Innovative Color Management Methods for RGB Printing

by

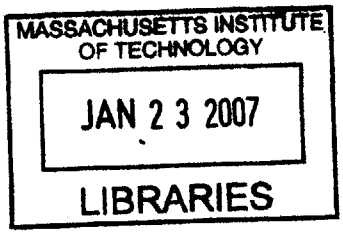
Wei Dong

Bachelor of Science, Applied Physics (2000)
Nanjing University of Science & Technology

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF ENGINEERING IN MANUFACTURING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SEPTEMBER 2006



Signature of Author: .....
Department of Mechanical Engineering
August 15, 2006

Certified by: .....
Kamal Youcef-Toumi
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by: .....
David E Hardt
Professor of Mechanical Engineering and Engineering Systems
Co-Chair, Manufacturing System and Technology, Singapore MIT Alliance

Accepted by: .....
Lallit Anand
Professor of Mechanical Engineering
Chairman, Department Committee on Graduate Students

Innovative Color Management Methods for RGB Printing

by

Wei Dong

Submitted to the Department of Mechanical Engineering
on August 15, 2006 in Partial Fulfillment of the
Requirements for the Degree of Master of Engineering
in Manufacturing

ABSTRACT

Re-calibrating a printer in response to systematic changes is measurement and labor intensive. In this study, a fast correction method with cycle-to-cycle control was proposed. The process includes two steps: the creation of look-up table using a characterization data set, and image color compensation in conjunction with Windows printing architecture.

Several types of correction models for determining printer characterization were proposed and evaluated, including polynomial models and neural network models. The most successful of these methods was the quadratic spline interpolation model, which removed most errors introduced by the changes of colorant and printing substrate. A significant reduction in error was realized by incorporating this technique into the color management program.

Thesis Supervisor: Kamal Youcef-Toumi
Title: Professor of Mechanical Engineering

ACKNOWLEDGEMENTS

The author wishes to express sincere appreciation to Professor Kamal Youcef-Toumi and Professor Andrew Nee for their assistance in modeling and solving the problem. They always made time to meet us, and keep the team on the right track. Thanks to Professor Kamal Youcef-Toumi for taking the trip all the way to Singapore to review the progress and to point out the areas that needed improvements. Thanks to Professor Andrew Nee for inspiring us and taking the time to review the thesis.

In addition, special thanks to Dr. Allan Zhang whose expertise in color printing system was helpful during the project. He instructed us every step of the way. He not only explained a lot of concepts to us, but more important encouraged us to explore new approaches. Thanks to Mr. Winson Lan for sponsoring this project and giving us the motivation with industrial background.

Thanks also to Xian Du, Curtis N. Vanderpuije and Ryan Wong for their valuable input. They are also the driving force behind the problem solving and software development efforts. Without them, the project would have less results to provide.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1. General Background

Due to the rapid progress on the technologies of computer graphics and printing devices, color printing systems have become more affordable to both business and home users. An investigation by Gartner Dataquest and IDC (International Data Corporation) indicated that color reproduction for digital cameras and web-based color applications is driving the printer market fervently [1-2]. Many commercial giants "are reaping hefty amounts of profits from the color printing market, and are looking forward to earning even more by reaching out to photo-quality reproduction." [3-4]

With the growth of the market, the extraction, manipulation, and reproduction of color images have become interesting research topics. One of the major problems among these topics is how to reproduce the color accurately. According to a test conducted on five typical color printers, it was found that output color shifted over the course of hours; larger or smaller clumps are forged due to the alternations of temperature and humidity; new toners offer better color stability than the old ones [5]. Some other tests showed an erratic color spectrum in some commercial printers. Color mismatch phenomenon is severe in the existing printing system.

The color mismatch may result from the nonlinear characteristics, for example, gamut mismatch between a scanner and printing device, printhead deterioration, printer halftone technique, cartridge or toner instability, and environmental fluctuation. In addition, the interaction of the colorants on print substrate is highly nonlinear, and has not been clearly addressed scientifically.

1.2. Current Industrial Solutions

Printer characterization, along with calibration, is critical to the performance of a printing system. To maintain color resemblance between color input and output, a lot of color calibration systems were developed in the past decades. They can be categorized into two basic proposes: *operator experience-based calibration system* and *ICC-based color management.* J. Rich weighted the two systems in terms of cost, ability, workflow setup and other three aspects, and summarized that the latter proved to be more efficient and better [6].

Conventionally, most ICC-based CMSs (Color Management System) are based on a look-up table scheme, where the content of the look-up table is manually defined. In such an optimization process, the first step is to characterize the printer by printing a number of color patches with known device control values, measuring the colors obtained and generating the errors between them. With these data, a characterization table -- ICC profile is generated, which maps color input to the corresponding output colors for the specified output device. Finally, the color printing system refers to the ICC profile, and then compensates color output at the OS level (Operation System).

ICC-based color management has three inherent disadvantages: Firstly, a thorough device characterization is a relatively expensive process considering the printing and measurement of a large number of color samples. Figure 1.1 demonstrates the Printopen RGB (Red, Green, Blue) Basic 135 test chart, which contains a representative selection of 135 color patches as the important color areas required for color calibration [7]. The measurement of the 135 test chart usually takes up to twenty minutes (data based on the experience with a Gretag Macbeth i1 Spectrophotometer). A Printopen RGB Extended 840 test was also introduced to capture the device characteristics more accurately. Performing such a test dramatically increases the time and labor involved. In addition, the printing error can only be reduced by calibrating the system periodically or as necessary.

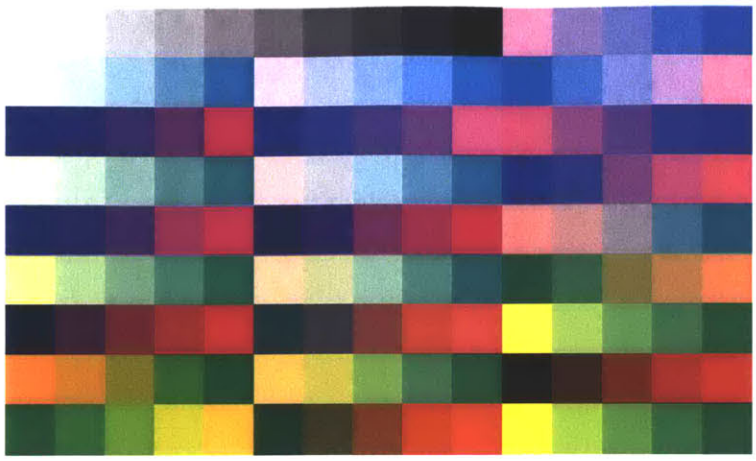


Figure 1.1
The Printopen RGB basic 135 test chart (adopted from *Test Charts and Control Elements User's Guide* [7]) contains a representative selections of 135 color patches and hence the important color areas required for color calibration process.

Secondly, the measurement instruments for device characterization are expensive and usually unavailable to most low-end users. A spectrophotometer, the most precise and complete device for calibrating color printers, measures the spectral reflectance of an object and describes completely a color's essential light feature. But the device is fragile and cost-prohibitive: the ProfileMaker system, including its hardware and software, costs $4490 (quoted from Gretag Macbeth online store, July 2006); a colorimeter breaks light down into its three "standard observer" (RGB) values as read by a light sensor or detector receiving reflected light from a test surface illuminated with RGB LEDs or lamps. It is relatively less expensive, but the measured values are valid only under the perfect standard measurement conditions.

Thirdly, the accuracy of calibration largely depends on interpolation algorithms integrated into OSs. Because of its size limitation, the information in ICC profile is not sufficient to rebuild the whole RGB color space. As Microsoft Windows is not an open source operation system, it is hard to implement and evaluate other interpolation methods under its CMS. Even worse, some vendors disable Windows CMS in their

printer drivers or image processing software. A similar problem exists in other platforms like the Apple Mac OS X.

In summary, ICC-based color management requires huge design efforts and lacks self-adaptation. An alternative with lower cost and better interpolation algorithm becomes increasingly necessary.

1.3. Purpose of Study and Thesis Outline

The goal of this study is to update a RGB printing system in response to systematic changes in print-device characteristics. To be successful, the method must be efficient relative to a full system characterization. In addition, it must reduce overall colorimetric errors without introducing significant new errors in any part of the gamut.

This thesis discusses three major problems in color calibration: modeling the process with feedback control, implementing three major modules, and testing the entire workflow under Windows printing architecture. All these topics will be presented in the following chapters respectively,

Chapter 1 introduces the general background and reviews several industrial solutions. The disadvantages of current CMSs are summarized in terms of the implementation costs and effect of compensation mechanism. An innovative color management method is becoming necessary to replace the current methods.

Chapter 2 begins with the analysis of system variables and basic hypothesis, proposes a new correction method, and briefly introduce the implementation of three control modules. The most important components, compensation controller and printing process, will be discussed in chapters 3 and 4 respectively.

Chapter 3 focuses on the compensation controller module. Since the new method tries to fit printer characteristic curves with limited number of samples, the accuracy of the compensation largely depends on the effect of interpolation methods. Among four approaches evaluated in this chapter, the quadratic spline interpolation is the best one in that it can reduce wriggles and improve curve smoothness with less computational efforts.

Chapter 4 focuses on another key module - printing process. With the characteristic curves obtained from the previous chapter, color images are updated pixel by pixel in image rendering plug-in of Universal Driver. In addition, this chapter gives color fundamental related to the design of new test sheet.

Chapter 5 carries out the test plans and evaluates the results quantitatively and qualitatively. Chapter 6 concludes the project, and heads the future study into three directions.

CHAPTER 2
COLOR CALIBRATION MODEL

2.1. Introduction

Printer calibration is difficult due to the nonlinearity of the printing process, and the wide variety of printing techniques (e.g., lithography, ink jet, dye sublimation, etc.). The major goal of a calibration system is to keep the printing error small for any input in response to the expected parameter changes. Now it can be converted to a fundamental problem in control system design. This following study constructs a theoretical model of the calibration system, and briefly discusses the process of system implementation.

2.2. Variables and Hypothesis

The printer response and consequently, its characterization function may change due to a number of factors. These influences can be divided into two categories as random and systematic variables. Some factors are unpredictable in that they produce random disturbances. Examples include ambient humidity, temperature fluctuation, ink instability, change in toner performance, drum or printhead attributes. In addition, printers also drift daily or even hourly in response to throughput.

Another type of variation happens systematically, like the difference in colorant and print substrate. The substrate can influence the color characteristics of the printer in several ways. Certain substrate properties such as surface roughness can affect chemical interactions between colorants and substrate, in turn affecting the resulting color [8]. Optical properties play an important role in terms of the spectral scattering or diffusion within the substrate, as well as reflectance of the substrate.

