8-1-2010

Print engine color management using customer image content

Michael William Elliot

Follow this and additional works at: <http://scholarworks.rit.edu/theses>

Recommended Citation

Elliot, Michael William, "Print engine color management using customer image content" (2010). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by the Thesis/Dissertation Collections at RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.

Print Engine Color Management Using Customer Image Content

by

Michael William Elliot

A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Master of Science
in Computer Engineering

Supervised by

Associate Professor Dr. Juan Carlos Cockburn
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
August 2010

Approved by:

Dr. Juan Carlos Cockburn, Associate Professor
Thesis Advisor, Department of Computer Engineering

David C. Craig, Color Controls Manager
Committee Member, Xerox Corporation

Dr. Andres Kwasinski, Assistant Professor
Committee Member, Department of Computer Engineering

Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title:

Print Engine Color Management Using Customer Image Content

I, Michael William Elliot, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

Michael William Elliot

Date

© Copyright 2010 by Michael William Elliot
All Rights Reserved

Dedication

To my wife - 'To whom I owe the leaping delight...'
To my parents - Your wisdom, knowledge, and love have shown me the
way.

Acknowledgments

I would like to thank Professor Cockburn for his attention to detail, constructive criticism, and eternal patience, which enabled me to complete this thesis. I am also forever in debt to David Craig, who's practical knowledge and approach made this a happy experience. I would also like to thank Professor Kwasinski for participating in this journey. Lastly, I offer my regards to all of those who supported me in any respect during the completion of the project.

Abstract

Print Engine Color Management Using Customer Image Content

Michael William Elliot

Supervising Professor: Dr. Juan Carlos Cockburn

The production of quality color prints requires that color accuracy and reproducibility be maintained to within very tight tolerances. Variations in the printing process commonly produce color shifts that result in poor color reproduction. The primary function of a color management system is maintaining color quality and consistency. Currently these systems are factory tuned by printing a large set of test color patches, measuring them, and making necessary adjustments. This time-consuming procedure should be repeated as needed once the printer leaves the factory. In this work, a color management system that compensates for print color shifts in real-time using feedback from an in-line full-width sensor is proposed. Instead of printing test patches, this novel attempt at color management utilizes the output pixels already rendered in production pages, for a continuous printer characterization. The printed pages are scanned in-line and the results are utilized to update the process by which colorimetric image content is translated into engine specific color separations (e.g. CIELAB->CMYK). The proposed system provides a means to perform automatic printer characterization, by simply printing a set of images that cover the gamut of the printer. Moreover, all of the color conversion features currently utilized in production systems (such as Gray Component Replacement, Gamut Mapping, and Color Smoothing) can be achieved with the proposed system.

Contents

Dedication	iv
Acknowledgments	v
Abstract	vi
1 Introduction	1
2 Supporting Work	4
2.1 State of the Art Digital Printing	4
2.1.1 The Printing Process	4
2.1.2 Digital Printer Work-flow	7
2.2 Imaging and Sensing	9
2.2.1 Digitization of Images	9
2.2.1.1 Image Sampling	9
2.2.1.2 Image Quantization	11
2.2.1.3 Color Image Digitization	11
2.3 Color Management	12
2.3.1 Color Spaces	12
2.3.1.1 CIEXYZ and CIELAB Color Spaces	14
2.3.1.2 Color Difference in CIELAB	16
2.3.2 Color Measurement	17
2.3.3 ICC Profile Standard	18
2.3.4 Printer Characterization / Output Profile Generation .	19

2.3.4.1	Recharacterization Approaches	21
3	The Proposed System: Continuous Recharacterization . . .	23
3.1	High Speed Rendering	24
3.1.1	Tri-linear Interpolation	25
3.1.2	Tetrahedral Interpolation	25
3.2	Performance Evaluation of a 3D-LUT	26
3.2.1	Performance Metrics	26
3.2.2	The Importance of a General Color Management Work-flow	28
3.2.3	Printer Model Generation	31
3.2.3.1	Static Printer Model	31
3.2.3.2	Dynamic Printer Model	32
3.3	3D-LUT Generation	35
3.3.1	Gray Component Replacement and Under Color Re- moval (GCR/UCR)	36
3.3.1.1	K-Restricted GCR Strategies	37
3.3.1.2	The K Suppression Strategy	37
3.3.1.3	GCR Implementation	38
3.3.2	Source Profile and 3D-LUT Structure Generation . .	40
3.3.3	Gamut Mapping	44
3.3.3.1	Gamut Boundary Descriptors	44
3.3.3.2	Segment Maxima Method	46
3.3.3.3	Gamut Boundary Implementation	50
3.3.3.4	Gamut Mapping Strategies	51
3.3.3.5	Implemented Gamut Mapping Strategies .	57
3.3.3.6	Assessment of the Gamut Boundary De- scriptor (GBD)	63
3.3.4	Characterization Techniques	63

3.3.4.1	Mathematical Background: Implicit Function Theorem	64
3.3.4.2	Moving Matrix Interpolation (MMI) . . .	64
3.3.4.3	Iteratively Clustered Interpolation (ICI) . .	66
3.3.4.4	Adaptive Local Linear Regression (ALLR)	67
3.3.4.5	Continuous Recharacterization (CR) Technique	68
3.4	Customer Image Feedback with an Inline Scanner	69
3.4.1	Selecting Customer Image Areas for Use in Feedback	71
3.4.1.1	Image Segmentation using Color Similarity	71
3.4.1.2	Assigning a Color Number	72
3.4.2	Customer Image Capture	76
3.4.2.1	Timing the Scanner	76
3.4.2.2	Scanner Profile / Colorimetric Captured Image	78
3.4.3	Updating the Color Library and the 3D-LUT	79
3.4.3.1	Updating the Color Library	80
3.4.3.2	Updating the 3D-LUT	84
4	Experimental Results	85
4.1	Characterization Performance	85
4.2	Characterization Routines and Drifting Printers	91
4.3	Recharacterization Performance	92
4.3.1	Full Printer Gamut Coverage Performance Analysis .	93
4.3.2	Partial Printer Gamut Coverage Performance Analysis	108
4.3.3	Characterization for Free	111
4.3.4	Neutral Drift Experiment	112
5	Conclusions and Future Work	114

Bibliography	116
---------------------	------------

List of Tables

3.1	Computational Complexity of Tri-linear and Tetrahedral Interpolation	25
3.2	Gamut Corner Points in CMYK Printer	29
3.3	Performance Comparison of Clustered Regression and Global Regression Printer Models	34
3.4	Color Library Entries	80
3.5	New Color Data Example	81
3.6	Color Library Entries After Update	82
4.1	Characterization Technique Performance Summary	86
4.2	Continuous Recharacterization Characterization Performance	91
4.3	ICI Performance with Printer Drift (FOGRA39, FOGRA43)	92
4.4	CR Performance with Printer Drift (FOGRA39, FOGRA43)	101
4.5	DeltaE Image Decoding Values	101
4.6	Lena Image Feedback Results Using CR Algorithm with Printer Drift	111
4.7	Performance of Printing Full Gamut Patches Versus Formal Characterization	112

List of Figures

2.1	Xerographic Process	5
2.2	Print Station Work-flow	8
2.3	Two and Four Degree Observer Angles	15
2.4	ICC Profile Work-flow	18
2.5	Rendering Work-flow of a CMYK Printer	21
3.1	Proposed System Work-flow	24
3.2	High Speed Rendering Work-flow	25
3.3	Color Corners	29
3.4	Simulated Printer Characterization Work-flow	30
3.5	Simulated Printer Characterization Performance Analysis Work-flow	31
3.6	Performance Graphs for Clustered Regression Printer Model (Fogra41L)	33
3.7	Performance Graph for Global Regression Printer Model (Fogra41L)	33
3.8	3D-LUT Generation Work-flow	36
3.9	Neutral Response of K with $\alpha = 0.01$, $\beta = 1.5$, and $\eta = 0.5$ for Various Values of L_0	38
3.10	Chromatic Response of K with $\alpha = 0.01$, $\beta = 1.5$, $\eta = 0.5$, and $L_0 = 102$ for Various Values of L	39
3.11	Maximum-K GCR Dark Blue and Dark Black Corner Plots .	41
3.12	Minimum-K GCR Dark Blue and Dark Black Corner Plots .	42
3.13	K-Constrained GCR Dark Blue and Dark Black Corner Plots	43

3.14	Spherical Coordinate System for CIELAB	48
3.15	Printer Gamut vs Aqua Image Gamut	52
3.16	Printer Gamut vs Ski Image Gamut	53
3.17	Closest Point Gamut Mapping Algorithm	55
3.18	Straight Clipping Gamut Mapping Algorithm	55
3.19	Centroid Clipping Gamut Mapping Algorithm	56
3.20	Gamut of Aqua (a) and Ski (b) Images after Centroid Gamut Mapping	58
3.21	Aqua (a) and Ski (b) Images with Closest Point GMA	60
3.22	Aqua (a) and Ski (b) Images with Centroid Clipping GMA .	61
3.23	Aqua (a) and Ski (b) Image with Straight Clipping GMA . .	62
3.24	ICI Augmented Printer Concept	66
3.25	Local Model Description	70
3.26	Proposed System Work-flow	71
3.27	Segmentation and Capture	73
3.28	Tuning of <i>colorSimilarity</i>	75
3.29	Positioning and Timing Scanner in Printer	77
3.30	MaxDeltaE/CMYK Tradeoff	83
4.1	Continue Recharacterization Characterization Performance (FOGRA39)	87
4.2	Adaptive Local Linear Regression Characterization Performance (FOGRA39)	88
4.3	Moving Matrix Interpolation Characterization Performance (FOGRA39)	89
4.4	Iteratively Clustered Interpolation Characterization Performance (FOGRA39)	90
4.5	Neutral Drift Using ICI Algorithm	93
4.6	Art Image Results Using ICI Algorithm with Printer Drift . .	94

4.7	Lena Image Results Using ICI Algorithm with Printer Drift .	95
4.8	Spring Image Results Using ICI Algorithm with Printer Drift	96
4.9	Art Image and Gamut	97
4.10	Lena Image and Gamut	98
4.11	Spring Image and Gamut	99
4.12	Art Image Results Using CR Algorithm with Printer Drift . .	102
4.13	Lena Image Results Using CR Algorithm with Printer Drift .	103
4.14	Art Image Results Using CR Algorithm with Printer Drift . .	104
4.15	CR Algorithm Dark Red Corner Plot with Printer Drift . . .	105
4.16	Printer Gamut with Printer Drift	106
4.17	Art Output Image Gamut with Printer Drift	107
4.18	Lena Image <i>DeltaE</i> Feedback Results Using CR Algorithm with Printer Drift	109
4.19	Lena Image <i>Printed Image</i> Results Using CR Algorithm with Printer Drift	110
4.20	Neutral Drift Using CR Algorithm	113

Chapter 1

Introduction

The printing industry remains a multi-billion dollar industry and has recently seen a large growth in the demand for color images. Within the color print industry there are a large range of color accuracy requirements, the most rigid and demanding of those being the requirements necessary for the graphics arts business. Here, color accuracy and reproducibility must be maintained to within very tight restrictions. Numerical accuracy of color transforms, process noises, and high-speed rendering requirements make this activity particularly difficult. As a result, it is common for the quality of a device characterization (the mapping from an absolute color space to a device specific color space) to degrade over time, resulting in a time-consuming calibration or characterization procedure. Instead, it is desirable for color accuracy to be maintained while printing by means of a feedback mechanism. However, the printing of patches on customer images is too intrusive. Thus, during production controls are typically limited to sensors on the printing apparatus, such as on a photoreceptor in the electrophotographic process. This level of feedback is not an accurate predictor for color shifts, particularly for colors that require overlays of multiple device colorants. Thus, while it is desirable to capture and adjust based on image content on the printed page, no such system has been introduced to the industry due to customer requirements.

State of the art printer *characterization* or *system identification* utilized in *color management systems* is performed through a static snap-shot of a print engine's sensitivities (color mapping), by measuring a pre-determined test pattern with a spectrophotometer. By measuring a set of known, previously

determined image contents, the system is able to generate the inverse printer mapping and create a model, typically via look-up-table or *LUT*, that represents the mapping from CIE color spaces to device specific color spaces. Process Controls and linearization techniques within the print engine are then responsible for maintaining print to print color consistency. This empirical characterization approach outperforms any closed form model of the printer, due to the dynamics and non-linearity of most printing processes. However, despite attempts to recalibrate certain regions in the printing process, the quality of the characterization can degrade as the printer drifts. This can occur due to a variety of process noises and engine conditions that cannot be adjusted for by process controls. Likewise, it is not capable of adapting to color gamut regions not measured by the characterization process. Thus, color performance degrades with printer drifts until characterization is performed again. Not only does the characterization process require customer consumables and machine down-time, it is also subject to arbitrary user intervention, since the machine is not directly capable of sensing or detecting dissatisfying color reproduction.

The central goal of this thesis is to design and assess the feasibility of using customer image content to ensure color stability, in a novel color management scheme. In the proposed system, once the images are printed, they will be scanned by an in-line scanner, and the results will then be utilized to adjust the manner in which color recipes are generated. The goal is not to eliminate the characterization process, which still provides the foundation of the color management system, but instead to augment the current procedure during print by continuously updating the printer inverse map. Color conversion features such as Gray Component Replacement, Gamut Mapping, and Color Smoothing are also explored in detail, since these topics are also subject to the dynamics of during print adjustment and optimization.

