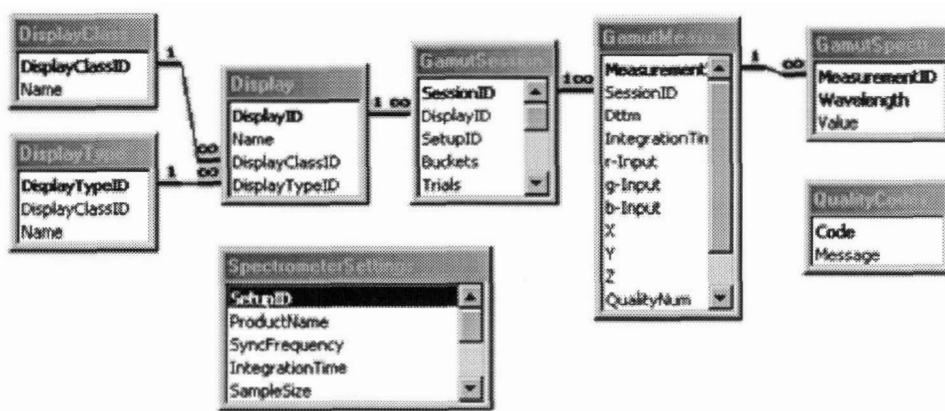


Figure 16: The Database Schema

The *DisplayType* and *DisplayClass* tables are lookup tables for the display types (CRT, LCD, Projector) and classes (Monitor, Projector...), and are referenced by the display table. The records of the *Display* table are then used by the *GamutSession* table, which contains one record for each test session collected using the Gamut Measurement Tool. Each measurement of the session is stored in the *GamutMeasurement* table. If the “Record Spectral Data” option was checked on the Gamut Measurement interface, then 100 spectral data records will be stored in the *GamutSpectralData* table for each measurement.

In the latest version of the database, two additional tables have been added – *StandardTestName* and *StandardTestPoint*. These tables make it easy to pull out a data set for a standard test (101 linear, 10 interaction, etc) when the data for multiple tests is collected in a single gamut session. *StandardTestPoint* lists the RGB points that make up a standard test. The “*GMAP Query*” in the default database gives an example of how to query data for a standard test using these and other tables.

Accessing the Data

It is possible to query and update the data in the database from any application that supports ODBC, including Matlab. Several Matlab files that query data directly from Matlab are stored in the Utilities folder. In order to get started using these files, the Database Toolbox must be installed in Matlab and the SpectraCOM DSN must be installed on the computer (see *Installing the SpectraCOM DSN* on page 28). Once that is done, connecting to a database and running a query can be done in four-lines.

```
db = database('SpectraCOM', '', '');  
cur = cursor(db, 'SELECT * FROM Display');  
dat = fetch(cur);  
mat = dat.Data;
```

The above example will return a matrix containing all of the information in the *Display* table in the database. For more details on querying and updating data in the database, refer to the files in the Utility folder, which contains a number of examples and comments.

Troubleshooting SpectraCOM

In order to debug spectrometer problems, it is most useful to use the SpectraCOM Interface Dialog box. The following problem/solution pairs use this dialog box. First, make sure everything is plugged in and turned on.

Problem: No Response

Solution: If you are getting no response from the spectrometer, first check the connections – make sure you have the correct COMM port selected in the Connections panel of the setup screen. There is a toggle switch on the serial cable for the spectrometer. Try toggling the switch and re-setting the spectrometer again. Ensure that the Comm Port selection is correct, and reset the spectrometer again using the “Reset Spectrometer” button.

Problem: The Toolbox does not open

Problem: CreateObject Does not work in Scripting

Solution: It is possible that the SpectraCOM object did not get registered when the software was installed. Locate the SpectraCOM.ocx file on your hard drive, and execute the command “regsvr32 [ocx path]”.