Typically, the impact of random factors is minimized through careful design of the printer. The following study is based on a printer that did not drift, that had uniform

spatial and environmental sensitivity, with a stable, predictable set of colorants. With this assumption, a method is proposed and evaluated to recalibrate the printing system in response to systematic changes such as the changes of substrate or colorant.

2.3. Feedback Control Modeling

Control systems can be classified according to the information used to compute the controlling action: if the controller does not need a measurement of the system output, the system is called an open-loop system; whereas in closed-loop, or feedback control system, the controlled output signal is measured and fed back for use in the control computation. Compared with open-loop control, feedback can be used to reduce steady-state error, and the system's sensitivity to parameter variations.

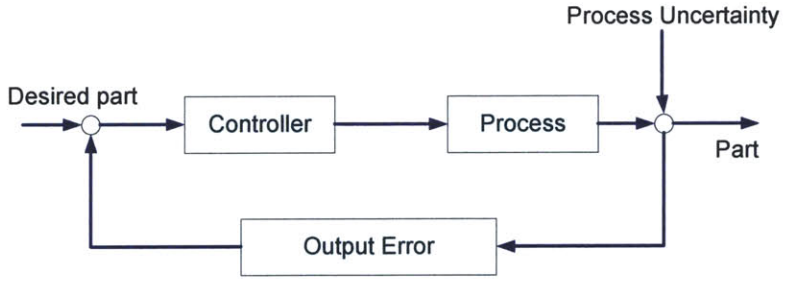


Figure 2.1
The cycle-to-cycle control model is proposed for printer calibration. Three major blocks involved are the output error, controller, and process. The output error between the real and desired output is detected by human eyes or measurement tools like a scanner; the controller draws printer characteristics curves from error sampling, and gives the compensation gain for each color; the process block represents the printing architecture and related APIs in Windows.

CtC control (cycle-to-cycle) is a method for using feedback to improve product quality for processes that are inaccessible within a single processing cycle. Input parameters are changed for the production of a subsequent part based on the difference between the part produced and the target specifications. The system continues modifying its

parameters until the output error is reduced to an acceptable level. A CtC control model is shown in Figure 2.1.

In this case, image sRGB (standard RGB) data are regarded as the input of the control system. Three major modules in the control system are output error collection, compensation controller, and printing process. Using printer characteristic curves (or error curves), the controller modifies the system input, and then feeds it into the printing process. As most colors in a printing system come from a combination of primary colors, changing those primary colors to improve the output of one color may result in the degradation of another color in the printing system. Because the complexity of color printing makes control target almost inaccessible via a single trial, CtC control is highly recommended in the calibration of printing system.

In the above system, the printout variance is illustrated as,

$$\Delta Y = \frac{\partial Y}{\partial \alpha}\Delta\alpha + \frac{\partial Y}{\partial u}\Delta u \tag{2.1}$$

where

$\frac{\partial Y}{\partial \alpha}$ is the disturbance sensitivity,

$\Delta\alpha$ represents the disturbances in the control system,

$\frac{\partial Y}{\partial u}$ is called control sensitivity or gain, and Δu gives the control input.

The design objective, as the output strictly equals to the input,

$$Y(s) = R(s) \tag{2.2}$$

can be obtained by

$$\frac{\partial Y}{\partial u}\Delta u = -\frac{\partial Y}{\partial \alpha}\Delta\alpha \tag{2.3}$$

From above analysis, the objective of the control system is to reduce the output error. One way to do this is to design a controller which always satisfies equation (2.3). In the context of controller design, output errors at control points need to be detected either

manually via human eyes or automatically with measurement tools like a scanner or spectrophotometer. These data are collected for fitting the characteristic curves, which represent the output accuracy of a printer over the whole color space. Then the controller looks up these curves to figure out the offset for each color, and compensate the image in Windows printing process. As the calibration is carried on for several cycles, most color output will become approximate to the input.

2.4. Brief Discussion of Model Implementation

The previous section describes the system in the view of a feedback control. The work flow of the methods is shown in Figure 2.2. The implementation modules can be categorized into two parts: the construction of a look-up table with a characterization data set, and color compensation in conjunction with Windows printing architecture.

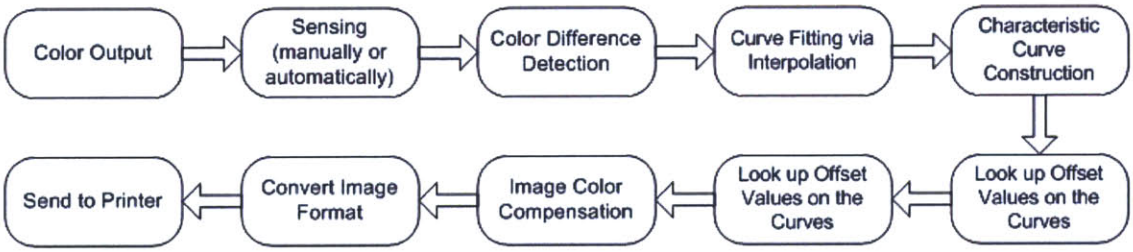


Figure 2.2
The flowchart of an innovative color management method

In the first step, a set of error is collected by printing a number of color patches with known RGB values, measuring the colors obtained, and comparing with a standard reference sheet. The rest of the characteristics data is generated from the interpolation of the error sampling. The characterization function is often implemented as a multidimensional look-up table in the form of an array. The array contains 256 entities as $[u_R\ u_G\ u_B]_{index}$ where the index ranges from 0 to 255.

In the second step, the characterization function determines the device control values required to produce a color specified in device independent color space. The final color correction is implemented through Windows GDI (Graphical Device Interface) programming. The output is improved in a device independent color space (e.g., CIELAB).

The advanced topics related to the two aspects of the implementation are discussed in later sections. In chapter 3, three interpolation methods are proposed and evaluated in terms of the algorithm efficacy and efficiency. Chapter 4 gives the detailed information on how to correct colors under the frame of Windows printing architecture. It also presents color fundamentals for a better understand of the color corrective mechanism.

2.5. Summary

This chapter begins with the analysis of key factors that influence printing stability, and identifies the basic assumptions for the following study. Consequently, the feedback control model is proposed for color calibration, followed by the procedure of constructing a controller in the control system. It also brings up two fundamental problems as interpolation algorithm and color compensation implementation, which are discussed in chapters 3 and 4 separately.

CHAPTER 3
PRINTER CHARACTERISTIC CURVE

3.1. Introduction

With limited number of color patches on the test chart, the sampling only covers a small proportion of 24bit colors ($256 \times 256 \times 256$) in RGB color space. In other words, the sampling might not be sufficient to fully describe a printer's behavior. Thus any value between the control points needs to be estimated by fitting printer characteristic curves via numerical methods.

The accuracy of the compensation largely depends on the interpolation method in terms of its capacity in reducing wriggles and improving curve smoothness. In addition, the interpolation algorithm should also be suitable for computer implementation. It should also be reasonably sensitive to the primary sources of problems, namely roundoff errors in the computations.

This chapter will focus on the analysis of interpolation methods, including polynomial and neural network models. The input of the interpolation function is the output error measured at each control point, which can be detected either manually via human eyes or automatically with measurement tools like a scanner or spectrophotometer. And the output will be the error curves constructed by fitting those control points via interpolation. Appendix A contains the Matlab program for all the interpolation methods in the following section.

3.2. Training Data

In this chapter, the experiments in the following sections are based on a sample set labeled as "*Control Point Data*" part, Appendix A.2. The following is abstracted from the appendix.

```
% Control Point Data
% x0  color index of the control point
% y0  color output error
x0  = [0 3 5 7 9 11 12 13 14 15];
y0  = [0 1.2 1.7 2.0 2.1 2.0 1.8 1.2 1.0 1.6];
```

The sample [x0, y0], which is collected from a HP® 9500PCL printer on June 8th, 2006, has two entities as color index and output error. Since the printer is a typical commercial printer with stable performance, the data collected from it is used to represent a printer that did not drift and had uniform environmental sensitivity as mentioned in basic hypothesis, section 2.2.

This sample has a serial of control points, where the color index ranges from 0 to 15, and the output error ranges from 0 to 2.1 correspondingly. These points reflect the low-dynamic character of a typical trend of printer error. Typically, the output errors of a commercial printer only vary smoothly within a limited range because the printers are well designed to reduce sudden and large-range errors [9]. It is why the data collected from the HP printer can be used in the following experiments.

It is noticeable that there are two training sets for the neural network methods in section 3.5. The first set is exactly the same as "*Control Point Data*" part, Appendix A.2. It provides a reference to compare the interpolation effect to previous methods. The second set is given in "New Training Set" section, Appendix A.4. The training set is abstracted as follows,

```
% New Training Set
% p   color index
% t   (output error)/15
```

```
p = [0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105
110 115 120 125 130 135 140 145 150 155 160 165 170 175 180 185 190 195
200 205 210 215 220 225 230 235 240 245 250 255];

t = [0.000 0.267 0.400 0.000 -0.067 -0.800 -0.467 0.200 -0.067 -0.400 -
0.467 -0.333 -0.333 0.000 0.267 0.000 -0.133 -0.067 0.267 0.333 0.000
0.267 0.133 -0.333 -0.133 0.333 0.933 0.400 0.667 0.067 0.133 -0.067 -
0.533 0.133 0.000 -0.200 -0.733 -0.333 0.267 0.467 0.800 0.400 -0.200
0.067 0.333 -0.267 -0.067 -0.467 -0.800 -0.733 -0.067 0.000];
```

The second set is different from the previous one in terms of the frequency of error fluctuation. In order to test the neural network's capacity in representing a printer with high dynamic errors, the sample with 50 pairs of RGB error was collected from an Epson C43 inkjet printer, which has lousy output caused by poor maintenance level. The indexes are color inputs at control points, which range from 0 to 255 with a step of 5.

3.3. General Interpolation Methods

The interpolation idea comes from constructing the convex hull with a set of points. In general, any interpolation method can be expressed in the form of

$$f(t) = \sum_{i=0}^{dof} a_i p_i(t)$$

where

dof is the polynomial degree of freedom,

a_i are the coefficients, and p_i are the basic polynomials as replaceable elements in different interpolation methods.

More specifically, the method uses n control points $(c_i)_{i=1}^{n}$ to construct a spline as in

$$f(t) = \sum_{i=0}^{n} c_i B_{i,d}(t) \tag{3.1}$$

This is often referred to as *Schoenberg's variation diminishing spline approximation* [10].