In order to evaluate this concept's performance, we utilized an empirical printer model and introduced printer drift. With this in hand, a suitable recharacterization technique was designed, implemented and run against the simulated printer. The system's performance was then compared to other state of the art methods in both real-time usage and also in a variety of color performance measurements, including round-trip numerical accuracy,

gamut utilization, and color smoothness. From our simulations it was very clear that engine color shifts were not handled by existing art. Thus we concluded that our strategy is not only capable of real-time adjustments to accommodate for printer color drifts, but it is also similar in computational complexity and color performance.

The remainder of the thesis will have the following format. We will first visit some background material required to understand the proposed system in Chapter 2. This will include details on Digital Imaging and Color Management. We will then decompose the proposed system into three distinct problems, motivate these problems, and outline the implemented work. The outline will include any performance evaluation used to aid in the selection of a particular strategy in the final system. Chapter 4 will cover detailed experiments run to evaluate the performance of the proposed system. We will close with some final thoughts and avenues to pursue in future work.

Chapter 2

Supporting Work

In order to understand the complexities of the proposed system, one must first have a solid grasp of digital imaging and color management. This knowledge is essential to comprehend how the proposed system extends the state of the art.

2.1 State of the Art Digital Printing

2.1.1 The Printing Process

Image reproduction technologies include, amongst others, offset (Lithography) printers and digital printers. Digital printers utilize a variety of imaging processes including solid ink, ink jet and xerographic or electrophotographic. While the reproduction quality and consistency of offset printers continues to outperform digital printers, digital printers offer additional benefits including low-cost short run jobs and print-on-demand abilities. Furthermore, digital printers provide pixel level variation on page boundaries, which is currently not possible in the offset printing environment. As a result of this flexibility, opportunities for advanced control methods are enabled [1].

The general xerographic process includes six stages, as seen in Figure 2.1. First, a static charge is placed on a circular photo-conductive belt or drum. The *photoreceptor* is subsequently exposed using a light or laser. The latent image is then developed, by transferring toner from a donor or magnetic brush to the photoreceptor. In tandem engine environments, the

Figure 2.1: Xerographic Process

developed image is transferred to an intermediate belt before the image is transferred to paper and then fused. Before returning to the charging station, the photoreceptor (and possibly intermediate belt) is cleaned with a brush [1, 2].

The xerographic process developed by Chester Carlson was introduced in the first commercial printer in 1960 by Xerox. While this device followed the same basic xerographic process outlined above, it had no closed loop controls. As a result, any changes to control parameters had to be manually updated to obtain reasonable reproduction. In the 1970s Xerox introduced the use of an Automatic Density Control Sensor (ADC), which was able to sense the *developed mass* that was used as a surrogate to measure the development rate of the xerographic engine. This controls architecture was utilized in the Xerox 4000, 5600, and 9200 duplicator families. In the 1980s, in order to continue to improve the quality of imaging and allow for less stable, but less expensive drum/charging technologies, Xerox introduced an electrostatic-voltmeter (ESV) into the 1075, 1090, and 4050 products. In this configuration, two patches were utilized to represent the customer's image content. By reading a low density patch with the ADC, the

developability could be controlled, while the ESV could provide feedback for the charging process. Both systems were designed as single input single output (SISO) controllers. While the changing of electrostatic parameters does change the development process, this was managed by performing the charge control loop at a much higher frequency than the developability control loop [1, 3].

Color xerography introduces additional complexities, especially when trying to compete with offset printing technologies and reproduction accuracy. Particularly difficult is keeping each station's control stable, in the presence of customer job noises (*area coverage*), shifts in the environment and device aging effects. In addition, controlling image to image (separation) timing accurately is also required. Under these standards, grays and highlights are the most difficult to keep consistent due to their sensitivity to small changes in developability [2, 4]. The two most common technologies for xerographic color systems include different trade-offs in simplicity. The image-on-image process, as seen in the iGen3 and iGen4 products, exhibits interactions between separations, since in that technology charging and development occurs onto the same photoreceptor belt. Whereas the tandem or intermediate belt technology, as seen in the DC8000 product, introduces additional noise in separation timing due to the additional transfers from drum to intermediate belt [1, 3, 2].

In digital printing, the output is generated by combining a set of Cyan, Magenta, and Yellow (CMY) ink. In order to use less ink an additional dark or black color, known as *Key*, is added to the set of colorants, which as a group are abbreviated CMYK. A black generated with only CMY is sometimes called a composite black, where a black generated with CMYK is sometimes called a rich black. In order to realize more than six colors, a process known as halftoning is utilized to change the intensity of a particular color separation. In particular, a twenty percent halftone of cyan is simply a lighter version of a solid cyan patch, since the halftoning process adds or removes dots in order to achieve the desired intensity [1].

Modern color controls in the iGen3 and iGen4 products range from digital image processing to digital MIMO systems. Within engine process controls are responsible for the control and maintenance of the xerographic

process, in particular the stability and the page-to-page accuracy of each reproduction. Controls at this level include electrostatic, developability or toner concentration (TC), and TRC controls. These are done with a hierarchy of controls, starting with simpler local controls with additional higher level controllers performing higher level adjustments. Electrostatic controls can compensate for shifts in development and charging by changing set points in charging, exposure, development, and fusing. Toner concentration controls actuate the toner to carrier relationship in the developer housing system. Indirectly changing the toner concentration changes the effective charge (tribo) of the toner particles, changing the manner in which toner is developed. Tighter TC and tribo controls could greatly improve the stability of a separation's reproduction. However, due to the large noises introduced by the job stream (i.e. printing) and the lack of a good, reasonably priced tribo sensor, this is not a reality. Finally, in order to improve the stability of image generation, TRC adjustments can actuate a variety of components in the system. These include TC and electrostatic targets, as well as digital actuation through the use of 1D engine LUTs [1, 2, 3].

2.1.2 Digital Printer Work-flow

The general work flow of a printer can be seen in Figure 2.2 and is as follows: A user, in the pre-press phase of development, generates all content in and through the use of desktop publishing software. This generates device independent content either in post-script (PS, PDF) or page description language (PDL). This step may include a soft-proofing process, which takes into account a model of the printer's imaging process, and can be seen on a calibrated monitor. The user's document is then submitted to the first line of processing, referred to as the Digital Front End (DFE). Here various image processing steps take place including image resolution enhancement, noise removal, trapping, anti-aliasing, and segmentation. More importantly, however, are the color management features performed after these steps. In the color management phase, a document described in the RGB color space is translated into the CIELAB color space and then to device specific CMYK contone values. The DFE then submits the job to the print station,

Figure 2.2: Print Station Work-flow

for production[1].

Despite attempts to control the process, machine to machine differences in sensing and actuation can still cause digital printer color shifts. The reproduction of neutral, memory colors, such as skin is the most difficult task. Here small decreases in magenta can alter the hue towards green, where small increases of magenta can cause the reproduction to be clearly magenta. As a result, system performance is measured against a standard. This requires the use of a calibrated colorimetric device (e.g. calibrated scanner or spectrophotometer). Readings from such devices are the cornerstone on which modern color management is built and will be discussed further in Section 2.3[4, 1, 5].

2.2 Imaging and Sensing

Image content can be sampled from nearly every type of radiation. Our focus is on the sampling, digitization and reproduction of visible light. All image capture starts with a radiating element such as the sun. The radiation hits scene objects and is reflected. The resulting radiation may hit sensors which can produce electrical signals representing the effective scene. The human eye acts as one such sensor in the human visual system. Similarly, digital cameras and scanners act as sensors for computational systems. Here, instead of communicating with the human brain to create scene experience, the electrical signals are then digitized and processed to extract information about the scene or image by computers. Digital image processing offers a large span of exciting and new possibilities for improving image reproduction [6, 1].

2.2.1 Digitization of Images

By nature signals and images available in the environment are naturally analog. In particular they are multi-dimensional continuous time signals. The human visual system processes these signals as such, but in order for computers to utilize and interpret them, a process of digitization is required. This results in multi-dimensional image content that consists of a discrete set of possibilities. The Digitization process requires two distinct steps: sampling and quantization [6].

2.2.1.1 Image Sampling

In order to process an image digitally, the image must first be sampled or digitized. This involves looking at the tonal levels in a 2D grid and mapping those into a set of bits to represent each area. The ideal sampler captures information each $\nabla x, \nabla y$ step or period. As a result the ideal sampler can be represented as follows [1].

$$s(x, y) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} \delta(x - n\nabla x, y - \nabla y) \quad (2.1)$$

The sampled image, $f * (x, y)$ is therefore:

$$f * (x, y) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} f_a(n\nabla x, m\nabla y) \delta(x - n\nabla x, y - \nabla y) \quad (2.2)$$

where $f_{sx} = \frac{1}{\nabla x}$ and $f_{sy} = \frac{1}{\nabla y}$ are the sampling frequencies in the x and y directions.

The spectrum of the sampled image is obtained by taking the Fourier Transform of $f * (x, y)$ [1].

$$F * (\omega_x, \omega_y) = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} f(n, m) e^{-j(\omega_x n \nabla x + \omega_y m \nabla y)} \quad (2.3)$$

$$F * (\omega_x, \omega_y) = f_{sx} f_{sy} \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} F(\omega_x + n2\pi f_{sx}, \omega_y + m2\pi f_{sy}) \quad (2.4)$$

Since the Fourier Transform of the sampled signal describes a periodic extension of the Fourier Transform of the analog signal, the original signal can be reconstructed by low-pass filtering the sampled signal. This is true assuming that the sampling frequencies satisfy the Nyquist Frequency (i.e. $\nabla x \leq \frac{1}{2f_{x0}}$, $\nabla y \leq \frac{1}{2f_{y0}}$, where f_{x0} and f_{y0} are the fundamental frequencies of the sampled signal). Simply put, if a signal in an image is to be captured, we must sample at least two times that rate, lest aliasing will occur, causing high-frequencies to appear as low frequencies. Avoiding this issue is as simple as low-pass filtering the input signal *prior* to sampling to ensure that the image $f(x, y)$ is band limited to half the sampling frequency. Such a filter is commonly called an anti-aliasing filter. This is important because once aliasing occurs, it is impossible to recover the original image.

2.2.1.2 Image Quantization

A very typical approach for quantizing an analog signal is to utilize Uniform Quantization. A Uniform Quantization defines a set of *decision levels* and *reconstruction levels* uniformly over a line. Each decision level is separated by a step size which can be determined by solving for Δ , where $Bits$ refers to the number of bits used to represent the reconstruction levels.

$$\Delta = \frac{x_{max} - x_{min}}{2^{Bits}} \quad (2.5)$$

For uniform distributions, such a system is optimized for mean square error, where the signal to quantization noise ratio is lowered by 6 dB per bit that is added to the system [1]. However, for systems such as imaging systems, the underlying analog signal's change in intensity does not directly map to shifts in the human visual system response. It turns out that the human visual system is sensitive to percent shifts in intensity. The human visual system can sense small increases in the number of photons at low intensities, but large increases in the number of photons at high intensities may not be perceivable. As a result, in order to prevent defects caused by a strictly linear mapping, as in Uniform Quantization, imaging systems typically sample at larger than 8 Bit levels, typically 10 to 16 Bits. The approximately logarithmic system that maps photons to intensity as perceived by the human eye can then be approximated into 8 bit values [6, 1]. These perceptual quantizations are known as non-uniform Scalar Quantization.

Vector Quantization generalizes regions into decision groups. This technique has less error with a similar number of bits. However, the process for computing the optimal quantizer, from a means square error perspective, is more complicated for vector quantization. The Lloyd algorithm, or k-means algorithm is often used to design the optimum decision regions and reconstruction vectors for Non-Uniform Scalar Quantization [6, 1].

2.2.1.3 Color Image Digitization

The human eye senses colors as a near-linear combination of long, medium, and short wavelengths. These correspond roughly with the three primary

colors that are utilized in standard cameras and scanners: Red, Green, Blue (RGB). Thus, the above process of sampling and quantization is performed on each of the three planes independently. Three separate planes are generated for representing digitized color image content. The combination of these values in an output device can ideally reproduce the original color digitized in the previously mentioned process. However, it must be noted that the processes of image sampling and quantization are device and sensor specific or are *relative*. Therefore, relating the results of quantization from separate relative sensor and digitization systems is not possible directly [6, 1].

2.3 Color Management

Color management is the process of converting device specific input values (such as from a scanner or digital camera) into device specific output values for imaging on an output device (such as a monitor or printer). Primarily this transformation is interested in ensuring that the colors reproduced on the output device accurately reflect the input document and produce a pleasing color output. The International Color Consortium defines a standard paradigm for managing color across multiple device types. This rendering technique involves translating the input content into a device independent color space, mapping colors that can't be represented by the output device, and then mapping the mapped device independent values into a device specific color space. We will be focusing our study on the conversion from device independent values to an output device, specifically a printer.

2.3.1 Color Spaces

In order to understand how and why colors are converted from input to output color spaces, we must first understand what color spaces are and how they interact.

A color model or color space is an abstract mathematical description of colors as tuples of numbers. These are often groupings of three or four

numbers (RGB-red, green, blue or CMY/CMYK-cyan, magenta, yellow, black). Each color model has a gamut or footprint that it can represent with sequences of its tuples [7]. Color spaces can either be absolute (device independent) or relative (device specific).

A color space that has a defined gamut is referred to as an absolute or colorimetric color space, which ensures the reproduction of a consistent color given the same set of tuples. Adobe RGB, sRGB, CIEXYZ, CIELAB, and HSV or HSB color spaces are examples of absolute color spaces. Device specific color spaces, on the other hand, do not guarantee a consistent color response and are typically utilized in reproduction of color either in output devices such as monitors and printers, or input devices such as scanners and cameras. These values are sensitive to the device specific processes, pigments, and noises [5, 7]. These dynamics prevent most input and output devices from utilizing a colorimetric color space directly and as such require characterization or calibration to an absolute color space.