Problem: Cannot connect to the database

Solution: Run the ODBC Administrator (Start→Run→”odbcad32”) and check in the System DSN tab to ensure that there is a “SpectraCOM” DSN. If the DSN is not there, you can create one using the ODBC Administrator interface. Select “Add”, then choose “Microsoft Access Driver”, set the name to SpectraCOM and use the Select button to locate the database.

If the SpectraCOM DSN is there, you can verify that it points to the correct location by double-clicking the name and then clicking the “Select” button in the window that appears.

If the SpectraCOM DSN is present and it points to the correct database file, but there is still a database problem, then it is possible that the database could be corrupted. Try pointing to a new, blank database file and see what happens.

APPENDIX B: THE GAMUT VIEWER TOOL

Early on in the project we discovered that simply mapping forward on one device and backward on another will not work very well unless the gamuts of the two devices are identical. In practice, there are always colours that one device can produce and the other cannot – out of gamut (OOG) colors. In order to help visualize this situation, I developed a simple C++ application called GamutViewer that can overlay two gamuts and allow for real-time rotation and translation of the image. The program takes as input the names of two gamut map raw data data files (GMAP files). These files must contain comma separated value (CSV) data with one record per row and columns for R,G,B,X,Y,Z. An example of the interface is given in Figure 17.

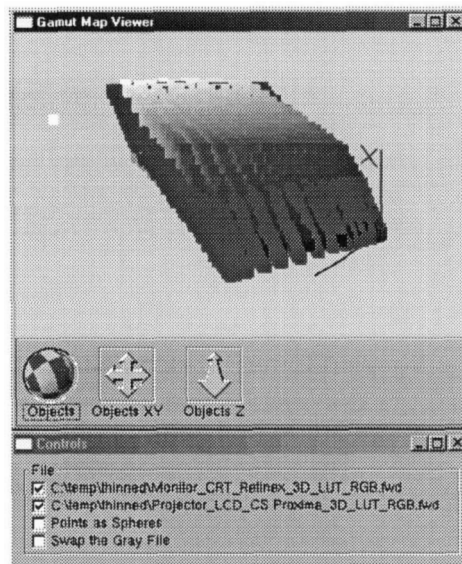


Figure 17: The GamutViewer Interface

In order to use the application, simply place the GamutViewer.exe program in the same folder where the two gamut files are located, then highlight the two files and drag them on top of the GamutViewer.exe file, as shown in Figure 18

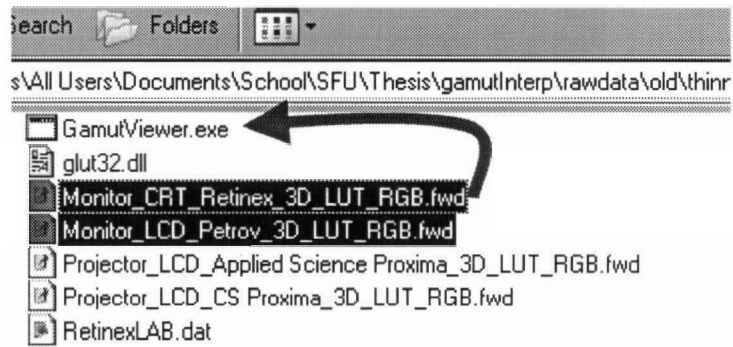


Figure 18: Starting GamutViewer by selecting two files and dropping them on the executable

Once the viewer is displayed, the user can show or hide either file, display the gamuts using either points or spheres, and toggle which gamut is shown in all gray.

Some examples are shown in Figure 19.