Given the Lagrange polynomials of degree d $\{l_{i,d}\}_{i=1}^{d}$, **Lagrange interpolation** is expanded in terms of formula (3.1) as,

$$f_{0,d}(t) = \sum_{i=0}^{d} c_i l_{i,d}(t) = c_0 l_{0,d}(t) + c_1 l_{1,d}(t) + \ldots + c_d l_{d,d}(t) \qquad (3.2)$$

where

$$l_{i,d}(t) = \prod_{0 \le j \le d, j \ne i} \frac{t - t_j}{t_i - t_j}$$

In practice, the computation is referred to as **NA method (Neville-Aitken)** [11]. Given the $d+1$ control points $(c_i)_{i=0}^{d}$ with $d+1$ strictly increasing parameter values $(t_i)_{i=0}^{d}$, there are

$$f_{0,d}(t) = c_i \qquad (i = 0, 1, \ldots, d)$$

and

$$f_{i,r}(t) = \frac{t_{i+r} - t}{t_{i+r} - t_i} f_{i,r-1}(t) + \frac{t - t_i}{t_{i+r} - t_i} f_{i+1,r-1}(t)$$

Figure 3.1 demonstrates an example of using the above method to compute $f_{0,3}(t)$,

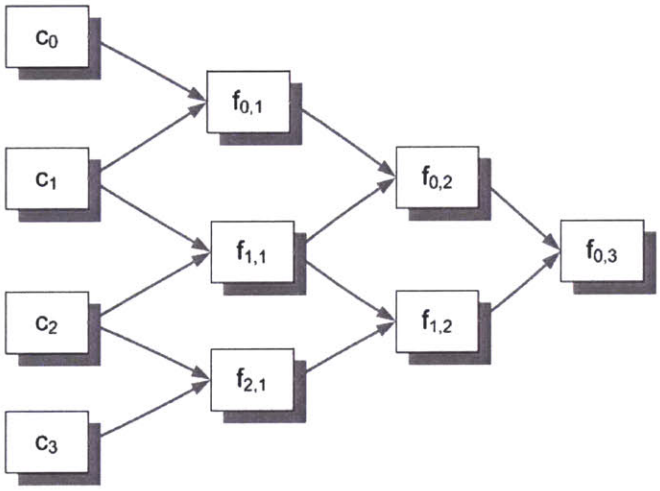


Figure 3.1
Computing a polynomial curve with the NA method
(The computation starts from c_i and ends by $f_{i,r}$, following the pointer directions.)

The interpolation result with ten control points is shown in Figure 3.2. The wriggle between the point 0 and 1 indicates that Lagrange interpolation introduces obvious local errors in this scenario.

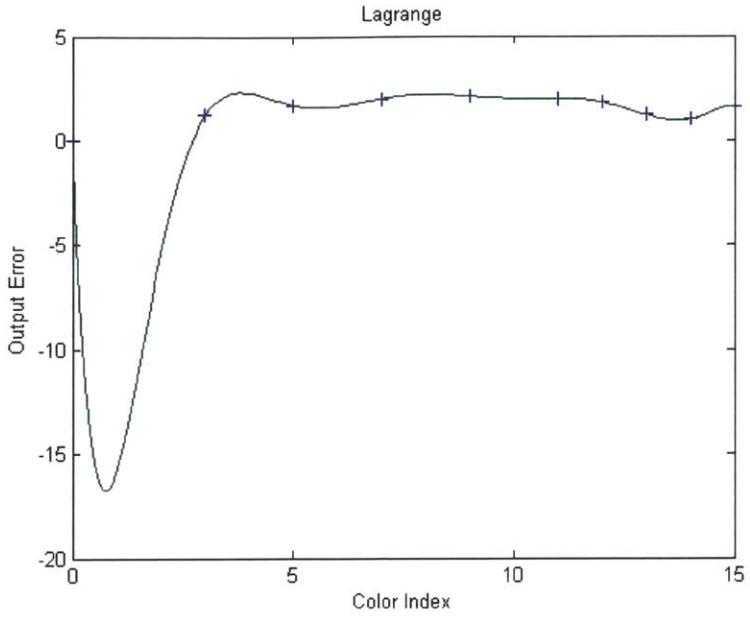


Figure 3.2
The result of Lagrange interpolation
training sample size: 10

Piecewise linear interpolation connects each pair of neighboring points with a straight line. More specifically, n numbers $(t_i)_{i=2}^{n+1}$ with $t_i < t_{i+1}$ for i=2, 3, ..., n define the curve f by

$$f(t) = \begin{cases} p(t \mid c_1, c_2; t_2, t_3) & t \in [t_2, t_3) \\ p(t \mid c_2, c_3; t_3, t_4) & t \in [t_3, t_4) \\ \cdots & \cdots \\ p(t \mid c_{n-1}, c_n; t_n, t_{n+1}) & t \in [t_n, t_{n+1}] \end{cases} \tag{3.3}$$

The points $(C_i)_{i=1}^{n}$ are called control points of the curve; while the parameters $t = (t_i)_{i=2}^{n+1}$, which give the value of t at the control points, are referred to as the knots of the curve. With the piecewise constant function $B_{i,0}(t)$, $f(t)$ can also be written as,

$$f(t) = \sum_{i=2}^{n} p_{i,1}(t) B_{i,0}(t) \tag{3.4}$$

where

$$p_{i,1}(t) = p(t \mid c_{i-1}, c_i; t_i, t_{i+1}) = \frac{t_{i+1} - t}{t_{i+1} - t_i} c_{i-1} + \frac{t - t_i}{t_{i+1} - t_i} c_i$$

$$B_{i,0}(t) = \begin{cases} 1 & t_i \le t \le t_{i+1} \\ 0 & otherwise \end{cases}$$

Using the above formulas, this study interpolates ten control points with the piecewise linear method. The experimental result in Matlab is shown as Figure 3.3. It turns out that sudden changes are at the control points between neighboring pieces. In other words, the polynomial curve has discontinuous derivatives at control points.

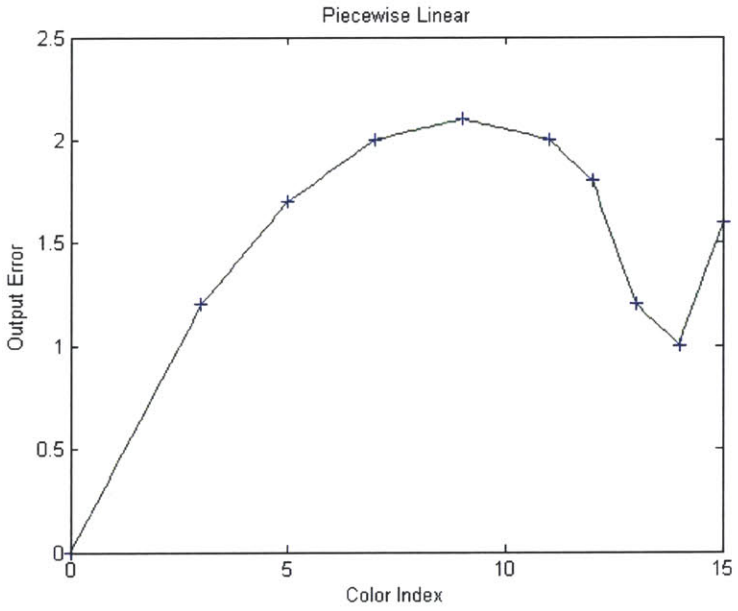


Figure 3.3
The interpolation result of piecewise linear method
training sample size: 10

3.4. Quadratic Spline Method

To solve the wriggle and discontinuity problems in previous methods, the study considers spline curves of higher degrees. Given $d+1$ points $(c_{j,d})_{j=i-d}^{i}$ and $2d$ knots $(t_j)_{j=i-d+1}^{i+d}$, a quadratic spline $p_{i,d}$ with a double knot is determined by the following computations.

First set $p_{i,0}(t) = c_j$ for $j = i-d, i-d+1, ..., i$, and then compute

$$p_{ij}(t) = \frac{t_{j+d-r+1} - t}{t_{j+d-r+1} - t_j} p_{j-1,r-1}(t) + \frac{t - t_j}{t_{j+d-r+1} - t_j} p_{j,r-1}(t) \tag{3.5}$$

for $j = i-d+r, ..., i$ and $r = 1, 2, ..., d$.

A spline curve of degree d with n control points $(c_i)_{i=1}^{n}$ and $n+d-1$ knots $(t_i)_{i=2}^{n+d}$ is given by

$$f(t) = \begin{cases} p_{d+1,d}(t) & t \in [t_{d+1}, t_{d+2}] \\ p_{d+2,d}(t) & t \in [t_{d+2}, t_{d+3}] \\ \cdots & \cdots \\ p_{d+n,d}(t) & t \in [t_n, t_{n+1}] \end{cases} \tag{3.6}$$

for $i = d+1, ..., n$, and $t_i < t_{i+1}$

Similar to formula (3.4), $f(t)$ can be expressed in terms of the piecewise constant function given by formula (3.6),

$$f(t) = \sum_{i=d+1}^{n} p_{i,d}(t) B_{i,0}(t) \tag{3.7}$$

where $$B_{i,0}(t) = \begin{cases} 1 & t_i \le t \le t_{i+1} \\ 0 & otherwise \end{cases}$$

Matlab experiments with this model prove that spline curves of degree d have continuous derivatives up. As shown in Figure 3.4, higher order spline interpolation is

relatively better than Lagrange and piecewise linear interpolation in terms of wriggles reduction and improving curve smoothness.