The final detail regarding color spaces is the difference between additive and subtractive color mixing models. In additive (e.g. RGB) color models, light emitted directly from a source or illuminant corresponding to a particular wave-length is *added* to a medium in order to achieve a rendered image. Combinations of varying intensities of the three wave-lengths associated with red, green, and blue are utilized to generate all available colors. This perceived image is relative to a dark or black medium (the absence of light). In subtractive (e.g. CMYK) color models, light emitted from a source is absorbed by ink pigments corresponding to particular wave-lengths. These wave-lengths are *removed* from a reflective medium in order to achieve a rendered image. Combinations of varying densities of the three or more wave-lengths associated with cyan, magenta, yellow and black are utilized to generate all available colors. This perceived image is relative to a light or white substrate (the existence of light). Digital cameras, scanners, and monitors use additive color models, while printers utilize subtractive imaging. Color management systems must utilize both types of color models and require conversion into a device independent color space before a color difference can be measured [7].

2.3.1.1 CIEXYZ and CIELAB Color Spaces

CIEXYZ and CIELAB are the most important color spaces utilized in color management systems, since they reflect an absolute color space with unique properties enabling advanced control methods that would otherwise not be possible.

CIE 1931 XYZ was one of the first mathematically defined color spaces. In order to prevent a phenomenon known as *metamerism* (two colors appearing the same, despite being different) this color space well defines the angles by which the color is to be observed. Thus, CIEXYZ is based on a two degree observer and is recommended for up to a four degree observer. The observation angle determines the minimum range of vision that is utilized in experiencing the color image. Figure 2.3 shows how increasing the observer angle increases the field of view [8].

CIEXYZ is an all positive description, where Y represents the luminosity of the color and X and Z have zero luminance. The tristimulus values are intended to match the human visual system's response to the three primary color receptors, (i.e. a color sensation). The human eye has receptors which detect short wavelengths (420-440nm), medium wavelengths (530-540nm), and long wavelengths (560-580nm). CIEXYZ do not directly correlate with these eye receptors, which are considered roughly red, green, and blue, respectively. The CIEXYZ was designed to perform color matching operations and also to explicitly define a lightness factor (Y), such that X and Z describe the chromaticity (the amount of colorants) [8].

CIE 1976 LAB is a nonlinear remapping of the CIEXYZ color space coordinates, which attempt to create a more perceptually uniform system. That is to say the distance between two stimuli's values is predictive of the perceived color shift between them. This quality makes it very attractive to color management systems, since a *delta* between colors can be computed, whose magnitude reflects the perceived color difference. Short-handed as CIELAB, CIE 1976 LAB is not an absolute color space. In practice however, it is very common to standardize the reference white light to be CIE illuminant D50 (Horizon Lighting). CIELAB values can also be relative to unprinted substrate in certain printing applications. The coordinates of

Figure 2.3: Two and Four Degree Observer Angles

CIELAB represent the lightness of a color (L^* , $\text{Range}(L^*) = [0, 100]$, dark->light), its position between red and green (a^* , $\text{Range}(a^*) = [-128, 127]$), and its position between blue and yellow (b^* , $\text{Range}(b^*) = [-128, 127]$), respectively [8]. The following equation can be utilized to convert CIEXYZ values into CIELAB values, where $\{X_r = 96.42, Y_r = 100.000, Z_r = 82.49\}$ are the reference white medium tristimulus values for the two degree observer under D50 illuminant.

$$\begin{aligned} L^* &= 116 [f(Y/Y_r) - 16] \\ a^* &= 500 [f(X/X_r) - f(Y/Y_r)] \\ b^* &= 200 [f(Y/Y_r) - f(Z/Z_r)] \end{aligned} \quad (2.6)$$

where,

$$f(t) = \begin{cases} t^{1/3} & t > (6/29)^3 \\ \frac{1}{3} \left(\frac{29}{6}\right)^2 t + \frac{4}{29} & \text{otherwise} \end{cases}$$

2.3.1.2 Color Difference in CIELAB

Since CIELAB provides a coordinate system to easily describe the perceived color difference, the Euclidean distance between color values is a natural selection for determining color difference. Such a difference is often termed *deltaE* or ΔE . A $\Delta E \leq 1.5$ is said to be imperceptible by the human eye. Since its original documentation in 1976, further studies have revealed that while CIELAB describes the perceptual differences much more uniformly than does CIEXYZ, CIELAB contains certain non-uniformities [8, 7]. This results in coordinates whose small shifts represent color shifts, while larger shifts do not result in perceivable changes to the color. Updates to the *deltaE* equations have been made in CIE94 and further refined in CIE2000 [8, 7]. Equation 2.7 shows how to compute the color difference or *deltaE* given two colors, $L_{ref}^* a_{ref}^* b_{ref}^*$ and $L^* a^* b^*$.

$$\Delta E_{76ab} = \sqrt{(L_{ref}^* - L^*)^2 + (a_{ref}^* - a^*)^2 + (b_{ref}^* - b^*)^2} \quad (2.7)$$

Color accuracy has many definitions. In order to determine how accurately a color is rendered, tight specifications on the environment must be made. The lighting, reference media (paper properties), color difference equations, and target colors all play a role in determining how accurately colors are being generated. Measuring a set of printed samples with a spectrophotometer against a known set of target values (i.e. an ICC-absolute colorimetric intent) provides the means to determine if color differences are visible to the human eye [9].

A *deltaE*₇₆ of less than 1.5 has been found to be unnoticeable to the human eye. However, when placed next to similar colors of less than 1.5 difference, the human eye can detect the color shifts. In other words the human eye can detect small differences (of less than 1.5 *deltaE*) in colors when those colors are side by side, but when a color has no reference colors, it is more difficult for the human eye to determine the absolute color presented. In addition to having a quirky color similarity sensitivity, the human eye also tends to do more averaging. This attribute makes it harder for the eye to detect color differences in images with high frequency components. Here

it is said that a ΔE_{76} can be roughly 3 before it is noticeably different [9].

2.3.2 Color Measurement

In order to perform color management system functions, a means to measure colors must first be described. In general, the readings of a *colorimetric* device, which generate tristimulus values in an absolute color space (i.e. CIEXYZ or CIELAB) must be used either directly or indirectly. If a non-colorimetric device must be utilized, it can be calibrated by a colorimetric device in order to create a transform that translates its reads into a colorimetric color space.

Colorimetry is the science of quantifying and describing human visual response to light. Colorimetric devices are described by those that are capable of producing tristimulus values in co-ordinance with the CIEXYZ standard. It is possible to create a sensor whose response to light corresponds to the XYZ axis. However, this is not a practical endeavor. Devices that attempt to recreate tristimulus values must sample visual light (400-700nm) at regular intervals. A spectrophotometer measures the spectral data using a large number of narrow band filters, and then takes advantage of standard CIEXYZ curves to calculate XYZ. Typically this involves 16 to 32 filters to recreate the XYZ curves accurately. Research has also shown that as few as 6 or 7 filters can recreate the XYZ axis [10, 11, 12].

Alternatively one can convert or calibrate low-cost sensors or scanners. This can be done by printing a set of patches, reading them with a colorimetric device and with the device to be calibrated (i.e. the low cost sensor). Using regression techniques, a transform from the non-colorimetric device to device independent CIEXYZ or CIELAB can be generated. Often media, due to paper property differences, greatly impacts the non-colorimetric measurement sensitivities. As a result, calibration is required on each media-type. There have also been studies which claim that using the black (K) channel in the color characterization of sensors can greatly improve the quality of the mapping [10, 11, 12]. The form of the output of a calibration technique on RGB scanner data is typically a 3x3 matrix mapping device

Figure 2.4: ICC Profile Work-flow

dependent RGB values to CIEXYZ values.

Low cost sensors and scanners also suffer from forms of *metamerism*. Metamerism occurs when the human eye cannot see a difference, but an input device responds differently to the two printed patches on different media types. This can be handled by getting a scanner which has better designed channel-sensitivities, which are linear combinations of color-matching functions, or by treating these calibrations as unique to a media type. Despite attempts to generalize scanners and digital cameras to prevent metamerism, it seems that custom calibration must occur on each media type [10, 11, 12].

2.3.3 ICC Profile Standard

The International Color Consortium (ICC) specifies a standard *profile format*. A profile represents, either in the form of a matrix or uniformly spaced Look-up-table (LUT), the means to map one device color space into another, ensuring an accurate reproduction of an input image. Color management is a critical element in state of the art color reproduction and enables the conversion from one device's independent color space to another.

The architecture for this profile standard maps all device specific color spaces to and from a *Profile Connection Space*. The profile connection space

is a well-defined color space, specified by CIE 1931 XYZ or CIELAB, with a CIE illuminant D50. Thus all profiles specify their conversion from device specific color spaces to CIELAB or CIEXYZ, which are convertible by a closed form equation [7].

ICC Profiles are utilized when converting from RGB color space to CMYK color space and each profile describes a *rendering intent*. A rendering intent outlines the manner in which the color spaces should be translated, through a process known as *gamut mapping*. Rendering intents are required since it is very rare for the gamut of two devices to be the same. As a result, color trade-offs must be made when rendering colors. A rendering intent defines how to make these trade-offs. For example, color mappings can be translated to ensure preservation of perceptual intent, relative intent, absolute intent, or saturation intent. Perceptual intent keeps the visual relationship between colors consistent, while relative intent utilizes all colors within the output device's color space possible. Absolute intent maps colors outside of a device's color space at extremes (boundaries) of the device's gamut. Finally saturation intent maps all colors to the strongest saturation possible [13, 7, 14].

2.3.4 Printer Characterization / Output Profile Generation

A printer, based on subtractive imaging (i.e. CMY/CMYK color planes), reproduces an image which has an absolute CIELAB from paper at D50 lighting. The mapping from CMY(K) to CIELAB is referred to as the forward printer map. In order to convert between the Profile Connection Space and CMYK values, we must invert the forward printer mapping. This mapping is highly non-linear and since there does not exist a practical model based on first principles, a characterization (system identification) must be made by empirical and interpolation-based approaches [1].

Empirical methods involve the use of a large data set of corresponding input (CMYK) and output (CIELAB) pairs. Such a system views the process as a black box, without regard for how the process will generate the image from the CMYK data. To obtain a sufficiently large set of input-output data, thousands of colors must be printed in the form of test patches and then

read by a colorimetric device. This process occurs on demand since color patches can only be printed while the customer is not utilizing the device (e.g. a diagnostics mode). It would, however, be desirable to constantly perform this generation process, but customers do not desire patches to be printed on their documents. As a result, during run controls maintain process set-points with sensor reads on the printing apparatus (i.e. patches on a photoreceptor). However, these controls do not impact the color management system [5].

The test patch data should be structured (i.e. uniformly span the input or output spaces). In the case of printing devices with three colors (CMY), if the data is properly structured it is possible to generate the forward and inverse printer maps simply by reversing the order of inputs and outputs (CMY->CIELAB, CIELAB->CMY). However, due to non-linearities in the printing process, it is not reasonable to expect both the forward and inverse maps to uniformly sample their respective spaces [1]. In printing devices with more than three colors in their process, it is equally important that the printer's gamut is fully covered and that sensitive areas in the process have sufficient density to properly capture the non-linearities in the process. Optimization of this characterization data set is often dependent on the process and the characterization algorithms chosen. Despite this there are a variety of standard targets that are often utilized [5].

Figure 2.5 shows the work-flow of a CMYK printer which is utilized while rendering an input image or document. While most printers, to save toner or ink and ensure better engine stability, utilize a Black (K) component, the inclusion of the K component introduces many additional issues. These issues include gamut extension/reduction, graininess, and color smoothness. It also creates a many to one CMYK to CIELAB mathematical issue, which must be dealt with prior to the creation of the inverse printer map. As a result, complex (device specific) constraints are often introduced to handle a variety of trade-offs. This process is known as *Gray Component Replacement (GCR)* and *Under Color Removal (UCR)*. Therefore, as seen below, the inverse printer map goes to CMY values, not to CMYK values. The assumption is that these constraints are one-to-one and therefore reversible when the input data for system identification are (CMYK,

Figure 2.5: Rendering Work-flow of a CMYK Printer

CIELAB) values [1].

There are a variety of methods for finding the forward and inverse printer mappings, whose outputs are typically a uniform 3D-LUT. The 3D-LUT can be efficiently interpolated between during the rendering process using Trilinear Interpolation or Tetrahedral Interpolation. Generation of the 3D-LUT can be achieved using Conjugate Gradient Optimization, Shepard Interpolation, Spline fits, Moving Matrix (recursive least squares) Interpolation, Iteratively Clustered Interpolation, and Piece-wise linear modeling techniques using Recursive Least Squares, Neural Networks, Principal Component Analysis, and Stability Control methods [5, 1].

The final input to output ICC Profile, or 3D-LUT must include all color management features to ensure accurate color reproduction at high rendering speeds. These features include the Gray Component Replacement strategy, the Gamut Mapping strategy (rendering intents), and any smoothing required to make the output images pleasing and defect free.

2.3.4.1 Recharacterization Approaches

While the techniques enumerated in Section 2.3.4 are able to perform a printer characterization, they do not take into account printer drift. The study done in attempts to address the printer drift issue by performing *recharacterization* to adapt to print device characteristic changes such as device age, ambient temperature and humidity, or changes in pigment or substrate.

In the system outlined in [15], recharacterization involves printing a small subset (18 or 36 patches) and using this feedback to correct for the current set of characterization errors. Care must be taken when selecting these colors, since their location in the printer gamut drastically impacts

their effectiveness. This new data can then be utilized to either apply a corrective method on the original characterization or be combined into an augmented characterization data set for recharacterization [15].