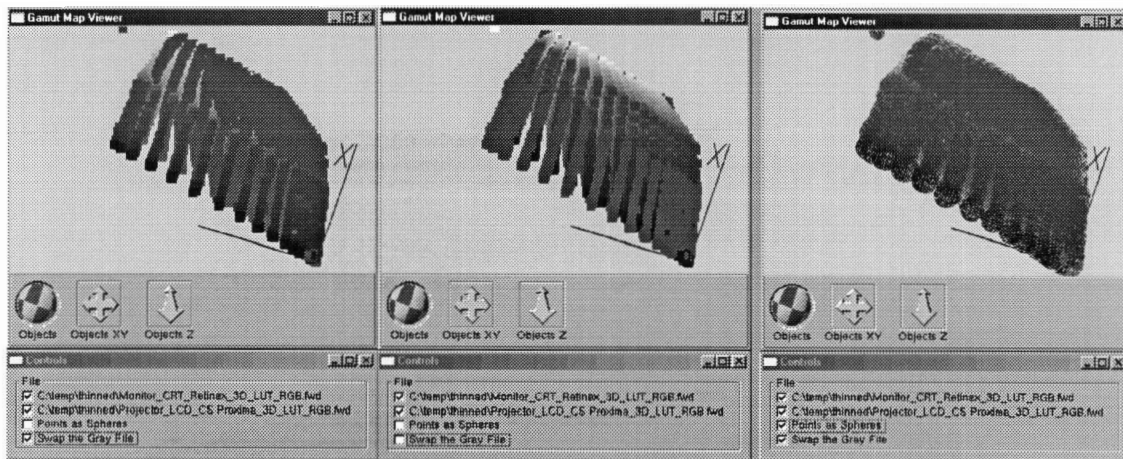


Figure 19: Example of swapping the gray file and showing points as spheres.

APPENDIX C: THE MS POWERPOINT® SAMPLE INTERFACE

One of the initial goals of this project was to make a friendly user interface for predicting the colours on presentations. One of the friendliest ways to do this is to incorporate the software directly into Microsoft Powerpoint®. This can be done fairly easily using the Visual Basic for Applications (VBA) components provided with PowerPoint. However, VBA is not a very powerful language for doing large calculations, as would be required for colour mapping. Therefore, I proposed a two-part architecture, where the colour mapping calculations are embedded in a C++ COM object, and the interface is supplied by VBA. I developed a sample application using this architecture to serve as a platform for future work.

The ColorPreview.ppa file is available on the project CD under the PowerPointPreview folder. The application automatically adds a few menu options to Powerpoint, as shown in Figure 20. The application is incomplete at this point, but provides a starting point from which to build the project.

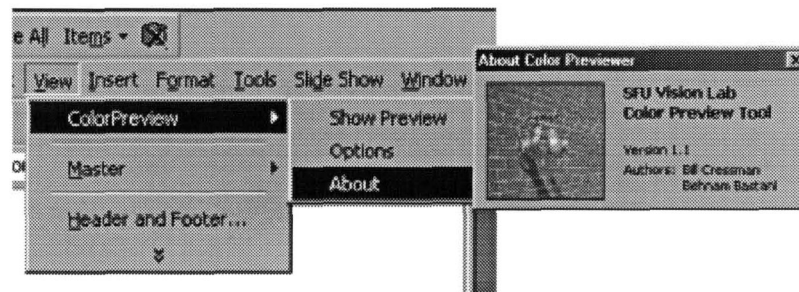


Figure 20: The clmageBuddy object in the project toolbox

Also included on the project CD is the code for the clmageBuddy object – an COM object written in C++ that can be referenced from VBA. Documentation specific to clmageBuddy can be found in the source folder on the project CD.

APPENDIX D: A NOTE ON GAMUT MAPPING

One of the initial goals of this project was to demonstrate successful prediction of colours across two devices. Originally it was postulated that the mapping might be possible if one gamut is completely inside the other, but even if this were the case it would lead to unsaturated colours on the device with a larger gamut. In practice, we found that it is not possible to map from one device to another without some form of gamut mapping.

Figure 21 shows two gamuts that have been aligned by removing monitor flare (black-point translated to the origin in XYZ space) and performing a diagonal matrix transform to scale the white-points to $[1,1,1]$ in XYZ space. Both gamuts are similar in shape (6-sided), with eight corners corresponding to the primary phosphor colors (RGB), the secondary mixed colors (CMY), black and white. For the devices used in this experiment, aligning the gamuts in this manner will always result in almost all corners of one gamut being outside the other gamut, and vice versa.