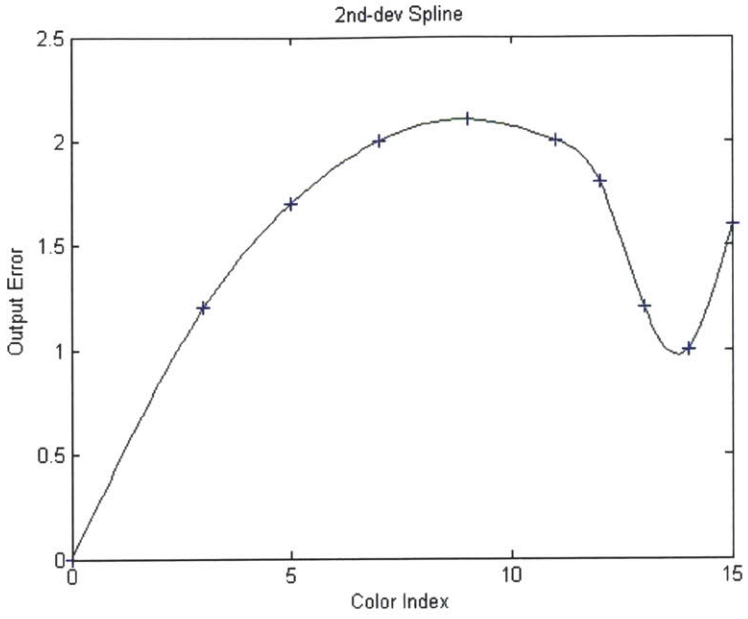


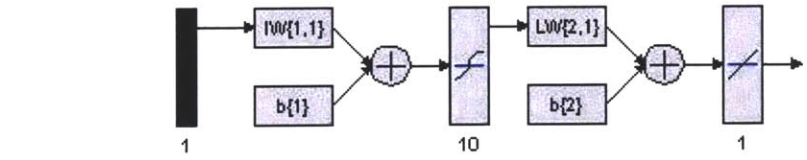
Figure 3.4
The interpolation with 2nd-dev spline curve
training sample size: 10

3.5. Neural Network in Geometrical Error Compensation

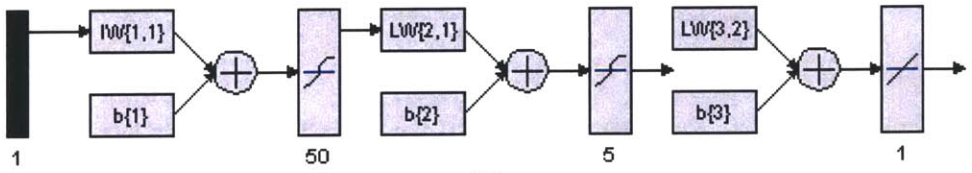
In practice, the printing system is often characterized by highly complex structures and dynamic behavior caused by nonlinearities and the influence of unpredictable disturbances. Thus, the previous approaches are likely to be less accurate than a full system recharacterization in that they do not consider colorant interactions and dynamics. The following study discusses the neural network interpolation and its capability in describing printer characteristics with RGB vectors.

A neural network is composed of simple elements operating in parallel, and is inspired by a biological nervous system. As in nature, the network function is determined by the connections between elements. A neural network can be trained by adjusting the values of connections, or weights, between these elements. The BPNN (backpropagation neural network) was created by generalizing the Widrow-Hoff learning rule to multilayer networks and nonlinear differentiable transfer functions. The term backpropagation refers to the manner in which the gradient is computed.

Typically, a network with biases, a sigmoid layer, and a linear output layer is capable of approximating any function with a finite number of discontinuities. The BPNN model (Figure 3.5) proposed in this study consists of three layers as input-hidden-output, and uses tan-sigmoid, purelin functions as transfer functions. Input vectors and the corresponding target vectors are used to train the network until it can approximate printer characteristics functions.



(a)



(b)

Figure 3.5
The BPNN network structure
(a) simple BPNN trained with original data set
(b) multi-layer BPNN trained with new training set

Firstly, a single BPNN with 10 neural in the hidden layer (Figure 3.5 (a)) was trained with the same sample set in previous sections. The mean squared error goal was set as 10^{-6}, and the training process reached the convergence stage after 20 epochs. Figure 3.6 presents the interpolation result from this single BPNN structure.

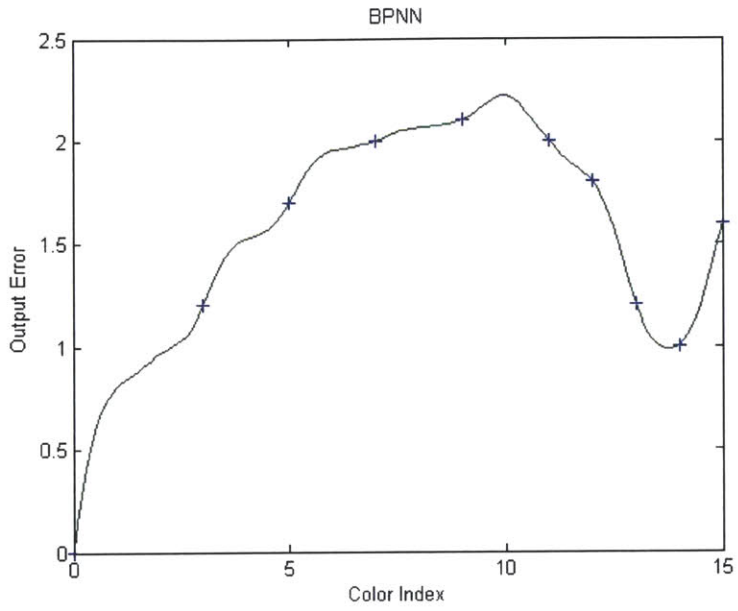


Figure 3.6
The BPNN interpolation with the original training set
training sample size: 10

It is noted that the result from the BPNN might not look as good as previous methods. This can be explained from two aspects: the selection of training data and the complexity of network structure. Generally speaking, the better result can be achieved from a more complex network trained by a lager and well-designed sample. Thus, the test was carried on to the following scenario.

In this experiment, a multi-layer BPNN (Figure 3.5(b)) was trained by *"New Training Set"*(in section 3.2) with the mean squared error goal of 10^{-2}. The training reached its

convergence stage after 92 epochs (Figure 3.7). It took about 90 seconds on a 1.6GHz Pentium M computer with 1 Gbyte memory.

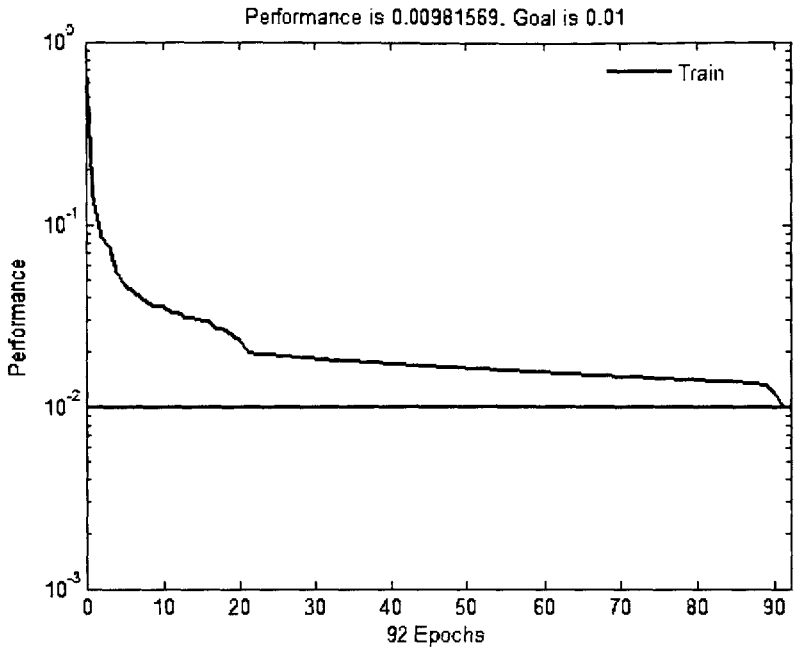


Figure 3.7
The convergence of BPNN training

Figure 3.8 shows the experimental result: the pink dash line illustrates the real printing errors in the R channel, while the blue line is the interpolation result from the BPNN. The result here is reasonable, because the test set error and validation set error have similar values, and there is no significant mismatch between the two sets.

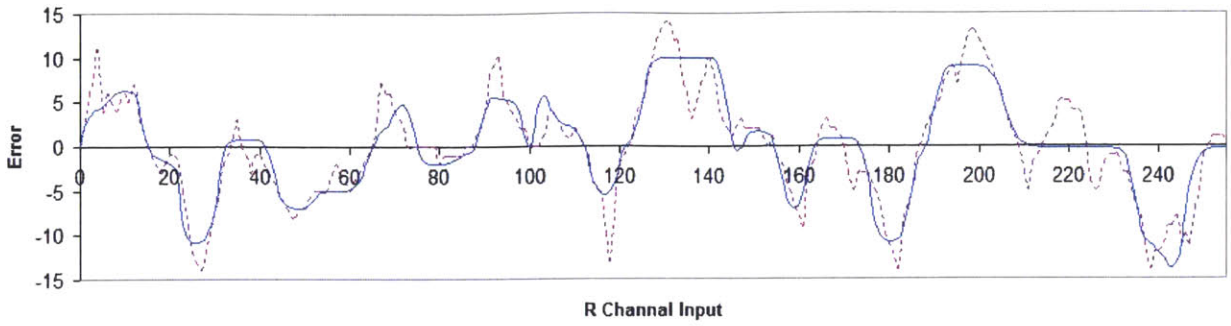


Figure 3.8
The BPNN interpolation with a larger training set
training sample size: 50

A tradeoff exists between numerical and neural network approaches: the quadratic spline interpolation is capable of constructing a smooth curve for each individual channel (R/G/B); as it is trained using RGB vectors, the BPNN method models the interaction between colorants and shows good response to high colorant dynamics over the entire color space. In other words, the BPNN method is good at solving multi-dimensional interpolation problem. Unfortunately, BPNN is relatively expensive considering the computational time, and its advantage over numerical methods is not significant since most printing errors are caused by changes in substrate and colorant.

3.6. Summary

To construct a controller for a printing process, an interpolation method would need to be implemented with a discrete set of known data points. Several types of interpolation methods are evaluated in this chapter, including numerical methods and a neural network method. The most successful of these methods is the quadratic spline interpolation, which is able to remove most errors without introducing significant new errors. A significant reduction in error can be realized by incorporating this technique into the color printing system.

It should be noted that this conclusion is only valid in certain scenario. Generally speaking, the higher order the polynomial is, the better interpolation result it can get. Unfortunately, as the polynomial order increases, the computational efforts involved in interpolation increases consequently.

In this specific case, interpolation methods are used to fit printer characteristic curve. Since most commercial printers don't have sharp peaks with high dynamic range along the error curves, the quadratic spline interpolation is capable to perform the task. It is adopted in this project because of the tradeoff between the interpolation effect and computational cost.

In addition, the study also gives a BPNN solution for the printer with high dynamic error fluctuation. However, this extreme case, or so called high dynamic error range, seldom happens on a commercial printer in which the random error is minimized through careful design.

CHAPTER 4
COLOR PRINTING SYSTEM

4.1. Introduction

The study now switches to two problems left in model implementation: one is how to design a color test chart containing key control points; the other one is how to update image color pixel by pixel in the Windows printing system.

This chapter focuses on these model implementation topics. Detailed information of Windows printing architecture can be refereed to Microsoft DDK and MSDN documents [12-13]. Please refer to Curtis N. Vanderpuije for more details about the workflow and image modification through the Windows GDI [14].

4.2. Color Test Chart

The description of color is subjective since it is based on the perception of the human visual system. Color results from sensation produced on the eye by rays of light reflected or emitted from an object. Through a prism, white light, which consists of radiation at all wavelengths of the visible spectrum, can be separated into red, orange, yellow, green, blue, indigo, and violet. Out of this understanding comes the concept of primary colors.