Experiments were run in the presence of a printer drift environment in the forms of a new cyan pigment and a paper type change [15]. Using this recharacterization procedure improved the accuracy of the characterization with both techniques (error correction and recharacterization). However, the amount of data available (18 or 36 printed patches) was often not sufficient for performing a complete recharacterization routine, and in some cases actually increased the error in reproduction. While the performance improves the quality of the characterization, the procedure is limited due to the lack sufficient data to properly characterize the entire printer gamut.

In [16] a method to correct for errors in a single image is proposed. After printing an image can be scanned and the required corrections computed. Adjustments to the input image can be made, and then the image can be reprinted with the desired color output. While this is a novel technique, since its corrections only apply to single images, it does not handle general printing systems where previous images are not predictive of future content. Moreover, it does not handle how to apply correction for colors that are not in the gamut of the printer.

Chapter 3

The Proposed System: Continuous Recharacterization

The proposed system, referred to hence forth as *Continuous Recharacterization* (CR), utilizes customer image content and an inline scanner system to correct and keep during print color reproduction accurate. The system can be viewed in Figure 3.1. Again, the main goal is to achieve accurate color reproduction despite printer drifts.

The proposed solution can be decomposed into three main problems. First is the issue of high speed rendering of customer image content. Given an input image, in a device specific color space, we must quickly generate the associated CMYK printer values for the entire image prior to sending the image to the printer to print. The second problem, as described in Section 2.3.4, is the generation of a 3D-LUT, which is utilized during high speed rendering. The third and final problem is how to utilize a feedback mechanism based on customer image content to augment the features used to solve the first two problems and ensure color accuracy. This problem will further decompose into a segmentation phase, a capture phase, and a phase that updates the 3D-LUT.

The following sections will demonstrate how the proposed system solved these problems. We will further show in Chapter 4 that without introducing additional patches onto the customer's images, we can keep and maintain color errors within tight tolerances ($\Delta E < 2$), ensuring a pleasing color image output.

Figure 3.1: Proposed System Work-flow

3.1 High Speed Rendering

The process of rendering image content is the primary goal of any color management system, ours included. This process is often implemented in terms of a 3D-LUT. Typically, input device values (RGB) are applied to a previously generated 3D-LUT, creating the appropriate CMYK values for printing. This process can be seen in Figure 3.2. In high speed digital printers this process must be done in real-time, despite high resolution requirements. As a result, rendering is often done via hardware assist.

The foundation of all high-speed rendering is the 3D-LUT. This structure uniformly spans the input space, as described in Section 2.3.4. Ideally every value in the input space would be represented in the 3D-LUT, however this would result in an unacceptably large memory foot-print, since it requires 255^3 (roughly sixteen million) entries in the 3D-LUT to accomplish this. As a result, typical 3D-LUTs contain somewhere between fifteen and eighteen elements per dimension. Thus input values not directly in the 3D-LUT structure must be interpolated from the available data.

Uniform interpolation is a somewhat cheap, yet accurate means of performing high speed rendering. Tri-linear Interpolation and Tetrahedral Interpolation are the most common of these. A comparative performance analysis is provided in Table 3.1 [17].

Figure 3.2: High Speed Rendering Work-flow

	Multiplies	Additions	Comparisons
Tri-linear	7N	7N	0
Tetrahedral	3N	3N+2	2

Table 3.1: Computational Complexity of Tri-linear and Tetrahedral Interpolation

3.1.1 Tri-linear Interpolation

Tri-linear interpolation is a class of multidimensional interpolation techniques that requires the input space be uniformly spaced. The CIELAB to CMYK function space means that we need to find an estimate for $CMYK = f(CIELAB)$, a three to four mapping. The first step in this process involves choosing the *eight* closest 3D-LUT nodes to the CIELAB input values. The input and output values are denoted $n_1, n_2, n_{...}, n_8$ and $p_1, p_2, p_{...}, p_8$ respectively. Bi-linear and linear interpolation techniques are utilized to find an estimate of the CMYK values to be utilized during reproduction [1].

3.1.2 Tetrahedral Interpolation

Tetrahedral Interpolation also requires the 3D-LUT to be uniformly spaced. Like Tri-linear interpolation the first step in the process is to choose the eight closest input values. These are again labeled as n_i and p_i values, as in tri-linear interpolation. However, instead of using bi-linear interpolation to find an estimate for the new input value, the formula is as follows.

$$p = p_0 + p_l \left(\frac{l - l_0}{l_1 - l_0} \right) + p_a \left(\frac{a - a_0}{a_1 - a_0} \right) + p_b \left(\frac{b - b_0}{b_1 - b_0} \right) \quad (3.1)$$

where, p_l, p_a, p_b depend on which tetrahedral the input values reside within [1].

Both of these techniques are widely used in a variety of real-world applications. It must be noted that while Tetrahedral has roughly half the number of required computations, the cost of performing the comparisons to choose the correct tetrahedral can in some applications lead to a large performance penalty, due to modern processor pipe-lining. As a result, Tri-linear interpolation was chosen for implementation in our system. Moreover, since the primary goal of this thesis is the maintenance of a printer characterization, the input space of the 3D-LUT is CIELAB.

3.2 Performance Evaluation of a 3D-LUT

The second problem in creating our novel color management system is the generation of a 3D-LUT. Before moving into how we generated our 3D-LUTs, we must first describe the manner in which we evaluated and compared their performance. Since all color management features are embedded into the 3D-LUT, the ability to render a pleasing output image is ultimately the goal. However, there are additional metrics that can be utilized to estimate the quality of the 3D-LUT. In order to make an objective comparison between techniques, we leveled the playing field between routines. To accomplish this, color management features were implemented independently and each feature was applied with equal tuning in all comparisons. Finally, we created and described a simulated printer from which we obtained performance analysis data.

3.2.1 Performance Metrics

The difference from the desired to the rendered colorimetric values is the most critical and natural metric used for performance of output device characterization systems. The residual error in the characterization data set can be utilized as one such quantitative metric. However, there are a variety of issues with using the characterization data as a metric for overall accuracy of the inverse printer map.

1. Error is only available for training samples.

2. Error is not always available in an absolute color space.
3. Process noise and rendering imperfections seen from run to run are not handled by any characterization techniques.

The first issue is best resolved through the use of two test patterns, or a test pattern which is broken into training and evaluation sections. For printer characterization evaluation one can print a set of known device-dependent (CMYK) patches, and measure the resulting patch with an appropriate sensor (CIELAB values). These will be utilized for generating the inverse printer map. One can then print a set of in-gamut device independent patches (CIELAB values) by passing them through the inverse printer map and then comparing the resulting prints to the target CIELAB values. This can be done either on a simulated model of the printer or directly on the output device. This procedure also avoids the second problem, since printing a second set of patches can take into account the newly acquired characterization information. Thus, error is available in an absolute color space (CIELAB). The final issue is readily handled by averaging the results from a set of characterization procedures to capture temporal and process noises and averaging the results.

The errors created from the above procedure can now be reported. It is common to provide the mean, standard deviation, minimum and maximum values of ΔE for the evaluations patches. A complete statistical description of ΔE can be given in the form of a ΔE histogram. A 3-D grid from CIELAB to ΔE can also be constructed. It must be noted that this static characterization procedure is only as accurate as the measurement system performing the profiling as well as the stability and reproducibility of the process itself. With all this said, analysis of patches is not always a reliable indicator of perceived color quality [5].

In addition to colorimetric accuracy or round trip accuracy, performance of device characterization must include gamut utilization and profile smoothness. In particular the allocation of the black (K) channel can greatly reduce the gamut exposed by the characterized printer, if not handled carefully. Meanwhile, significant shifts in CMYK values seen while traversing a printer's gamut can lead to image contouring (a sharp, sudden color shift in

image content), a major dissatisfier. Therefore, we conclude that the uncovered gamut space and artificial image contours within the printable gamut are important factors to consider when comparing the quality of different printer profiles [1].

Gamut Corner Plots are useful tools that can describe the achievable gamut and its transient smoothness. A CMYK printer has fourteen distinct corners at its gamut boundary. These are the extreme printer values that describe cyan, magenta, yellow, white, red, green, blue, dark cyan, dark magenta, dark yellow, dark black, dark blue, dark green, and dark red. Figure 3.3 shows a selection of these colors and how they relate to the white point. A description of these points and their associated printing values can be seen in Table 3.2.

The gamut utilization and gamut smoothness can be analyzed by moving from the white point to the associated CIELAB value for each of these gamut corners. Each of these lines can be interpolated, resulting in a new CIELAB value. This value can then be applied to the inverse printer map and then printed (either in simulation or on a real machine). The *deltaE* and CMYK responses can be graphed as a function of position (point) from the white point. These graphs can demonstrate the smoothness of the response and also easily describe if the transformation is capable of reaching the gamut boundaries of the output device.

3.2.2 The Importance of a General Color Management Work-flow

While there are a variety of metrics to evaluate color performance, there are even more ways to tune color management systems for optimal color reproduction. Thus in order to make a fair comparison between characterization routines, the implementation and parameterization of various techniques must be kept consistent. For example, in our experiments the size of the output 3D LUT was kept to 15^3 or 3375 points for all routines tested. Moreover, evaluation was made on the *same* data-sets with the *same* color management features enabled. Although it is common to take advantage of information generated as part of some characterization techniques in GCR or Gamut Mapping routines, in order to keep the playing field level, color

Gamut Corner	C	M	Y	K
Cyan	255	0	0	0
Magenta	0	255	0	0
Yellow	0	0	255	0
White	0	0	0	0
Red	0	255	255	0
Green	255	0	255	0
Blue	255	255	0	0
Dark Cyan	255	0	0	255
Dark Magenta	0	255	0	255
Dark Yellow	0	0	255	255
Dark Black	0	0	0	255
Dark Blue	255	255	0	0
Dark Green	255	0	255	255
Dark Red	0	255	255	255

Table 3.2: Gamut Corner Points in CMYK Printer

Figure 3.3: Color Corners

Figure 3.4: Simulated Printer Characterization Work-flow

management features were implemented independent of the characterization techniques explored.

Figure 3.4 shows a high-level diagram representing the simulation environment and Figure 3.5 represents how round trip errors were calculated with the simulated printer. The important thing to outline here is that only the *simulated printer*, not the available characterization data, knows the *real* mapping from CMYK to CIELAB values. That is to say, the input characterization data is only utilized by the Printer Model and all other CIELAB data is then obtained from the Printer Model directly. This is important in this environment, since the Printer Models contain some errors, as shown in Table 3.3. Performance metrics taken from the characterization data provided by file would be misleading, since they would include errors in the printer model. Instead we treat the printer model as the real system, avoiding this complication.

Figure 3.5: Simulated Printer Characterization Performance Analysis Work-flow

3.2.3 Printer Model Generation

In order to evaluate previous and proposed algorithms, given the absence of a physical printer and scanner system, a printer model was utilized. This model provided us with the forward printer mapping: CMYK->CIELAB. Moreover, in order to demonstrate feedback quality over time, a time based drifting printer model was also employed. There are a number of static and dynamic printer models in literature [1, 5, 18, 19]. Unfortunately, static printer models were not a real option due to the need for dynamics. On the other hand, dynamic printer models described in literature demand high computational complexity in addition to involving a large embodiment of work behind creating a functional system. Simple parameterization requires many months of study given a printing process. Due to these high costs, other means had to be found. As a result, we created a dynamic printer model, based on a piecing together a set of static printer models.

3.2.3.1 Static Printer Model

An empirical approach based on available characterization data was created using Least Squares Regression, $Ax = b$. Two prototypes were generated, one based on clustering and one based on a global regression function. The clustered regression took the form seen in equation 3.2, while the global regression took the form seen in 3.3. The clustering regression first performs the *kmeans* clustering algorithm prior to making local models for each cluster. When computing CMYK->CIELAB values, the closest cluster in

the Euclidean distance sense is chosen as the model to estimate the output CIELAB values. The global regression model includes all cross terms of CMYK, although for simplicity reasons these are not all shown in Equation 3.3. As shown in Section 3.3.3.2, the pseudo inverse can be utilized to solve for x^* , the parameterized least squares regression model as in equation 3.5.