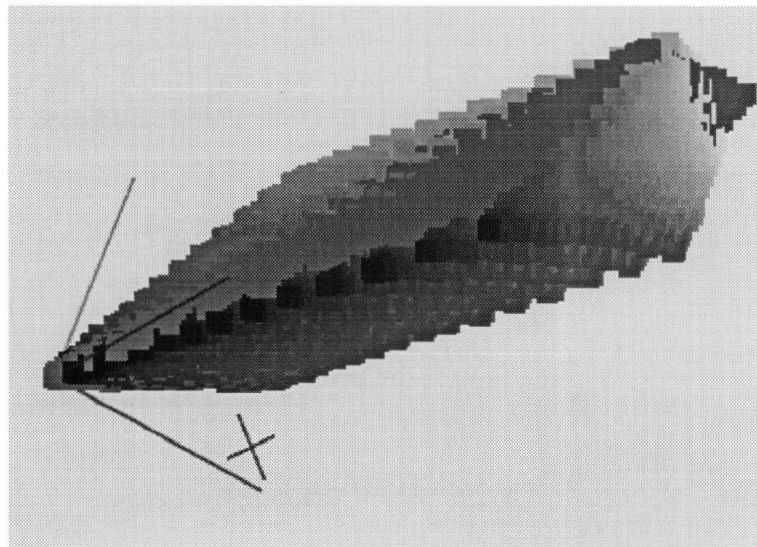


Figure 21: Sample output from the Gamut Visualization application developed for this project. (a) two gamuts, with one in gray, (b) one gamut with points shown as spheres

This observation makes it clear that characterization alone will not be sufficient for mapping between two devices – all of the pure colours will be out of gamut.

Diagonal Transform Scaling

Our first implementation of colour prediction did not do any special gamut mapping – it simply translated black to the origin and applied a diagonal matrix that set white to (1,1,1). As shown above, this technique will align only the the black and white corners, but not the other edges. This leaves us with a problem – how do you find a suitable RGB input value for an XYZ tristimulus value outside the gamut that a monitor can produce? The task of assigning values for these out of gamut (OOG) points is referred to as Gamut Mapping.

Clipping

Our first implementation of gamut mapping used a technique referred to as clipping – any point v that is outside of the gamut is orthogonally projected on to the gamut surface. In order to find the closest surface patch, we pick any point q_1 on the interior of the convex hull (preferably the centroid), and then compute a point set Q such that the faces of the gamut convex hull are perpendicular bisectors between each point and q_1 . Now the nearest point in $\{q_1, Q\}$ to v is associated with the nearest edge for projection. If it's q_1 then v is not out of gamut.

The biggest problem with this approach is that points near the corners of the gamut will still be out of gamut after projection (Figure 22). In this case, we iterate until the point comes inside gamut.

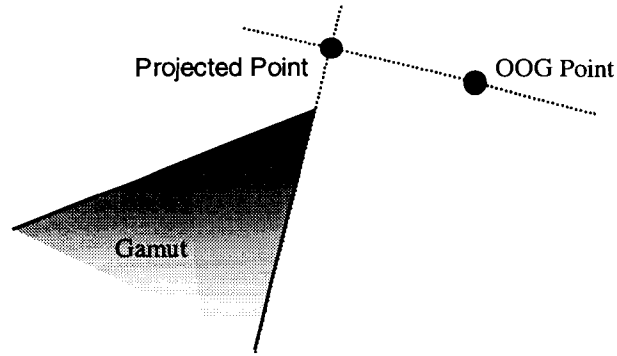


Figure 22: Example of a clipping operation that results in another OOG point

This technique proved to be very slow and led to some poorly selected colours. Recall that the shapes of digital display devices are usually similar (cube-shaped) and that most of the out of gamut points tend to fall in the corners, which is where the clipping algorithm is the slowest.