Primary colors, when combined in different ways, result in a broad range of colors [15]. Many types of primary color sets exist. The most principal sets are additive and subtractive ones (Figure 4.1). Projected together as beams of colored light, the additive primaries will mix or overlap to produce other colors. The additive primaries are used for a computer monitor or television screen. On the other hand, the subtractive primaries (cyan, magenta, and yellow) are used to create color separations in photography and

printing. In subtractive color mixing, pigments such as ink or paint absorb or subtract all the colors of the spectrum except the color that the pigment reflects to the eye.

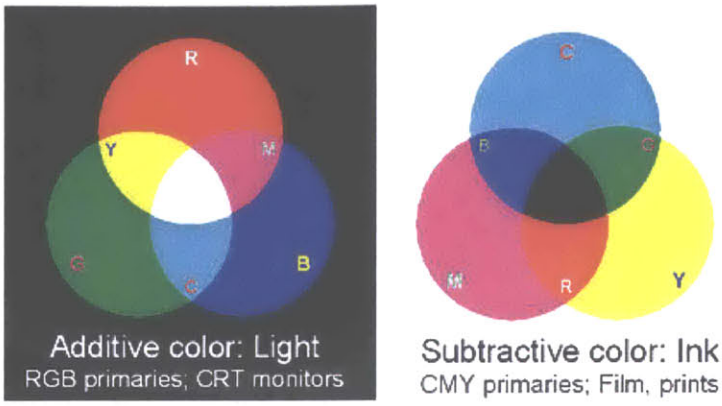


Figure 4.1
Images of additive and subtractive primaries (adopted from *Real World Color Management* [14]) show how the primaries are combined in different ways, and result in a broad range of colors. The two methods have their own applications in CRT monitors, photography and printing respectively.

Color models, or color spaces, are methods of organizing the set of possible human color perceptions in a systematic way. Colors or their properties are expressed numerically in these models. A color space is a 3-D geometric representation of the colors that can be produced using a certain color model. Color models can be divided into two main categories, perceptually-based and display-based (device-dependent).

Device-dependent models (or display-based models), such as RGB, are used to create millions of colors on a computer monitor or television screen by combining different values of red, green, and blue. RGB is an example of a device-dependent color model because each device emits a slightly different shade and intensity of red, green and blue light. In the RGB system, each color component of a pixel is measured by a number from 0 to 255, for a total of 256. The RGB model is also used in color scanners and color photography. However, this model does not work well in the printing process.

The CMY (cyan, magenta, yellow) model is a subtractive color model that complements the additive model. CMY model are used in printing because the process is based on light reflecting from colors, such as those contained in a printed image on paper. However, the combination of CMY in full values does not produce a pure black color. Therefore, black (K) must be added to the separations, resulting in the CMYK color model. The CMYK model forms the basis for process printing.

Appendix B shows the new chart designed to perform printer characterization in this study. The new chart consists of both CMYK and RGB bars, ranging color density from highlight, midtone to shadow in each row. One unique feature over conventional charts is the key line (line 5) in which black is registered as a combination of CMY rather than print directly using black ink as in line 4. Comparing the printout with a standard chart, the calibration system can identify the accuracy of the CMYK printing system on RGB image reproduction.

4.3. Implementation under Windows Printing Architecture

The Microsoft® Windows 2000 and later printing architecture consists of a print spooler and a set of printer drivers. Spooler and driver components are designed to be replaceable, so hardware vendors can easily add support for their hardware. The driver development usually requires the creation of data structures and methods for use with the UNIDRV (Universal Printer Driver).

The UNIDRV is the Microsoft Corporation's standard printer driver for non-Postscript printers. The UNIDRV components consist of DLLs, plus text and binary data files, as illustrated in Figure 4.2.

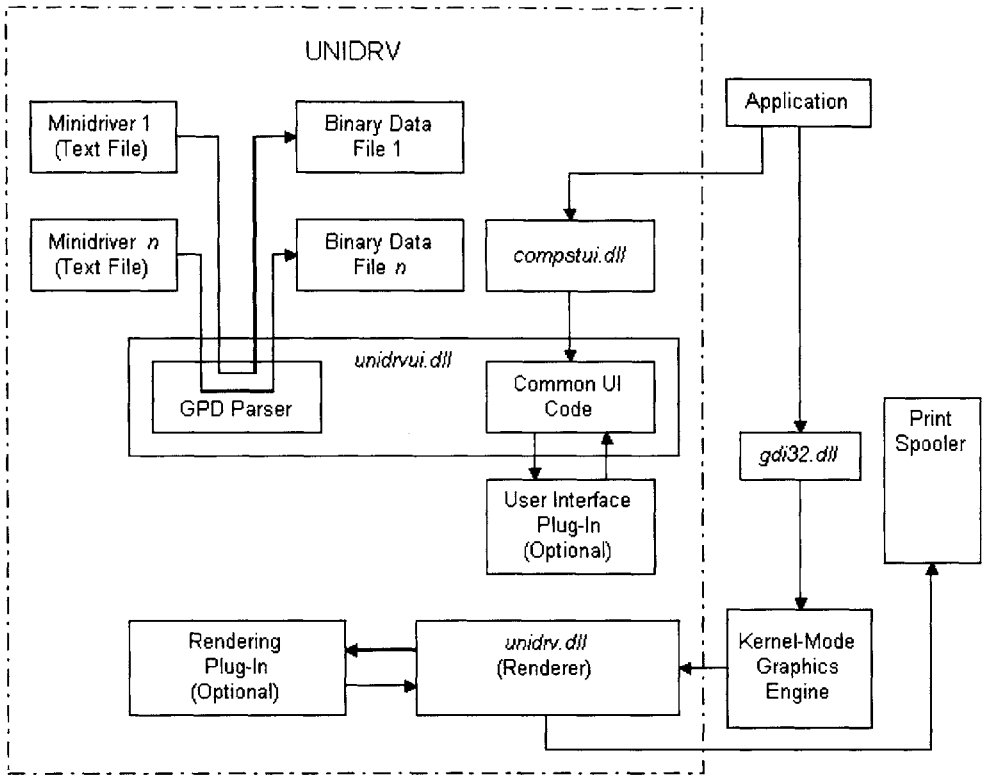


Figure 4.2
The architecture of UNIDRV (adopted from Windows DDK document [12]) (Two major components related to this project are User Interface and Rendering Plug-ins.)

Adding color control in printer drivers requires plug-ins development for two core UNIDRV modules -- User Interface and Rendering. Both plug-ins are implemented as COM (Component Object Model) in-process server. An in-process server is basically a DLL (Dynamic Link Library) that exports a very specific set of entry points, the most important one being DllGetClassObject. When called, this function returns a pointer to a COM class factory. The job of the class factory component is to manufacture all of the components that a given COM server supports. In the case of a UNIDRV plug-in, that means either the UI (User-interface) plug-in component or the rendering plug-in component.

The UI plug-in fine-tunes the user interface through the UNIDRVUI.DLL. With these Device Contexts, IPrintOemUI::DevicePropertySheets method is employed to create a color calibration tab as "Device Property" sheet during runtime for a printer. UI development also involves the implementation of color characteristic curves, via which the end users will be able to manipulate multi control points. Alternatively, the UI allows users to key in the compensator values for RGB channels (shown in Figure 4.3). This study also integrates quadratic spline interpolation, which works out the printer characteristic curve with the control points obtained from the UI.

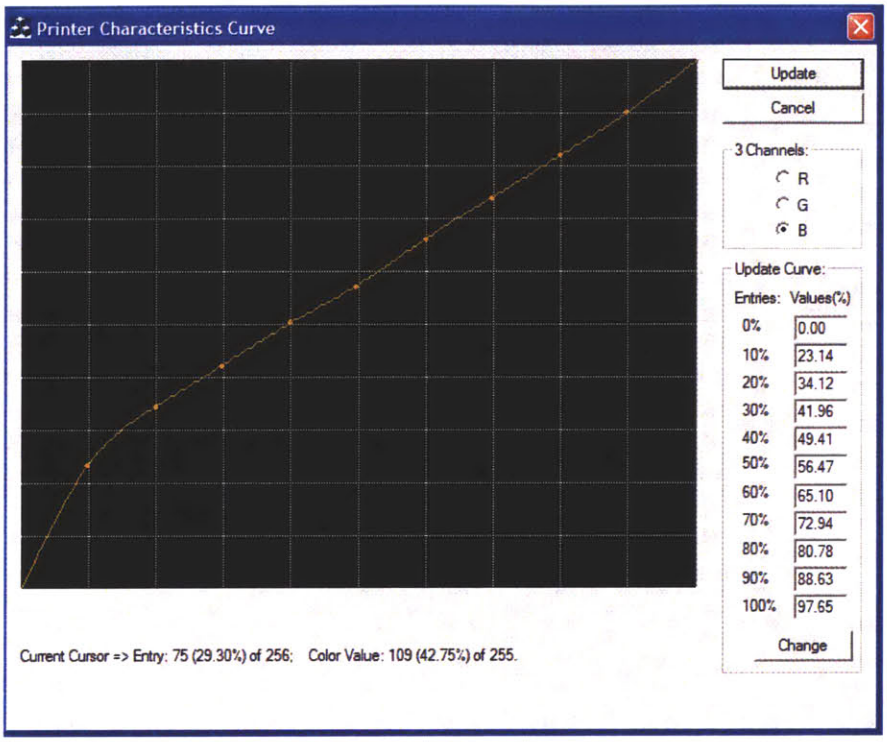


Figure 4.3
The control window of printer characteristic curve (The user can manipulate the curve by moving the eleven control points vertically.)

An image rendering plug-in, which performs customized color management, is implemented in order to fine-tune the graphic output of the core UNIDRV rendering

engine, UNIDRV.DLL. The UNIDRV provides two major imaging pipeline interfaces, IPrintOemUni::ImageProcessing and IPrintOemUni::FilterGraphyics.

Although the purpose for both interfaces is to deliver raster image data, there is a significant difference between them: The IPrintOemUni::ImageProcessing method delivers user definable multi-bit RGB raw image data (24 bit RGB for best sampling). As opposed to ImageProcessing interface, the IPrintOemUni::FilterGraphyics method delivers post processed raster image data. In other words, the data delivered by it are color converted and halftoned. A major gain of using this interface is a performance improvement due to the absence of color management. The obvious drawback is that the quality of output can not be controlled.

In this study, the IPrintOemUni::ImageProcessing method is used with UNIDRV-supported printers to modify the image bitmap data. The method modifies image color pixel by calling Color-Palette API (Application Programming Interface) and returns the updated bitmap to UNIDRV. The prototype also has the capability to deal with several image types including jpg, gif, tiff, etc.