The accuracy of both simulated printer models was found to be satisfactory from strictly a *deltaE* perspective. However, the additional computational complexity of the Clustered Regression (to find the appropriate cluster centers via k-means) made it less attractive than the Global Regression Printer Model. Performance of the models is compared in Figures 3.6 and 3.7, and summarized in Table 3.3.

$$A = \begin{bmatrix} 1 & C_0 & M_0 & Y_0 & K_0 & C_0^2 & M_0^2 & Y_0^2 & K_0^2 \\ 1 & C_1 & M_1 & Y_1 & K_1 & C_1^2 & M_1^2 & Y_1^2 & K_1^2 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & C_n & M_n & Y_n & K_n & C_n^2 & M_n^2 & Y_n^2 & K_n^2 \end{bmatrix} \quad (3.2)$$

$$A = \begin{bmatrix} 1 & C_0 & M_0 & Y_0 & K_0 & C_0M_0 & C_0Y_0 & \dots & C_0^2 & M_0^2 & Y_0^2 & K_0^2 \\ 1 & C_1 & M_1 & Y_1 & K_1 & C_1M_1 & C_1Y_1 & \dots & C_1^2 & M_1^2 & Y_1^2 & K_1^2 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & C_n & M_n & Y_n & K_n & C_nM_n & C_nY_n & \dots & C_n^2 & M_n^2 & Y_n^2 & K_n^2 \end{bmatrix} \quad (3.3)$$

$$b = \begin{bmatrix} L_0 & a_0 & b_0 \\ L_1 & a_1 & b_1 \\ \dots & \dots & \dots \\ L_n & a_n & b_n \end{bmatrix} \quad (3.4)$$

$$x^* = A^+b \quad (3.5)$$

3.2.3.2 Dynamic Printer Model

As mentioned, a dynamic printer model was required to demonstrate how current procedures fail to accommodate for transient dynamics seen in color

Figure 3.6: Performance Graphs for Clustered Regression Printer Model (Fogra41L)

Figure 3.7: Performance Graph for Global Regression Printer Model (Fogra41L)

	Global Least Squares (δE)			Clustered Least Squares (δE)		
Data-set	mean	std	max	mean	std	max
FOGRA39L	2.07	1.73	17.41	1.96	1.39	18.98
FOGRA41L	1.87	1.58	13.07	1.62	1.17	12.13
FOGRA42L	1.60	1.34	12.12	1.57	1.24	14.53
FOGRA43L	2.79	2.30	18.88	2.34	1.68	15.34
FOGRA44L	2.90	2.57	20.58	3.57	2.62	24.83

Table 3.3: Performance Comparison of Clustered Regression and Global Regression Printer Models

reproduction systems. However, the cost of creating and parameterizing a complete printer model would be overwhelming. In lieu of this, we generated a dynamic printer model by stringing together static printer models. Given two characterization data sets P_1 and P_2 , where $P_x \in f(cmyk_i \rightarrow lab_i)$, the data from P_1 and P_2 can be reorganized as $f(cmyk_i \rightarrow (lab_{i,1} lab_{i,2}))$. Then, through linear interpolation, additional characterization data sets and thus static printer models as described in 3.2.3.1, can be created by performing equations 3.6-3.8, on each $CMYK_i$ for each intermediate data set, $n \in [1, N - 1]$, for an additional $N - 2$ printer models.

$$\nabla Lab = Lab_i - Lab_{i+1} \quad (3.6)$$

$$\nabla step = \nabla Lab / (N - 1) \quad (3.7)$$

$$Lab_n = Lab_i - \nabla step * n \quad (3.8)$$

This generates new static printer models, which can be iterated through to represent printer shifts over time; each static printer model ($P_1, P_{n_1}, \dots, P_{n_{N-1}}, P_2$) represents a fixed moment in time. Controls for stepping time and resetting time are available in the simulation environment.

3.3 3D-LUT Generation

As mentioned, the second problem in defining the Continuous Recharacterization system is how to generate the original 3D-LUT used in high speed rendering. This process includes four major components or sub-problems. The first, Gray Component Replacement, represents the process by which the black colorant is utilized in the output image. The second problem is defining the input color space, translating it into device independent CIELAB values, and creating a 3D grid structure spanning this input color space. The third problem, gamut mapping, defines how to deal with colors that cannot be generated by the printer. The final problem is how to find the inverse printer map and output a 3D-LUT that contains all these features and creates a smooth, pleasing color image.

As specified in Section 2.3.4, there are a variety of methods available in literature to find the inverse printer map and generate a 3D-LUT. However, most of these do not piece together the entire puzzle. Details regarding how to describe the connection between the source profile and gamut mapping techniques are often not touched upon, nor is there typically an accurate description on how to generally handle GCR strategies when finding the inverse printer model. More often, these practices are dealt with independently, making the act of creating an entire Color Management System difficult. The process we created and implemented can be visualized in Figure 3.8 and is as follows.

1. Filter available characterization data with an appropriate GCR Strategy.
2. Generate a fixed grid structure uniformly spanning the input color space.
3. For each fixed grid point, determine if the input color is inside or outside of the printer's gamut, through the use of a *Gamut Boundary Descriptor*.
4. If the color is out of gamut, perform a gamut mapping algorithm to choose an appropriate in gamut color.

Figure 3.8: 3D-LUT Generation Work-flow

5. Use the filtered characterization data to find a reasonable CMYK value capable of printing the desired color.

This section will look at each subproblem and outline how we handled the required connections in order to implement a successful 3D-LUT generator.

3.3.1 Gray Component Replacement and Under Color Removal (GCR/UCR)

While CMY printers can theoretically reach all visible hues, there are a number of reasons why state-of-the-art printers utilize a K, or black separation. Firstly, the black colorant is significantly cheaper to produce than the other colorants. Secondly, black pigment allows an extension of gamut into darker regions of the visible spectrum that are not possible without it, providing enhanced shadows. Finally, it drastically reduces the amount of toner or ink required to render a color. However, while there are a variety of benefits, there are also a number of difficulties that arise from having an additional channel to control from color management's perspective. Firstly

and most importantly, the printer forward map goes from an $R^3 \rightarrow R^3$ to an $R^4 \rightarrow R^3$ mapping, which introduces duplicate solutions in the printer's inverse map. Additionally, the utilization of the K channel in certain colors (i.e. skin and sky tones) causes a dirty or grainy appearance. As a result the treatment of K is subtle and often requires many rigorous experiments to find the right formulation [5, 1, 20, 21, 22].

3.3.1.1 K-Restricted GCR Strategies

The most successful GCR strategies attempt to determine K based on a set of constraints. These algorithms first find all CMYK values that achieve a desired CIELAB (i.e. the grid point in the 3D-LUT), and then determine the best choice amongst these, based on the set of constraints. This effectively reduces the *many to one* problem introduced by having more than three imaging pigments. These constraints include total area coverage, minimum and maximum K, gamut volume, and spatial artifacts that can hide other image quality defects [5]. These strategies are not made based directly on a function of CIELAB nor on CMYK values, but instead act as a filter *prior* to characterization procedures. Colors deemed similar to each other (e.g. $\text{delta}E < \sim 6$), are filtered to behave with the desired K performance.

3.3.1.2 The K Suppression Strategy

One particular type of K-Restricted GCR Strategy is described in [1] and called the K-Suppression strategy. As mentioned, it is often desirable to prevent black from causing grainy appearances in facial tones and blue sky image content. In order to perform K suppression, based on input CIELAB, a function describing the target K for a given CIELAB value can be utilized and is seen in Equations 3.9 and 3.10 [1].

$$K = \frac{1}{U^\eta e^{-\alpha(a^{*2}+b^{*2})/L^{*2}} + 1} - \frac{1}{2}U^\eta \quad (3.9)$$

$$U = 2 \left(\frac{1}{e^{-\beta(\frac{L^*}{L_0-L^*})^2} + 1} - 0.5 \right) \quad (3.10)$$

Figure 3.9: Neutral Response of K with $\alpha = 0.01$, $\beta = 1.5$, and $\eta = 0.5$ for Various Values of L_0

Here, L_0 is selected to be at the top of the gamut ($L_0 = 100$), α describes how strong the dependence on a^* and b^* are, η determines the shape of the K-function at small levels of luminosity (L^*), and β describes the shape of the K-function for medium range of L^* . This function provides a means to control K as a multidimensional function of CIELAB values, keeping unwanted black colorant from tonal areas such as facial tones as well as light colors that can be produced with many CMYK values. Figures 3.9 and 3.10 show a visual of how CIELAB values impact the target K value. While this procedure constrains the K-function, it allows for the utilization of the entire printer gamut. This is possible because when a CMYK value generates a unique color, no K suppression method is imposed [1].

3.3.1.3 GCR Implementation

The GCR functions implemented in our color management system are very similar to those described in Section 3.3.1.1 [1]. One minor modification to decrease continual reprocessing of similar colors was made in our implementation. Instead of filtering characterization data with respect to the current 3D-LUT node being processed, our implementation did this filtering

Figure 3.10: Chromatic Response of K with $\alpha = 0.01$, $\beta = 1.5$, $\eta = 0.5$, and $L_0 = 102$ for Various Values of L

directly on the characterization data set. Our strategy works as follows:

Each CIELAB value available in the characterization data set is compared to every other color and color similarity (*deltaE*) is determined. If colors are deemed similar, then the GCR Strategy is employed.

A *deltaE* of roughly 5 units was found to be satisfactory for similarity in our experiments to ensure that the strategy provided the desired output but at the same time did not suppress the ability to utilize the printer's entire gamut. Moreover, this strategy has all the benefits described in 3.3.1.1, but reduces the over-all cost of performing the GCR strategy by employing the process only once for any pair of similar colors.

The following sections describe the available strategies in our implementation. We designed our color management system to allow the user to select which of these strategies to employ on a case by case basis. These can be chosen from in order to optimize the use of the K separation, and selecting them will depend on the particular printer and printing application.

Maximum-K The Maximum-K GCR function is useful for reducing ink and toner costs, since less pigment is required. When asked to choose between

two similar colors, it chooses the color with the most K. However, this technique results in the utilization of black colorant immediately, even in light areas of the printer's gamut. This can lead to grainy image appearances. Figure 3.11 shows the CMYK response for the Maximum-K strategy.

Minimum-K The Minimum-K strategy is useful to reduce as much grainy appearance from image content as possible. When asked to choose between two similar colors, it chooses the color with the least K. However, it does so at the cost of high utilization of CMY pigments in the dark regions of the gamut. Figure 3.12 shows the CMYK response for the Minimum-K strategy.

K-Suppression The K-Suppression strategy represents the most control over the choice of K levels. As described in Section 3.3.1.1, it defines a function of CIELAB values that determines the target K value for any CIELAB value. We used values of $\beta = 1.5$, $\alpha = 0.01$, $\eta = 0.5$, and $L_0 = 102$. This is plainly seen in Figure 3.13, where the characteristics of both the Minimum-K and Maximum-K strategies exist in the light and dark regions of the color sweep, respectively. The K-Suppression method demonstrates a controllable suppression technique for limiting K in low illuminosity colors and for ink savings in darker colors. This ensures that dark specks are not visible in facial tones and other light sensitive colors, but not at the expense of ink utilization across the entire printer gamut. Because of this dual feature, it was heavily utilized in our experimentation as the GCR strategy of choice.

3.3.2 Source Profile and 3D-LUT Structure Generation

The second problem in generating a 3D-LUT is defining the input side of the 3D-LUT. This typically includes applying a source profile on the input color space. The goal of the source profile is to capture the colorimetric intent of the image to be printed. For our purposes this involved the translation of RGB image content into CIELAB. Since our focus was on printer characterization and the generation of an output profile, image content was

a) Dark Blue

b) Dark Black

Figure 3.11: Maximum-K GCR Dark Blue and Dark Black Corner Plots

a) Dark Blue

b) Dark Black

Figure 3.12: Minimum-K GCR Dark Blue and Dark Black Corner Plots

a) Dark Blue

b) Dark Black

Figure 3.13: K-Constrained GCR Dark Blue and Dark Black Corner Plots

directly translated into CIEXYZ by a closed form solution, not taking into account any device specific parameters. Then a reference white point was used to convert into CIELAB values, as in Equation 2.6. This conversion system was not integrated into the 3D-LUT as is typical in color management systems. It was utilized here to allow us to focus on the rich set of other problems in creating the Continuous Recharacterization system. Hence the input color space to our 3D-LUT is CIELAB.

Creation of a fixed grid structure involves allocating sufficient memory to represent both the input values and the associated output values. Both of these allocations have the same 3D dimensions, $M \times N \times L$, although the output space does not have to be three dimensional, as is the case with CMYK printer values. For our purposes, since optimization of the 3D-LUT for our particular application was not a concern, we did not choose to emphasize any dimension. As such, we set $M = N = L = 15$, which represents a somewhat smaller 3D-LUT than high-end production devices. This smaller 3D-LUT size also excellerates the process of generating and updating the 3D-LUT, since our goal is a real-time feedback mechanism. Each element in the grid structure is then assigned an appropriate CIELAB value and is ready to be further processed with any gamut mapping required. Eventually, this is used to calculate the associated CMYK value.

3.3.3 Gamut Mapping

With colorimetric image content available, we move on to the third phase of generating a 3D-LUT, where we take into account the size and shape differences in the input color space and the output color space. However, before tackling Gamut Mapping, we must first find a means to describe the gamut of the printer. This will allow us to determine for example, if a color is in the gamut or out of the gamut of a printer.

3.3.3.1 Gamut Boundary Descriptors

The range of producible colors for an output device can be thought of as a volume in 3D color space. Specific characteristics of the printing process,

such as ink pigments, half-toning routines, and substrate, can all change the size and shape of the printable color gamut [23]. The importance of Gamut Boundary Descriptors (GBD) is best understood in the context of the Gamut Mapping problem. In many Gamut Mapping Algorithms (GMA), such as clipping methods, the first question asked is whether or not an input color is in the output gamut. Improper classification into these two groups results in major image quality defects, such as loss of detail, hue color shifts, loss of chroma, or shifts in lightness. Thus an accurate GBD is required to implement a pleasing gamut mapping strategy [24, 23, 25, 26, 27, 1].

In some iterative characterization techniques, since convergence criterion are on the basis of a target ΔE , it is possible to use accuracy measurements as a determinant of in gamut and out of gamut [1]. However, this limits characterization techniques to those that use iterative processes. As a result, it is desirable to obtain a gamut boundary descriptor that is independent of the chosen characterization techniques [23].

Early attempts at the development of a gamut boundary descriptor were based on primary and secondary coordinates in device specific color spaces such as CMY. Straight lines were drawn between each external color, producing a polygon. This is similar to the graphic in Figure 3.3 on page 29. The resulting gamut volume can sometimes be accurate, but in many cases, due to non-linearities in the printing process, it is not a reasonable estimate of the printer's gamut. Further generalization of this strategy involved constraining the gamut volume to any CMY coordinates containing the minimum or maximum printing values. While this provided the correct shape for the output gamut volume, it is limited due to the movement away from three component printers. Thus most gamut boundary descriptors utilize colorimetric values, such as CIELAB, as their input coordinate space. Known algorithms such as the Convex Hull routine can be utilized to describe the gamut surface assuming the gamut boundary is itself convex [23].