Scaling and Rotation

We could easily improve the clipping problem by first using a uniform scaling to set the length of the gray axis to $\sqrt{3}$ (the length of the diagonal through a unit cube). Using a uniform scaling rather than a diagonal transformation better preserves the relationship between colours. Two rotations are performed next. First, the gray axis is rotated to align white to (1,1,1). Next, the gamut is rotated around the gray axis to achieve the best alignment of corners.

Once the scaling and rotation have been performed, the clipping operation runs much more quickly and the resulting colours are better.

Warping

Warping is a novel gamut mapping technique specifically designed for two similarly shaped gamuts. We define a few anchor points on the source gamut with

known values in the target gamut (i.e. the corners) and divide the space into a Delaunay tessalation using the anchor points. We then warp the source gamut to the target using linear interpolation inside the tesselation cells.

My initial implementation of warping was applied to a pair of gamut maps before scaling and rotation, and the result was a gray axis that was severely curved, as shown in Figure 23.

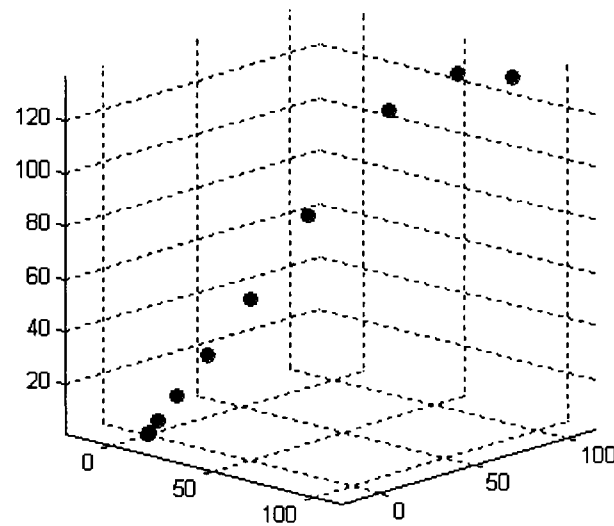


Figure 23: Example of gray axis curvature after warping without rotation or scaling

Applying scaling and rotation before warping greatly improves the performance, but the results are incomplete at this point. Mr. Bastani is currently extending this research by applying scaling and rotation before warping, and is achieving improved results.

BIBLIOGRAPHY

- [1] Amidror I. Scattered data interpolation methods for electronic imaging systems: a Survey. *J Electronic Imaging* 2002; 11.2:157-176.
- [2] Fairchild MD, Wyble DR. Colorime. Colorimetric Characterization of the Apple Studio Display (Flat Panel LCD). Munsell Color Science Laboratory Technical Report, 1998, <http://www.cis.rit.edu/mcsl/research/PDFs/LCD.pdf>. Accessed: 12/15/2003
- [3] Finlayson GD and Drew MS, White-point preserving color correction. Proc. IS&T/SID 5th Color Imaging Conference 1997; pp. 258-261.
- [4] Gibson JE, Fairchild MD. Colorimetric Characterization of Three Computer Displays (LCD and CRT), Munsell Color Science Laboratory Technical Report, 2000, <http://www.cis.rit.edu/mcsl/research/PDFs/GibsonFairchild.pdf>. Accessed: 12/15/2003
- [5] Kwak Y, MacDonald LW. Accurate Prediction of Colors on Liquid Crystal Displays. Proc. IS&T/SID 9th Color Imaging Conference 2001; 355-359.
- [6] Tamura N, Tsumura N, Miyake Y. Masking Model for Accurate Colorimetric Characterization of LCD. Proc. IS&T/SID 10th Color Imaging Conference 2002; 312-316.
- [7] Yasuhiro Yoshida and Yoichi Yamamoto, Color Calibration of LCDs. Proc. IS&T/SID 10th Color Imaging Conference 2002; 305-311.