4.4. Summary

This chapter presents color fundamentals related to the Windows printing system and discusses the new test chart. The advantage of this new chart is that it shows the color-reproduction accuracy of CMYK colorants on RGB image. With this chart, the user can calibrate the printing system by modifying the output of primary colors from highlight to shadow areas.

The study also demonstrates the architecture of the Windows printing systems and the UNIDRV driver. A printer characteristic curve window and the interpolation algorithm from chapter 3 are implemented by writing a UI plug-in for UNIDRV. A user can give

the value of control points by manipulating the printer characteristic curve. The image compensation function is integrated into the rendering plug-in.

CHAPTER 5
EXPERIMENTS AND DISCUSSION

5.1. Introduction

In order to test the effect of the printer calibration system, a series of tests were carried out on different printers and substrates. This chapter discusses the design of the tests, and evaluates the compensated output quantitatively and qualitatively.

5.2. Test Environment and Procedure

The test requirement are listed as follows,

1) a standard calibration chart
2) a commercial color printer with relatively stable output

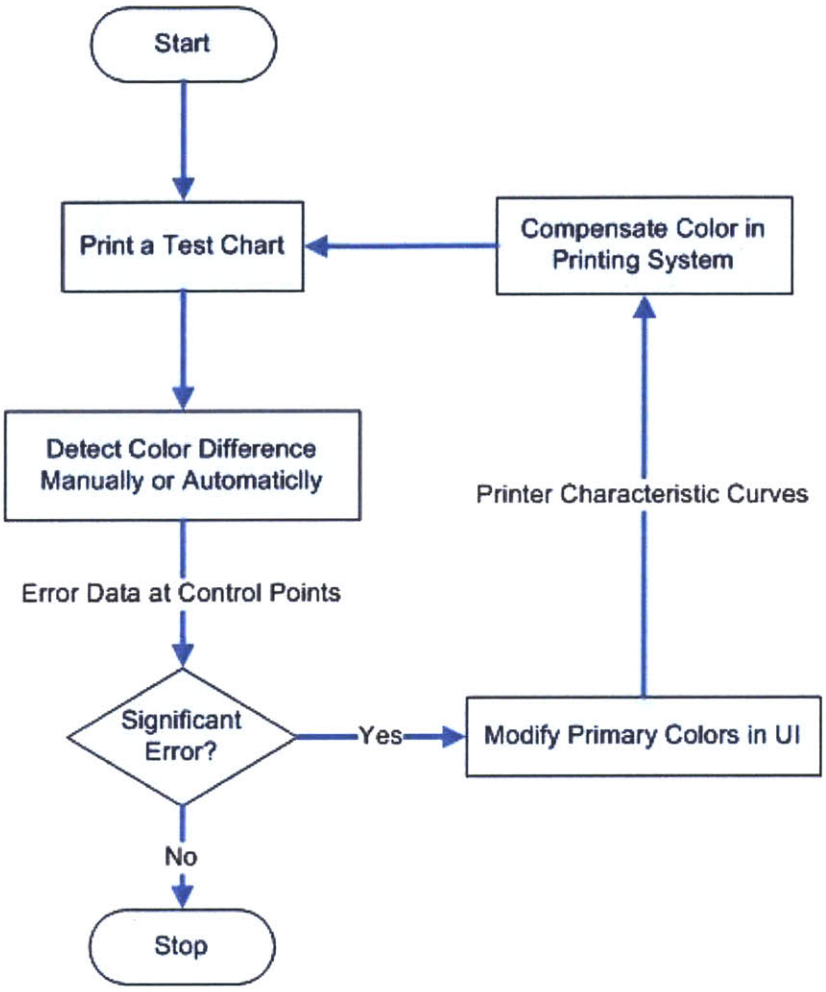


Figure 5.1
The test flowchart including the steps of test sheet printing, output error detection, color modification and color compensation

The test shown in Figure 5.1 goes through the following steps,

1) Print a test sheet from the Windows system

2) Compare the printout against the standard test sheet: the test sheet is designed to fit behind the standard chart window cutouts. Usually, the first test sheet would reveal a set of colors differing from the standard one, as shown in Figure 5.2. The

comparison between the printed copy and the master copy can be done either manually using human eyes or automatically with scanner measurement data.

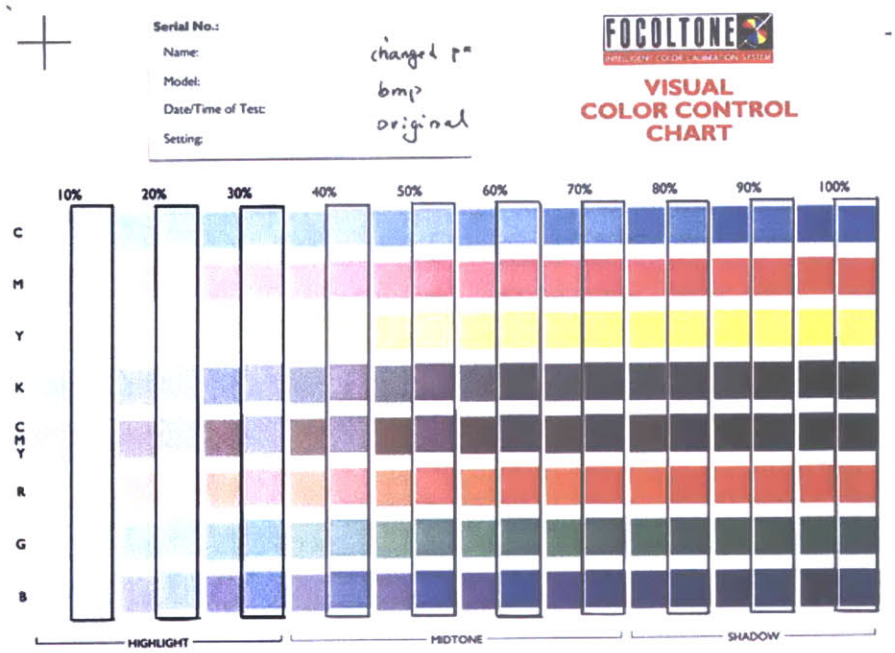


Figure 5.2

Comparing a printout with the standard calibration chart (The test sheet was designed based on Kikuze© test chart for CMYK color calibration. Through the cutouts - the black rectangles on the standard sheet, users can compare the color bars and tell the differences between the printout and the standard sheet.)

Currently the software prototype requires the user to perform the comparison manually. The user shifts two copies horizontally on the observed difference between the printout and the standard sheet, and decides the magnitude of adjustment required for the control points. For example, if the user notices that the 50% red patch on the printout looks the quite similar to 60% red on the standard sheet, he will know the red at 50% control point should be reduced by 10%.

Introducing a quantitative measurement tool can improve the comparison accuracy.

The software module will average several measurements of each color patch, and update the LUT accordingly.

For two reasons, the current prototype doesn't come with the automatic module. Firstly, the project only targets at low-end users who prefer a fast correction with their eyes rather than extreme color accuracy. Secondly, in order to use the data collected from those devices, the system needs an additional module, which is not implemented in the current prototype. This module will transfer the raw data, either delta E value from a spectrophotometer or printer RGB value from a scanner, into the image RGB color space. Only in this way, the data collected from the printout can be used to compensate an image. It should be clarified that transformation from one color space to another is a complex problem in that no single transfer function is available now. The neural network methods are usually used in the color space transformation. Future research will be carried on for the correction module that can use data automatically collected from measurement tools.

3) Modify primary colors from the UI: once the color differences are detected, the printer characteristic curve is updated by manipulating the curve of each channel from the UI.

   It should be noticed that this step only exists in manual mode. Table 5.2 in later part of this section summarizes the basic rules for translating the differences between the printed copy and the master copy to control information. When it comes to the automatic mode, the user will never see this UI. The quantitative measurements can be regarded as control point values directly, and the LUT updating will also be done automatically.

4) Compensate color output via the modification of image: based on the characteristic curve gained from the color analysis, the compensation engine automatically calculates the adjustments required and compensates the color data pixel by pixel.

To expound, the compensation engine first receives printer characteristic curves from step 3), and builds a LUT (look-up table) like Table 5.1. This table, converted from the curves, contains 256 items which represent 0-255 color stages. Each row has 4 entities relating color input to R/G/B channels outputs. Since the image is stored in computer system as a RGB array, the engine extracts RGB information for every pixel, and looks up the expected input of that pixel for the printer. For example, assuming the image has a pixel with RGB values as 3/5/4, according to Table 5.1, the RGB value sent to the printer should be 2/1/2. This process compensates every pixel of the image by offsetting the printer input, and the final rendered image is saved as a bitmap file.

The image after the compensation usually does not look that accurate on the screen, but once it is printed on that specific printer it was calibrated for, the output turns out great. The detailed implementation of updating the image pixels and getting involved in the Windows printing system was covered in section 4.3.

Table 5.1 Data Structure of the compensation LUT

Input	Output		
Color Index	R Channel	G Channel	B Channel
0	1	2	1
1	2	5	2
2	3	3	4
...	...	...	...
255	255	254	255

5) Reprint the test sheet to verify the improved color results, and re-run the experiment if necessary.

The following rules are used as guidelines for updating the printer characteristic curves:

1) The change in RGB space can result from the change in CMYK output. The transformation is qualitatively described by Table 5.2,

Table 5.2 RGB/CMY transformation

CMY decrease ↓	CMY increase ↑
C (highlight) ↓ = R curve (shadow) ↑	C (highlight) ↑ = R curve (shadow) ↓
C (shadow) ↓ = R curve (highlight) ↑	C (shadow) ↑ = R curve (highlight) ↓
M (highlight) ↓ = G curve (shadow) ↑	M (highlight) ↑ = G curve (shadow) ↓
M (shadow) ↓ = G curve (highlight) ↑	M (shadow) ↑ = G curve (highlight) ↓
Y (highlight) ↓ = B curve (shadow) ↑	Y (highlight) ↑ = B curve (shadow) ↓
Y (shadow) ↓ = B curve (highlight) ↑	Y (shadow) ↑ = B curve (highlight) ↓

For example, a desire to decrease the cyan shadow translates to an increase in the red highlight region due to the inverse relationship.

2) Updating the characteristic curve of a primary color affect gamma, brightness and contrast of that particular color. If people prefer to work with these concepts, Figure 5.3 can guide them in changing RGB curves.