However, if the surface descriptor has concavities, they will be masked by the polygon mesh used to define the gamut boundary. Thus, ray-tracing techniques similar to those utilized in computer graphics systems can be

employed to better describe the gamut surface. To accomplish this, colorimetric CIELAB Cartesian coordinates are converted into cylindrical coordinates or CIELCh coordinates in equations (3.11)-(3.13). Triangulation is utilized to create a regularly shaped grid structure. This process of surface fitting has the advantage of functioning well regardless of the shape of the gamut [25].

$$L^* = L^* \quad (3.11)$$

$$C_{ab} = \sqrt{a^{*2} + b^{*2}} \quad (3.12)$$

$$h_{ab} = \tan^{-1}(b/a) \quad (3.13)$$

Equations 3.14 through 3.16 provide the means to invert back out of cylindrical coordinates.

$$L^* = L^* \quad (3.14)$$

$$a^* = C_{ab} * \cos(h_{ab}) \quad (3.15)$$

$$b^* = C_{ab} * \sin(h_{ab}) \quad (3.16)$$

3.3.3.2 Segment Maxima Method

A particular Gamut Boundary Descriptor, called the Segment Maxima Method, is described and utilized in [26] and [27]. This method is somewhat similar to that described in [25] with additional rigor and detail, such that its application is natural and accurate. Additionally the technique is extended with the spherical coordinate system, rather than the cylindrical. This can be seen in Figure 3.14. The result is a natural boundary descriptor, the radius of the corresponding CIELAB values. Due to its generality and operation on a colorimetric space, this procedure can determine the gamut of devices as well as that of image content, described in colorimetric spaces.

The first step in the procedure is to define the values r, α, θ for all CIELAB values available to describe the gamut, as in equations (3.17)-(3.19), where E is defined as the center of the gamut CIELAB values $[50, 0, 0]$. A matrix

covering the α, θ spaces is then generated covering uniformly the range of those variables $(0, 2\pi), (0, \pi)$, respectively. A reasonable number of entries must be utilized to capture the essence of the gamut boundary; however, utilization of too many points causes defects in the gamut description. A practical number of elements per variable is 16, thus leading to a 256 element matrix. This matrix does not only contain the output point r , but it also contains the *actual* values of α, θ . Storing these values along with their associated CIELAB values leads to an accurate boundary description without interpolation techniques. For each entry in the matrix, the values with the largest r values are saved. Entries in the matrix where data does not exist utilize interpolation from neighboring points to estimate the gamut boundary [26].

$$r = \sqrt{(L^* - L_E^*)^2 + (a^* - a_E^*)^2 + (b^* - b_E^*)^2} \quad (3.17)$$

$$\alpha = \tan^{-1}((b^* - b_E^*)/(a^* - a_E^*)) \quad (3.18)$$

$$\theta = \cos^{-1}((L^* - L_E^*)/r) \quad (3.19)$$

Equations 3.20 through 3.22 provide the means to convert back to CIELAB values.

$$L^* = r * \cos(\theta) + L_E^* \quad (3.20)$$

$$a^* = r * \sin(\theta) * \cos(\alpha) + a_E^* \quad (3.21)$$

$$b^* = r * \sin(\alpha) * \sin(\theta) + b_E^* \quad (3.22)$$

Using the described GBD to determine if a test color is in gamut or out of gamut requires searching the GBD Matrix for intersection points. This is done by checking adjacent points from the GBD Matrix, a, b, c and determining if the intersection point from a test color to a target color is within the triangle formed by a, b, c . This procedure can be accelerated by checking only those points close to the test color in α, θ coordinates. Practice shows that a 5x5 sub-matrix is required for accurate intersection point determination [26].

Figure 3.14: Spherical Coordinate System for CIELAB

Determining intersection requires calculating the intersection of 3D lines and planes. Section 3.3.3.2 is devoted to the notation, equations and practical implementation details. If the determined point of intersection has an r_{int} value greater than or equal to the r_{test} value of the test color, the test color is in gamut; otherwise it is considered out of gamut. Note that these equations must be performed in a Cartesian colorimetric color space.

Mathematical Background: Intersection of a Line and a Plane Given two points l_a, l_b describing a line or three points P_0, P_1, P_2 to describe a plane, Equations (3.23) and (3.24) are functions to define a line or plane, respectively [28, 29].

$$P = l_a + (l_b - l_a)t \quad (3.23)$$

$$Q = P_0 + (P_1 - P_0)u + (P_2 - P_0)v \quad (3.24)$$

Combining these equations we get a formula for computing P , the intersection of the described line and plane. This can be seen in equation (3.25).

$$l_a + (l_b - l_a)t = P_0 + (P_1 - P_0)u + (P_2 - P_0)v \quad (3.25)$$

This can be further simplified in order to render it accessible in a matrix form in equation (3.26).

$$l_a + P_0 = (l_a - l_b)t + (P_1 - P_0)u + (P_2 - P_0)v \quad (3.26)$$

Thus the equations reduce to finding the inverse of a matrix and finding the parametric values of $[t, u, v]$ as seen in equations (3.27)-(3.28).

$$\begin{bmatrix} x_a - x_0 \\ y_a - y_0 \\ z_a - z_0 \end{bmatrix} = \begin{bmatrix} x_a - x_b & x_1 - x_0 & x_2 - x_0 \\ y_a - y_b & y_1 - y_0 & y_2 - y_0 \\ z_a - z_b & z_1 - z_0 & z_2 - z_0 \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} \quad (3.27)$$

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \begin{bmatrix} x_a - x_b & x_1 - x_0 & x_2 - x_0 \\ y_a - y_b & y_1 - y_0 & y_2 - y_0 \\ z_a - z_b & z_1 - z_0 & z_2 - z_0 \end{bmatrix}^{-1} \begin{bmatrix} x_a - x_0 \\ y_a - y_0 \\ z_a - z_0 \end{bmatrix} \quad (3.28)$$

Solving for $[t, u, v]$ provides a means to back calculate the point of intersection with equation (3.23). The values of $[t, u, v]$ also provide insight into the manner in which the intersection takes place. In particular if $t \in [0, 1]$, then the intersection occurs between l_a and l_b . Additionally if $u, v \in [0, 1]$ and $(u + v) \leq 1$, the intersection of the line and plane occurs inside the triangle spanned by P_0, P_1, P_2 [28, 29, 30].

Mathematical Background: Computing the Inverse of a Matrix Via Singular Value Decomposition (SVD) Computing the inverse of a matrix can bring with it many of the complexities of linear algebra. When attempting to invert a matrix A , it is possible that A not be square. Furthermore, the rows or columns of A may not be linearly independent. This means that the computations required to compute A^{-1} are ill-posed and cannot be computed directly. Thus it would be desirable to have a tool that enables detection of an ill-posed

matrix as well as a means to compute the inverse. The SVD provides the means to do just this. Equation (3.29) provides the basics of the SVD. Given any non-zero $m \times n$ matrix A , whose rank $r > 0$, the following factorization can be made, where P is an $m \times n$ column orthogonal matrix, Σ is an $n \times n$ diagonal matrix with positive or zero elements, and Q is the transpose of an $n \times n$ orthogonal matrix:

$$A = P\Sigma Q^T \quad (3.29)$$

This decomposition of A is defined such that $P^T P = Q^T Q = I$. When matrix A is square the inverse of A is seen in equation (3.30).

$$A^{-1} = Q\Sigma^{-1}P^T \quad (3.30)$$

When A is not square the SVD defines the *pseudo-inverse*, A^+ . This turns out to be equivalent to the least squares solution as seen in equation (3.31), where $x^* = A^+b$ is the least squares solution to the linear system spanned by A .

$$A^+ = (A^T A)^{-1} A^T = Q\Sigma^{-1}P^T \quad (3.31)$$

The SVD also provides the means to identify singularity in the decomposition of A . This is done through the *condition number* defined as the ratio between the largest and smallest elements of the Σ matrix. If $k(A) = \sigma_1/\sigma_n = \infty$, or more practically as it grows past single precision floating point accuracy, the condition number is said to be *ill-conditioned*, making the solution of $Ax = b$ more difficult to solve.

As a result of this powerful tool, a user can quickly and easily compute and detect problems with taking the inverse of both a square matrix or an over-determined system [31, 30].

3.3.3.3 Gamut Boundary Implementation

The implementation of a Gamut Boundary Descriptor in our color management system was done through the Segment Maxima technique as described in 3.3.3.2. This technique was chosen because it can be performed without

device specific values, and thus can be performed on image content as well as printer gamuts. Moreover, the resulting gamut boundary descriptor is easily utilized to perform a large number of gamut mapping strategies, which will be addressed in Section 3.3.3.4. These elements in combination make it a very attractive means for representing the printer's 3D gamut.

Our implementation, while it is based directly off work done in [26, 27], takes advantage of a few mathematical simplifications not provided in the original papers. In [26, 27], detection of intersection is done by computing and comparing the area of the triangles created at plane and line intersections. This was found to be merely overhead in this implementation, which utilizes the resulting parameterization of $[t, u, v]$ to determine where the intersection point is relative to the line and plane described. See Section 3.3.3.2 for details.

A visualization of a printer's gamut as compared to an image is shown in Figure 3.16. This graphic shows how the shape of the printer's gamut can be drastically different than that of an image. The accuracy and dependability of this boundary descriptor will better be determined through its use in Gamut Mapping in Section 3.3.3.5.

3.3.3.4 Gamut Mapping Strategies

Now that we have established a gamut boundary descriptor, the next step is to perform a particular gamut mapping algorithm. The remainder of this section provides details on various strategies and how they are implemented in our color management system.

A *color reproduction medium* is a device for displaying or capturing color information. These typically include a monitor, digital camera, scanner or printer. In the case of a printer, however, one must specify not just the printer, but also the colorants (pigments) and substrate. Each device has a *color gamut*, or a range of achievable colors [26, 27, 32]. These gamut sizes and shapes are inevitably different due to the physical properties of each color reproduction medium. Rather than not image them, unachievable colors must be mapped to achievable colors, to ensure a pleasing image reproduction. For example, imaging a saturated blue seen on a monitor with a

a) Original Aqua Image

b) Original Aqua Image Gamut

c) FOGRA39 Gamut

Figure 3.15: Printer Gamut vs Aqua Image Gamut

a) Original Ski Image

Gamut Descriptor

b) Sky Image Gamut

Gamut Descriptor

c) FOGRA39 Gamut

Figure 3.16: Printer Gamut vs Ski Image Gamut

typical CMYK printer will likely fail, and thus must be printed using some combination of printer pigments. Conversely, the bright cyan available on a printer may not be achievable on a typical monitor. These scenarios are described as a *gamut mismatch*. *Gamut Mapping Algorithms* (GMA) define the process for determining how to map input device gamut colors to output device gamut colors, in an attempt to preserve the original appearance of the target image. There are many techniques utilized for gamut mapping; however, none of them are globally satisfactory in producing pleasing color, contrast, lightness, chroma, and hue across all images and devices [33, 34, 35, 36, 37, 38, 1].

Early attempts at gamut mapping were broken into two categories: clipping and compression. Let the input gamut be represented by G_i and the output gamut be represented by G_o . Equation (3.32) represents all colors in the input gamut which are not in the output gamut, where G_o^C represents all colors not in the output gamut.

$$G_{i-o} = G_i \cap G_o^C \quad (3.32)$$

Clipping Techniques Clipping algorithms map colors in G_{i-o} to the surface of the gamut using a set of constraints to guide where on the gamut boundary they will land, as seen in equation (3.33).

$$C_0 \begin{cases} C_i & C_i \in G_{i \cap o} \\ C_{mapped} & C_i \in G_{i-o} \end{cases} \quad (3.33)$$

Closest point or *minDeltaE* clipping technique moves out of gamut colors towards the closest point on the gamut boundary. This can be seen in Figure 3.17.

Straight Clipping keeps hue and lightness constant at the expense of chroma, by moving all out of gamut colors towards the neutral L^* axis. This can be seen in Figure 3.18.

Centroid Clipping allows shifts in lightness and chroma but keeps hue constant by moving all out of gamut colors toward the same neutral point or range. This can be seen in Figure 3.19.

Figure 3.17: Closest Point Gamut Mapping Algorithm

Figure 3.18: Straight Clipping Gamut Mapping Algorithm

Figure 3.19: Centroid Clipping Gamut Mapping Algorithm

The primary disadvantage of all clipping techniques is that *blocking* may occur. Blocking occurs when colors in G_{i-o} are mapped to the same color in $G_{i \cap o}$. This may result in a loss of detail in image regions where blocking occurs. This is particularly a problem in closest point clipping. Despite this major issue, clipping techniques are often the favorites in side-by-side comparisons since in-gamut colors are not shifted [33, 34, 36].

Compression Techniques Compression techniques compress the colors of G_i to fit within $G_{i \cap o}$. These techniques fix or compress certain image qualities such as lightness, hue, or chroma in order to achieve the required gamut mapping. This can be done dependent of the image content or independent of the image content. The advantage of using image content is that less compression is required when not performed in a global sense. However, this requires a search through the entire image content to find the extremes. For example, one may perform L^* compression with equation 3.34, where $L_{out-min}^*$ represents the output device's minimum luminosity value.

$$L_{out}^* = \frac{100 - L_{out-min}^*}{100} * L_{in}^* + L_{out-min}^* \quad (3.34)$$

Alternatively one may weight compression ratios based on chroma values in the desire to maintain the trade-off between shadow and chroma. The main advantage of compression techniques is that blocking does not occur;

however this is done at the expense of shifting colors within G_{ino} .