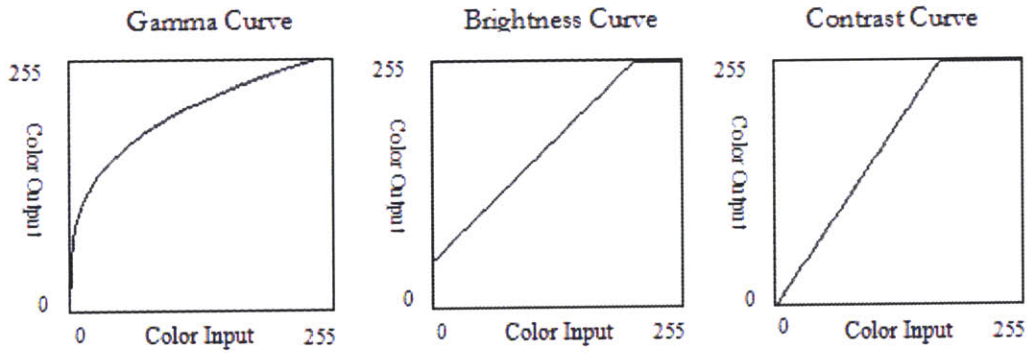


Figure 5.3
The effect on a color curve by changing its Gamma, Brightness and Contrast

5.3. Analysis of Measurement and Affect on Images

Table 5.3 gives the test plan under the combination of different printers and substrates. To verify the calibration effect, a Gretag Macbethi1 Spectrophotometer was used to detect the output errors of 50 color patches (the validation set) on the final printout. This instrument gives delta E values, which quantitatively illustrate the differences between the printout and the standard chart.

Table 5.3 Test Scenarios with Different Configuration

Test	Printer	Substrate	Cycles
A	HP® 9500PCL	APP® 70g normal paper	2
B	Canon® LASERSHOT LBP5200	APP® 75g acid-free paper	4
Note: in both cases, the color errors is detected manually with human eyes			

It should be noted that the test went through several cycles until output errors can not be detected by human eyes. The cycle number in Table 5.2 is the number of trials to achieve the best results under particular test configuration. For example, the color expert in Kikuze company can not tell the difference between the printed copy and master copy after cycle 2 in test A. As the test was carried on with one more trial, the result got worse because of over-compensation. So the test team decided to stop by cycle 2.

Test A Based on the estimation of color differences at the control points, the printer characteristic curves were updated for the HP printer. The curves after cycles 1 and 2 are shown in Figure 5.4.

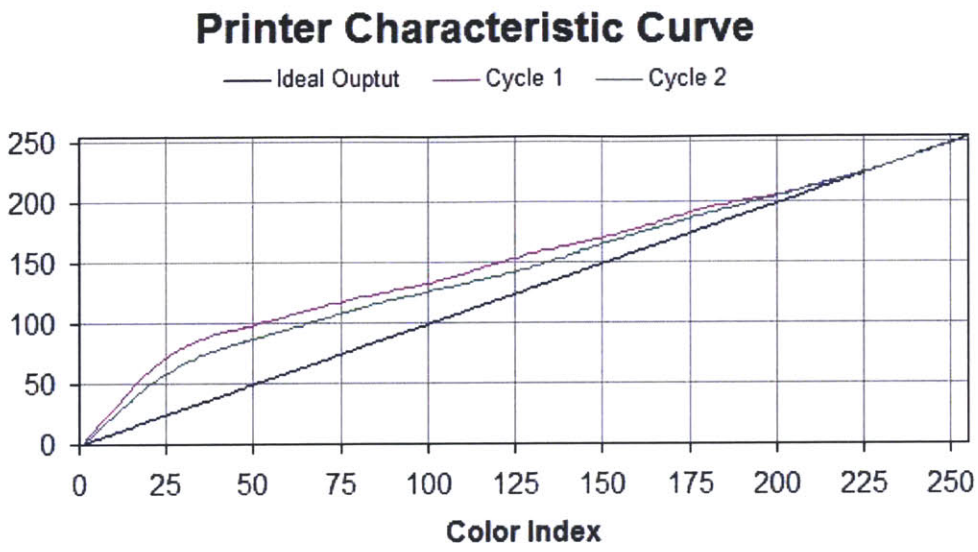


Figure 5.4
The printer characteristic curves with CtC control (R channel) (In an ideal color printing system, the output values strictly equal to their corresponding indexes, as shown in the black line; the pink and green curves, which are approximate to the reality, are updated with the comparison result.)

Figure 5.5 shows the average errors at different color density levels in the validation set of the final printout. As important scales of the color reproduction accuracy, the errors in CMY Black were also measured and shown in Figure 5.6. It is noticeable that the midtone errors on both figures are significantly reduced. However, the error reductions at other regions are less effective. This can be explained by that human eyes are not sensitive to color differences at highlight and shadow regions. Thus, a better result might be achieved if the printer characteristic curves are updated objectively rather than subjectively. For this purpose, a scanner would need to be introduced with a color space transformation function.

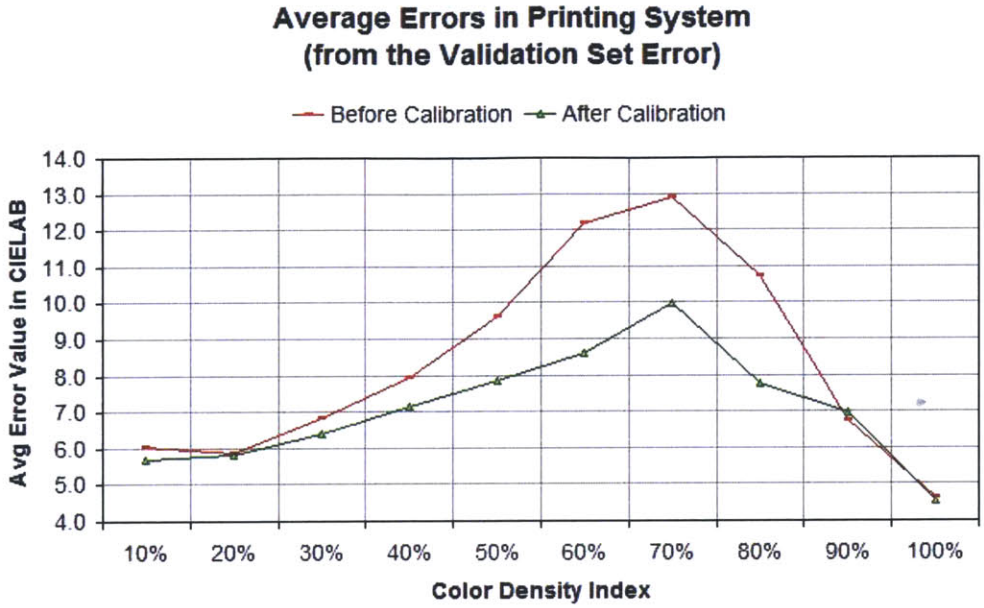


Figure 5.5
The average color errors in the printing system (data from the validation set, unit: delta E value calculated with CIELAB 2000 formula)

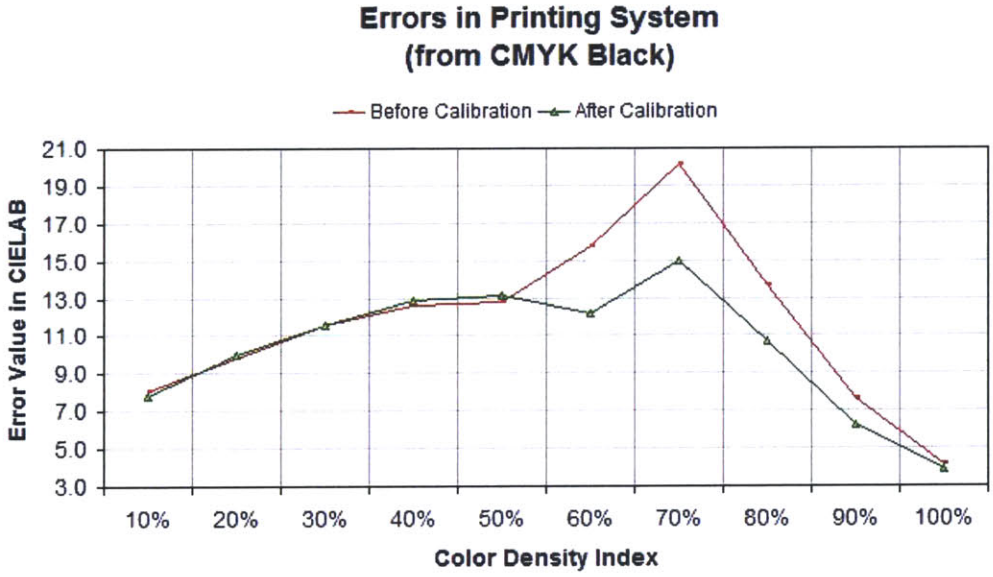


Figure 5.6
The average errors of CMYK Black in the system (data from the validation set, unit: delta E calculated by CIELAB 2000 formula)

The analysis of the picture quality was carried out with the printout shown in Figure 5.7. There are several problems in the original printout. For example, the faces in the top left picture and the dress of the Caucasian lady appear too reddish; and it is not easy to find the rumples on the curtain since the color is too dark; in addition, the fruits in the bottom left picture look too vivid. These reproduction errors imply that the red at midtone and shadow regions are too much. It turns out the magenta component there is too high. To correct the colors, the green curve at highlight and midtone needs to be modified. The pictures also confirm the conclusion drawn from a comparison of RGB color bars.

The test went through two cycles. The printout from cycle one (not shown in this thesis) had a whitish tint because the pictures were over-compensated. The second cycle gave the desired result: the skin complexion is close to reality, and the fruits look a lot more realistic; the whitish tint is significantly reduced. Qualitatively, this printout gives the best result, and proves that the workflow and color compensation mechanism are effective.

It should be noted once again that it is rather difficult to fully appreciate the print out quality through this image on screen. As discussed in section 5.2, the image was actually compensated for a particular printer rather than for the screen. The calibration process thus works as a "*What-You-See-Is-NOT-What-You-Get*" system. The electronic copies of the rendered image will not exhibit the same qualities seen when they were printed. Instead they will exhibit the effect from printing on the *MIT acid-free* paper.

Figure 5.7
The color image before and after the calibration (The pictures might look slightly different from the description in the thesis. This is due to the fact that the printing conditions and the substrate used for this thesis could be different from those under which the tests were carried out.)

Test B The test was also performed on a Canon LASERSHOT LBP5200 printer with different paper. The compensation controller significantly improves the color accuracy at midtone and shadow regions (as shown in Figure 5.8). This calibration of the printer improves image reproduction result. Indeed this qualitative analysis remains subjective though the team consulted three color experts - Ryan Wong, Xian Du and Dr Allan Zhang to verify the improvements (the image printout is not shown here).

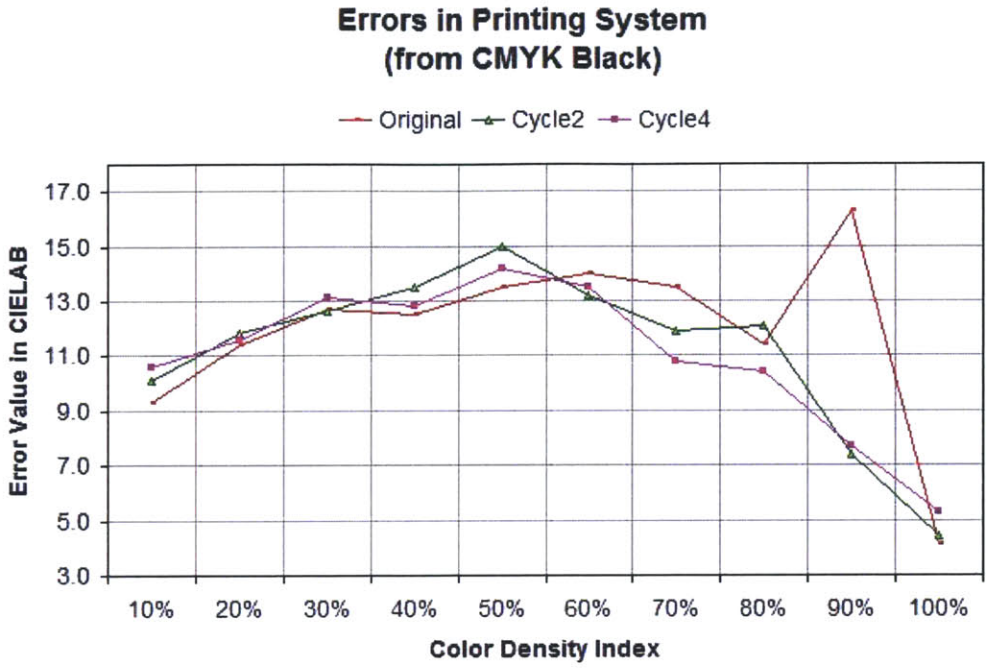


Figure 5.8
The average errors of CMY Black using different printing conditions and substrate (As the calibration process was carried on, Cycle 4 showed better error reduction effect than Cycle 2.)

5.4. Summary

In this chapter, the study designed a few test scenarios to verify the effect of the printer calibration system. The results from these tests show that a significant error reduction was realized by incorporating the interpolation method into Windows printing architecture. However, the error correction at highlight region is not so great because of the incapacity of human eyes in detecting color differences in this region. The result can be improved if a quantitative instrument (like a scanner) is introduced to detect the color differences at the control points.

CHAPTER 6
CONCLUSION AND FUTURE WORK

This study proposed and tested a fast correction method with cycle-to-cycle control in printer calibration process. Two major problems, the creation of look-up table and image compensation in Windows system, were discussed in chapters 3 and 4 respectively. Chapter 3 evaluated several types of interpolation methods, including polynomial interpolation and neural network approaches. The most successful one is the quadratic spline interpolation method, which accurately captures printer characteristic curves via limited number of error samples collected at the control points. Chapter 4 focused on how to integrate the color compensation mechanism into Windows printing architecture. Three major components, UI plug-in, Rendering plug-in, and Color-Palette API, are involved into the implementation process. A significant reduction in error was realized by incorporating the interpolation method into color management program.

Based on the test results and analysis in chapter 5, the project can be judged as a success. A working prototype is also ready for the demonstration to potential customers. However, a few issues still need to be addressed in the future development.

Firstly, the control module needs further development of automatic color correction. Currently the process requires the user to detect the color differences between the standard chart and the printed output, and modify printer characteristic curves manually. The subjective judgment from human eyes might lead to the under-compensation and over-compensation. The further development would focus on how to read the scanned data, and adjust printer characteristic curves automatically.

Secondly, further research should be carried on to improve the numerical method in characteristic curve construction. In order to get smoother approximations, the study will consider some advanced interpolation methods, such as *B-spline approach* and *Variation*

Diminishing Spline approximation. An alternative way is to adopt the neural network, but increase its convergence speed with fast learning functions.

Thirdly, the current solution needs further development to save the output in various image formats. In order to provide flexible choices on output quality and size, the software package should involve additional image-processing modules for TIFF, MPEG, JPG image formats. The investigation on their specifications and implementations then becomes a necessary step.

BIBLIOGRAPHIC REFERENCES

[1] Larry Dignan, *HP leads in soft printer market*, http://news.com.com/HP+leads+in+soft+printer+market/2100-1001_3-975514.html, November 2002

[2] *HP Resets Color Office Printing Market with Price and Breakthroughs*, http://www.creativepro.com/story/news/17745.html?origin=story, September 2002

[3] *Canon Annual Report 2002*, http://www.canon.com/ir/annual/index.html, 2002

[4] *HP Annual Report 2002*, http://www.hp.com/hpinfo/investor/financials/annual, 2002

[5] Bruce Fraser, *The Bigger Players Bring Quick Prints within Reach*, MacWorld, September 2002

[6] Jim Rich, *Calibration choices in Real World Workflow*, http://www.digitaloutput.net/content/ContentCT.asp?P=213, March 2002

[7] Heidelberger Druckmaschinen AG, *Test Charts and Control Elements User's Guide*, August 2002, 26-29

[8] Mark Shaw, Gaurav Sharma, Raja Bala, Edul N. Dalal, *Color Printer Characterization Adjustment for Different Substrates*, Color Research and Application, November 2002

[9] Michael J. Vrhel, H. Joel Trussell, *Color Device Calibration: A Mathematical Formulation*, IEEE Transactions on Image Processing, Vol. 8, No. 12, December 1999

[10] I. J. Schoenberg, *On variation diminishing approximation methods*, in On Numerical Approximation (R. E. Langer, ed), U. Wis. Press (Madison), 1959, 249–274

[11] G. Muhlbach, *The general Neville-Aitken algorithm and some applications*, Numer. Math. 31 (1978), 97-110

[12] Microsoft Corporation, *MSDN library*, 2006

[13] Microsoft Corporation, *Windows Server 2003 SP1 DDK*, 2006

[14] Curtis N. Vanderpuije, *Innovative Color Management Methods for RGB Printing*, Master thesis, Mechanical Engineering Department, MIT, September 2006

[15] Bruce Fraser, Chris Murphy, Fred Bunting, *Real World Color Management*, Peachpit Press, 2004, 19-21

[16] Xian Du, *Issues on the optimization of color print process control*, Ph.D. thesis, Mechanical Engineering Department, National University of Singapore, July 2006

APPENDIX A

A MATLAB PROGRAM FOR INTERPOLATION

A.1 LAGRANGE INTERPOLATION

```
% ##########################
% # Lagrange Interpolation #
% ##########################

function y = lagrange(x0,y0,x)

n = length(x0);
m = length(x);

for i=1:m
    z=x(i);
    s=0.0;

    for k=1:n
        p=1.0;

        for j=1:n
            if j~=k
                p=p*(z-x0(j))/(x0(k)-x0(j));
            end
        end

        s=p*y0(k)+s;
    end

    y(i)=s;
end
```

A.2 COMPARISON OF INTERPOLATION METHODS

```
% ########################################
% # A Comparison of Interpolation Methods #
% ########################################

% Control Point Data
% x0  color index of the control point
% y0  color output error
x0  = [0 3 5 7 9 11 12 13 14 15];
y0  = [0 1.2 1.7 2.0 2.1 2.0 1.8 1.2 1.0 1.6];

% Feedin
x   = 0:0.1:15;

% Calculate Output via Interpolation Methods
```

```
y1  = lagrange(x0,y0,x);                    % Lagrange
y2  = interp1(x0,y0,x);                     % Piecewise linear
y3  = interp1(x0,y0,x,'spline');            % Cubic spline
pp1 = csape(x0,y0);                         % Default
y4  = ppval(pp1,x);
pp2 = csape(x0,y0,'second');              % Match end second derivatives
y5  = ppval(pp2,x);

[x',y1',y2',y3',y4',y5']

% Result Plots
subplot(2,2,1)
plot(x0,y0,'+',x,y1)
title('Lagrange')

subplot(2,2,2)
plot(x0,y0,'+',x,y2)
title('Piecewise Linear')

subplot(2,2,3)
plot(x0,y0,'+',x,y3)
title('Cubic Spline')

subplot(2,2,4)
plot(x0,y0,'+',x,y5)
title('2nd-dev Spline')
```

A.3 NEURAL NETWORK WITH ORIGINAL TRAINING SET

```
% ##############################
% # BPNN Neural Network        #
% # (trained by original data) #
% ##############################

% Control Points Data
x0  = [0 3 5 7 9 11 12 13 14 15];
y0  = [0 1.2 1.7 2.0 2.1 2.0 1.8 1.2 1.0 1.6];

% Feedin
x   = 0:0.1:15;
% m = length(x);

m=1;

% Network initialization and training
net = newff(minmax(x0), [10, 1], {'tansig', 'purelin'}, 'trainlm');
net.trainParam.show = 5;
net.trainParam.epochs = 500;
net.trainParam.goal = 1e-6;
[net, tr] = train(net, x0, y0);

% Write the output array
for i=0:0.1:15
    y(m)=sim(net,i);
```

```
    m=m+1;
end

figure

% Plot
plot(x0,y0,'+',x,y)
title('BPNN')
```

A.4 NEURAL NETWORK WITH NEW TRAINING SET

```
% ##############################
% # BPNN Neural Network        #
% # (trained by lager sample) #
% ##############################

% New Training Set
% p   color index
% t   (output error)/15
p = [0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105
110 115 120 125 130 135 140 145 150 155 160 165 170 175 180 185 190 195
200 205 210 215 220 225 230 235 240 245 250 255];

t = [0.000 0.267 0.400 0.000 -0.067 -0.800 -0.467 0.200 -0.067 -0.400 -
0.467 -0.333 -0.333 0.000 0.267 0.000 -0.133 -0.067 0.267 0.333 0.000
0.267 0.133 -0.333 -0.133 0.333 0.933 0.400 0.667 0.067 0.133 -0.067 -
0.533 0.133 0.000 -0.200 -0.733 -0.333 0.267 0.467 0.800 0.400 -0.200
0.067 0.333 -0.267 -0.067 -0.467 -0.800 -0.733 -0.067 0.000];

% Construct a Multi-layer BPNN
net = newff(minmax(p), [50, 5, 1], {'tansig', 'tansig', 'purelin'},
'trainlm');

% BPNN Initialization
net.trainParam.show = 5;
net.trainParam.epochs = 5000;
net.trainParam.goal = 1e-2;

% BPNN Training
[net, tr] = train(net, p, t);

% NNet after the Training
a = sim(net, p);
```

APPENDIX B THE NEW TEST CHART