It is also common to put compression and clipping techniques together in tandem. For example L^* compression could be followed by either closest point or straight clipping techniques. Systems of clipping and compression attempt to better control the trade-offs associated with gamut mapping strategies [33, 34, 36].

Choosing a Color Space The space in which the gamut is represented and gamut mapping is performed can greatly influence the quality of color reproduction. It turns out that CIELAB and CIELuv each have significant non-linearities in the blue and yellow regions. Thus linear interpolation can cause hue shifts in these regions, especially as the distance to the gamut boundary increases. Several attempts have been made at a more linearized color system. A common relinearized color space is called the mLAB system proposed in [35, 37]. This color space keeps the lightness factor from CIELAB space, but remaps a^* and b^* to be linearized. It is based off the Munsell color system. The mLAB space preserves hue to much greater accuracy when performing gamut mapping transformations.

3.3.3.5 Implemented Gamut Mapping Strategies

The versions of gamut mapping that we implemented are based on previous works and include only image independent clipping processes as in Section 3.3.3.4. Our implementation provides the means to select which gamut mapping strategy to employ during image rendering. Multiple images were tested with each process; each showed different levels of success. These will be assessed in their respective sections. The original images seen in Figures 3.15 and 3.16 were chosen due to their large out of gamut content and also for their difference in high frequency components. They will be utilized to demonstrate the qualities of the various gamut mapping algorithms. Overall it will be clear that no one pixel-wise strategy is optimized for all image contents. In all cases, including the blocking defects created, the algorithms performed as expected.

Figure 3.20: Gamut of Aqua (a) and Ski (b) Images after Centroid Gamut Mapping

Closest Point Clipping The first implemented technique we used is closest point clipping. It has the advantage in high-frequency, multi-colored images. However, when similar out of gamut colors occur in high-frequency image content, it is common for closest point clipping to have large occurrences of the *blocking* defect. Figure 3.21 shows the results of gamut mapping using the closet point algorithm. Notice in the Aqua image not only were the blues of the original image purple, but also the high frequency bubbles and specks found in the original image were gone. Both of these are obviously objectionable. The first issue is a linearization problem in the implementation of all the gamut mapping techniques seen in this paper, since they were done directly in CIELAB space, instead of a more linear one as described in Section 3.3.3.4, the mLAB color space. However, the results of gamut mapping the Ski image with closest point clipping were highly successful. The image looked almost as good as the original, with the exception of some hue shifts seen in the color pallets.

Centroid Clipping The second implemented technique is centroid clipping. This maintains hue, clips chroma, and moves lightness towards the center of the gamut in order to create the output image. This is advantageous in many ways since it avoids the hue shifts which closest point gamut mapping can introduce. Figure 3.22 shows the results of gamut mapping using centroid clipping. As compared to the results of the closet point algorithm, the high frequency details of the Aqua image were not lost. However, you can see in the Aqua image, as compared to the original, some of the lightness/darkness shifted, but when compared against the straight clipping results, chroma was clipped less.

Straight Clipping The final implemented Gamut Mapping strategy is Straight Clipping. It maintains lightness (where possible) and hue, and clips only chroma. This procedure performed as well on the Aqua image in maintaining the details of the water, but resulting chroma was far less than the original. This can be seen in the orange of the fish. However, due to the white point issues with straight clipping without L* Compression, one of the spots had a darker than desired edge to it.

(a)

(b)

Figure 3.21: Aqua (a) and Ski (b) Images with Closest Point GMA

(a)

(b)

Figure 3.22: Aqua (a) and Ski (b) Images with Centroid Clipping GMA

(a)

(b)

Figure 3.23: Aqua (a) and Ski (b) Image with Straight Clipping GMA

3.3.3.6 Assessment of the Gamut Boundary Descriptor (GBD)

Assessment of our GBD descriptor was left until after discussing the results of Gamut Mapping since the output of GBD is utilized in GMA. The most natural of the GMAs in regards to estimating the gamut surface is the Centroid Clipping procedure. In this GMA algorithm, the centroid mapped to is the center of the gamut at $[50, 0, 0]$, the same utilized by the Segment Maxima technique. In order to determine how well the gamut surface can be utilized, the output images after centroid clipping were sampled to find their gamut. The results are seen in Figure 3.20. The goal of any clipping GMA algorithm is to constrain all out-of-gamut colors to the gamut boundary. Thus, it is natural that the union of the printer gamut and original image be representative of the best case. As is clear in Figure 3.20, the Aqua image was reduced to just inside the gamut of the printer, as was the Ski image. Some of the non-linearities in the blue region of the CIELAB space were clipped further inside the gamut boundary, but for the most part the gamut of the output image was properly clipped to the edge of the gamut boundary. Therefore the Gamut Boundary Descriptor generated from the Segment Maxima procedure was highly useful in finding the gamut boundary for use in Gamut Mapping Algorithms.

3.3.4 Characterization Techniques

The fourth and final phase in our 3D-LUT generation process finds a mapping from a profile connection space (CIELAB or CIEXYZ) to a device dependent color space (CMYK). This section goes into the details of three such previous strategies that were adapted for our purposes of generating a 3D-LUT from CIELAB to CMYK: Moving Matrix Interpolation (MMI), Adaptive Local Linear Regression (ALLR), and Iteratively Clustered Interpolation (ICI). We then outline a means to optimize their performance in combining and extending MMI and ALLR into our own characterization strategy.

3.3.4.1 Mathematical Background: Implicit Function Theorem

Defining the inverse printer map requires that we define a function from CIELAB- $\rightarrow$ CMYK based on empirical data. Mathematically speaking this requires taking advantage of piece-wise linear function theory and local function theory, which can be summarized by the Implicit Function Theorem [1, 39].

The Implicit Function Theorem allows relations to be converted to functions. Where a single function may not be available, restrictions placed on the domain of the relation may provide sufficient condition in order to define such a function. Specifically it looks at a function $f(x, y) = 0$, where x, y are in U, V respectively and where $0 \in W$, and $\partial f / \partial y \neq 0$ at some point, p , in f . Under these conditions there exists a function g , whose domain is some range about p such that $y = g(x)$ [29].

This property is very useful when it is used in applications where only local solutions and accuracy are required [40].

This property also has a relation to the *Jacobian* matrix, also called the differential or total derivative of a function. The Jacobian is a matrix of partial derivatives describing a function. The implicit function theorem states that if a Jacobian matrix exists describing the partial derivatives of the outputs and is invertible, then we can locally solve for g [29, 30].

3.3.4.2 Moving Matrix Interpolation (MMI)

The Moving Matrix Interpolation technique attempts to estimate associated CMY(K) values for a fixed grid CIELAB or CIEXYZ 3D LUT. It takes advantage of a weighted least-square regression technique.

Balasubramanian and Maltz first describe the use of weighted least squares for use in printer characterization in [41]. Their focus, however, was not in generating the inverse printer transformation, but instead modeling errors from LUTs created from printer models and making updates to the printer's image LUTs. This technique of refining the model based LUT with measured data by occasionally printing a subset of the printer's gamut shows great success in performance improvement as compared to those methods

without refinement. However, it has been found that the accuracy of empirical models outperform all attempts at closed form solutions for estimating the inverse printer map.

As a result, the Moving Matrix Interpolation algorithm had to be adapted for our purposes and implementation to map CIELAB to CMYK. As applied to finding the inverse printer map, Moving Matrix Interpolation works as follows:

For each point x_{lut} in the 3D LUT, compute w_i for each $x_i \in P_{lab}$, where P_{lab} are the available post-GCR filtered characterization data in a colorimetric space, as in equation 3.35.

$$w_i = \frac{1}{||x_{lut} - x_i||^\mu + \varepsilon} \quad (3.35)$$

Generate a regression matrix A'_{lut} , such that $b'_{lut} = A'_{lut}\theta_{lut}$, $A'_{lut} = W_{lut}A_{lut}$, and $b'_{lut} = W_{lut}b_{lut}$, where the matrix A_{lut} , b_{lut} , and W_{lut} are seen in equations 3.36, 3.37, and 3.38, respectively.

$$A_{lut} = \begin{bmatrix} 1 & L_0^* & a_0^* & b_0^* \\ 1 & L_1^* & a_1^* & b_1^* \\ \dots & \dots & \dots & \dots \\ 1 & L_n^* & a_n^* & b_n^* \end{bmatrix} \quad (3.36)$$

$$b_{lut} = \begin{bmatrix} C_0 & M_0 & Y_0 & K_0 \\ C_1 & M_1 & Y_1 & K_1 \\ \dots & \dots & \dots & \dots \\ C_n & M_n & Y_n & K_n \end{bmatrix} \quad (3.37)$$

$$W_{lut} = \begin{bmatrix} w_i & 0 & 0 & 0 \\ 0 & w_{i+1} & 0 & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & w_n \end{bmatrix} \quad (3.38)$$

Use the pseudo-inverse as described in 3.3.3.2 to solve for θ_{lut} via equation 3.39.

$$\theta_{lut} = A'^+_{lut} b'_{lut} \quad (3.39)$$

Figure 3.24: ICI Augmented Printer Concept

Now back substitute the target x_{lut} into equation 3.40 to obtain the appropriate CMYK values to store in the 3D LUT.

$$CMYK_{lut} = \theta_{lut} x_{lut} \quad (3.40)$$

The parameterization of μ, ε impact the locality of the regression. Larger values of μ and smaller values of ε result in an emphasis towards local behavior [41, 1]. In our simulations reasonable behavior of w_i was found with $u = 2$ and $\varepsilon = 0.01$. Performance of this routine will be presented in Chapter 4.

3.3.4.3 Iteratively Clustered Interpolation (ICI)

The ICI algorithm is a gradient based iterative optimization method. It too can be applied to finding an inverse printer map or as in the case of [42] CIELAB->CIELAB', which remaps a printer to print the desired color sets. The problem is defined as finding the inverse map of P , $\hat{P}$ such that $P(\hat{P})$ is as close as possible to the identity matrix in the Euclidean distance sense [42].

Again for our purposes, this procedure had to be altered to be applied to mapping CIELAB to CMYK.

The algorithm works in multiple phases. Initialization requires that the desired CIELAB (for each point in the 3D LUT) be “clustered.” That is to say, the closest data provided in P to the grid point be selected and sequenced. The closest point is given as the initial estimate. Subsequently, the procedure estimates a new solution, by iteratively computing:

$$y(k+1) = y(k) - \mu J_k^T \{P[y(k)] - x\} \quad (3.41)$$

Here J_k is the Jacobian matrix evaluated using numerical differentiation techniques from the printer forward map P . Iteration stops when k reaches a maximum value or when the $e(k)$ reaches some minimum acceptance value. The current iteration's $\text{delta}E$ is computed by applying the current $y(k)$ to a forward printer model as in 3.42. When iterating, our system utilizes the static printer model described in Section 3.2.3.1 to represent P .

$$e(k) = ||P[y(k)] - x|| \quad (3.42)$$

The parameter μ is critical in ensuring fast convergence for this algorithm. However, to ensure that the iterations converge, $e(k)$ must approach zero as k increases. This requires that all eigenvectors of matrix $(I - \mu J_k J_k^T)$ lie within the unit circle [1, 42]. In the case of finding an inverse printer map, $y(k) = [C \ M \ Y \ K]$, P is the forward mapping from CMYK to CIELAB, and $\hat{P}$ is the inverse mapping from CIELAB to CMYK. Once iteration completes, the final $y(k)$ value is stored as part of the 3D LUT. The aim of the characterization techniques described above is to generate a structure grid mapping CIELAB to CMYK to be interpolated during image rendering.

Performance of this technique will also be presented in Chapter 4.

3.3.4.4 Adaptive Local Linear Regression (ALLR)

Another method used to find the inverse printer map is known as Adaptive Local Linear Regression. It utilizes local learning methods in a unique monitor characterization method. Here local geometry and an *enclosing neighborhood* defines a set of colors that create a convex hull surrounding a CIELAB grid point (in an output 3D LUT). Once the enclosing neighborhood is defined, linear regression is performed to create a local model, which can then be utilized to generate an appropriate RGB value for each grid point in the 3D LUT. When using a convex hull to define the regression system, it has been proven, given some assumptions, that the error in the resulting least squares regression is limited to the measurement noise in the

samples [43].

Forming a neighborhood can be done by two standard methods: either by number of neighbors or by defining an inclusion distance from a target point. In [43], Gupta defines three neighborhood definitions that automatically choose the cluster elements (to be utilized in the linear regression) based on the geometry of the characterization samples. *Natural Neighbors* are defined as points whose cells are adjacent to the test point. *Natural Neighbors Inclusive* is a slight modification to include not just the natural neighbors, but also those points closer than the furthest Natural Neighbor to the test point. *Enclosing k Nearest Neighbors* attempts to prevent over-smoothing by limiting the distance considered within the neighborhood. All of these strategies attempt to find the smallest number of neighbors that defines a convex hull around a particular grid point.

Choosing the right number of neighbors plays a significant role in the resulting accuracy of the regression. When too few elements are in a local cluster, the regression can result in step extrapolations. On the other hand, too many elements in a local cluster can lead to over-smoothing. The optimum neighborhood size has been shown to be roughly 15 for three-dimensional spaces. Since CIELAB is designed specifically to match perception, traditional k-NN can be exercised without additional adaptive metrics [43].

In our implementation of Adaptive Local Linear Regression, we modified Gupta's technique in order to create a static grid structure mapping CIELAB to CMYK. The mechanical procedure for creating a 3D LUT is similar to that described in 3.3.4.2, but was performed with the W matrix set to the identity. K-nearest neighbors was chosen as the enclosing neighborhood to determine membership into the regression matrix A . An examination of the performance of this will be explored in detail in Chapter 4.

3.3.4.5 Continuous Recharacterization (CR) Technique

As is the case with all inverse printer mappings, we desire the ability to find an associated CMYK value for each CIELAB in the 3D-LUT. The intent

of adapting and implementing the above characterization techniques was to assess their abilities and determine their feasibility of being applied to our novel Continuous Recharacterization system. It turns out, however, that almost any chracterization technique can be applied to finding the inverse printer map in the proposed system, as long as it is sufficiently accurate and fast. This implies that generation of new 3D-LUTs can be completed faster than the printer can drift. With that said, we developed one that takes into account the best qualities of the MMI and ALLR methods.

Given an entry in the 3D-LUT, we define inclusion in the matrix A from the available Characterization Data based on the distance from each 3D-LUT grid point. This was done to speed up the computation of k-NN described in 3.3.4.4. A visual of a local model is seen in Figure 3.25. The *Color Inclusion Distance* specifies the maximum distance from the grid point that a color can be to get included in the local model. This number can be grown and shrunk to increase or decrease the smoothness of the color response. In practice a reasonable color inclusion distance is 50, which gives sufficient coverage and smoothness over the grid structure, but at the same time provides the accuracy desired. Each local model is computed via Weighted Least Squares regression based on the colors positioned sufficiently close to the grid point. The model takes on the same form seen in equations 3.35-3.40 utilized by the Moving Matrix routine. The difference from the adapted MMI technique is that not all colors in the available characterization data set impact every node.

Performance of this routine will be presented in Chapter 4.

3.4 Customer Image Feedback with an Inline Scanner

The third and final problem to be solved in creating the Continuous Recharacterization System is how to utilize customer image content to automatically update our rendering process. Again, this is to ensure a continuous pleasing output despite printer drift. This problem further decomposes into a segmentation phase, where suitable areas for capture are located, a capture phase, where printed image content is sampled from a scanner and filtered,

Figure 3.25: Local Model Description

and a phase that saves the filtered data into a *color library* and updates the 3D-LUT.

Figure 3.26 shows the feedback system. We will briefly walk through the data flow outlined in this drawing before visiting the three phases directly. First, the image content in CIELAB space is segmented based on color similarity, and assigned a color number. Areas and colors deemed useful for capturing are then located. The image content is then rendered using the current inverse printer map. The resulting CMYK values sent to the printer for production are saved. After the image is generated, it is scanned by a calibrated in-line scanner. The scanner's RGB values are then translated back to CIELAB space. From here, the measured CIELAB values for each color are filtered to reduce sensing noise. The contone values associated with the scanned area are also filtered. The results for each suitable color in the customer image content are added to the inverse printer map's memory. The new information can then be incorporated into the inverse printer map.

We will show further in Chapter 4 that the new color information was incorporated such that process colors were smooth and accurate, despite process drift. Furthermore, local color updates did not poorly effect the rest of the printer gamut color conversions.

Figure 3.26: Proposed System Work-flow

3.4.1 Selecting Customer Image Areas for Use in Feedback

In order to locate valid spots for sampling printed image content, an understanding for basic image segmentation is required. While object recognition is not the goal here, the techniques utilized in traditional computer vision can be utilized for locating the *color regions* in an image.

3.4.1.1 Image Segmentation using Color Similarity

Segmentation is the process of partitioning image content into a set of segments that have something in common such as color, texture, etc. This is done to reduce the complexity of extracting scene information such as object locations or boundary locations. There is a vast literature on this subject. The most common segmentation approaches include clustering algorithms, histogram based methods, edge detection, region growing, and neural network techniques [44, 45].

In [44, 45], a method to segment the image using color similarity is proposed. These papers outline different means to obtain a dominant set of colors from a scene, thereby reducing the dimensionality of the resulting color space. In [45], this is done directly in the CIELAB space by counting the number of occurrences of each color, and choosing the set of colors that

covers the majority of the image. Each pixel is then assigned to one of these dominant colors, thus reducing the dimensionality of the original image. This is possible since it is commonly known that *natural* scenes contain a relatively small number of colors [44]. Segmentation can be performed based on other features. However, since our focus is on color management we will not pursue this topic further.

3.4.1.2 Assigning a Color Number

The first phase in finding suitable areas for capture is to segment the input image based color similarity as described in Section 3.4.1. Our process assigns color numbers, starting with 1, to each sufficiently different color, in a *deltaE* sense. The algorithm is very straightforward:

1. For each CIELAB value Lab_i in the input image
 - (a) For each previously found color Lab_{found_j} in Lab_{found}
 - i. If $\Delta E(Lab_i, Lab_{found_j}) < colorSimilarity$
 - A. Mark Lab_i with a color number associated with Lab_{found_j}
 - (b) If no Lab_{found_j} meets this criteria, add Lab_i to Lab_{found} , allocating a new color number for Lab_i .

Figure 3.27 shows the original Lena Image and the resulting segmentation. Additionally the algorithm provides a number of colors found. In the case of this Lena image, the result was 657, when *colorSimilarity* was tuned to 5. As seen, color numbers are not assigned to regions, but to colors. Thus there are disjoint sections visible in the segmented image which have the same color number. While this might not be as effective when performing segmentation aimed at object identification, it is a key artifact of this strategy. The useful component utilized from this segmentation strategy is the reduction of dimensionality from three to one in the assignment of color numbers. Thus, future processing can be done quickly.

Once the image is segmented our algorithm searches for sufficiently large enough areas to capture. The algorithm works as follows:

1. For each pixel in the segmented image

(a) Original Lena Image

(b) Segmented Lena based on color similarity

(c) Areas deemed large enough for capture and feedback

Figure 3.27: Segmentation and Capture

- (a) If all adjacent pixels contain the same color number
 - i. Accept this area for capture
- (b) Else reject this area for capture, by marking that pixel with a color number of 0.

The areas still containing color numbers are deemed suitable for capture. Any number of algorithms could be utilized to find reasonable locations to capture. This process was chosen under the assumption that the capture scanner functioned at the same rate as imaging and thus resulted in a common *DPI*. This in reality is probably not true, but changes relating to scanner and imaging frequencies should only change the manner in which we capture the customer image for feedback.

The tuning and choice of *colorSimilarity* is a critical one in this system. In general, decreasing the magnitude of color similarity results in an increase of unique colors in an image. This also drastically increases computation expense, since each discovered color must then be checked against each pixel's color. More importantly, however, is that tuning *colorSimilarity* improperly can utterly destroy the system's ability to function. Figure 3.28(a) shows how choosing a color similarity too large can quickly impact the accuracy of the resulting color capture. This is due to the fact that each assigned color number is treated as the *same* color, resulting in only one mapping (CIELAB->CMYK) after sampling. Thus it is only natural that allowing colors significantly different (from a perceptual perspective) will introduce additional error in the characterization quality. Figure 3.28(b), on the contrary shows that choosing a color similarity too close to zero will result in no areas that can be captured according to the above algorithm, i.e. that adjacent areas must be of the same color number. As *colorSimilarity* approaches zero, so too does the percentage of the image content that can be captured. The data found in Figure 3.28 was roughly interpolated and estimated from a few experiments. The numbers are dependent on image content. In this case it follows the trends utilized with the Lena Image as seen in Figure 3.27.

(a) Best Case deltaE as a Function of Color Similarity Threshold

(b) Percent Area Capturable as a Function of Color Similiarty Threshold

Figure 3.28: Tuning of *colorSimilarity*

3.4.2 Customer Image Capture

The second problem in providing a feedback loop based on customer image content is capturing the already segmented and printed image with an in-line scanner. There are three main components to this part of the problem. First is timing of the capture to ensure that the correct image content is read by the scanner for color feedback. The second is translating the scanner's RGB data into a colorimetric space. The final problem is how to filter the colorimetric data and generate a CIELAB->CMYK mapping for the new data set.

3.4.2.1 Timing the Scanner

There are a variety of methods that can be utilized to time and coordinate capturing image content. These strategies are dependent on the paper feeding system provided in the application machine. For example in a *cut-sheet* product, a paper sensor can be placed a known distance prior to the sensor. Since the paper typically travels at a fixed speed through the machine, scanning can be triggered at the appropriate time. On the other hand, in a *continuous feed* system, one can utilize a mark sensor and time scanning based on frame marks. Alternatively one can tie in the scanner control card to the clock used in imaging. In any case an estimate for when to start scanning can be made. A drawing of the cut-sheet system can be seen in Figure 3.29. Notice that scanning occurs post fusing. In general any capturing to assess image quality parameters must be taken *after* the process has completed. In Xerographic machines this is post fusing.

Typically with systems such as this, a means of accurately finding and tuning the distance from the paper sensor to the scanner (in cut-sheet), or the distance from marks sensor to scanner (continuous feed) must be made. These distance finding routines typically lay down an easily captured image at some known location on the sheet or in the frame. Scanning is triggered to begin *early*. By some algorithm the edge of image content can then be located in the scanned image. Backwards calculating the time delta from sensor toggle to start of scan plus the delay in scan-lines, one can compute the distance from sensor (paper or mark) to the scanner.

Figure 3.29: Positioning and Timing Scanner in Printer

At this point the ability to time the triggering of the scanner based on a sensor transition has been described in detail. However, as sheets and images pass through the printer there will be a large number of sensor transitions. Most of these transitions will be for image content that we are not interested in. Thus a more advanced timing and tracking system is required to schedule a capture of image content, prior to the beginning of imaging. Previous work in [46] proposes a system of image and patch tracking that can easily be extended for the purposes of tracking sheet position and image content relative to the scanner apparatus. We propose that sensor transitions are only monitored in areas roughly close to the scheduled timing by which a sheet of interest is passing the scanning apparatus. Thus scanner and system band-width is only utilized when the image content scanned will be utilized for feedback.

DPI and Registration Once the lead-edge of the customer image can be found, another key aspect is lining up the DPI of the scanner to the DPI of the imaging system. One way to handle this is to ensure the frequency

of the imaging device be the same as the sample rate of the in-line scanner. If this cannot be ensured then image scaling may be required. This may cause changes to the manner in which areas for capture are deemed suitable. Regardless, there will be a method of ensuring only small amounts of pixel bleed that can be utilized in a practical implementation for resolving DPI differences between the imaging and scanning system. Again it is desirable to have the scanner system be as fast or faster than the imaging system. Thus, the concepts described in 3.4.1.2 will ensure that there is no residual content from other color numbers effecting the scanned results for areas deemed capturable.

Registration of image to paper and paper to scanner are left to future work. These could, however, be solved using current Computer Vision feature extraction techniques. This would involve capturing the image, finding the skew, and performing the inverse skew function on the image content scanned.

These issues were not taken into account in our simulation environment. The implemented system assumes that the first imaged pixel lines up with the first scanned pixel and that subsequent pixels occur at equal offsets.

3.4.2.2 Scanner Profile / Colorimetric Captured Image

As was outlined in Section 2.2, scanners sample at a particular rate and spacing. The previous section outlines how we handled this aspect. Once the timing of the scanner is accounted for, one must transfer the scanner data into a colorimetric space. During sampling, analog signals corresponding roughly to red, green, and blue, are quantized and scaled, in order to create the output content. When transferring this content into the colorimetric space, calibration of the scanner is necessary; this is covered by previous works as outlined in Section 2.3.2.

The scanner profile utilized in our system was derived directly from a previously performed calibration routine performed under the assumption of a two degree observer and a D65 illuminant. The generated profile is a 3×3 matrix, rather than a 3D-LUT, containing coefficients for translating

scanner RGB data into CIEXYZ data. This type of system is very common in scanner calibration and its transform is sufficiently accurate for our purposes.

CIELAB data is then computed directly from CIEXYZ, based on white-point compensation. This process occurs on the entire scanned image, providing us with colorimetric image data for the printed image. This is the junction point between the previously calculated color number image (segmented image after filtering) $colorNum_{i,j}$, the colors sent to the printer $CMYK_{imaged_{i,j}}$, and the captured image $Lab_{captured_{i,j}}$. At this point colors deemed suitable for capture (i.e. $colorNum[i][j]! = 0$) are collected. Each color's average CIELAB value and CMYK value utilized in the printing process is found, Lab_{ave_i} and $CMYK_{ave_i}$ respectively.

The definition of *colorSimilarity* from Section 3.4.1.2 also plays a critical role in how output colors are filtered. The procedure outlined uses mere averaging to take what could be thousands of color samples and thousands of CMYK recipes and average them to obtain a single CMYK->CIELAB mapping. This process only works since the size of the region under exploration ($colorSimilarity = 5$) can be estimated closely as a linear system. Thus the procedure of averaging only finds the center point of the linear system, making it a reasonable estimation of the CMYK->CIELAB mapping for this relatively small, almost imperceptibly variant color region. The color similarity feature of segmentation must therefore be kept to within a color difference that is locally linear or the color mapping captured here will not be representative of the forward printer map due to non-linearities in that system.

Once the captured image is available in CIELAB space, the process of updating the color library begins.

3.4.3 Updating the Color Library and the 3D-LUT

This data, Lab_{ave_i} and $CMYK_{ave_i}$ calculated in the previous phase is passed to the color library for selection, in the final phase of updating the 3D-LUT.

Entry Number	L^*	a^*	b^*	C	M	Y	K
...	...	...	...	...	...	...	...
2400	48.4619	62.9944	-16.4801	30	218	6	2
2401	43.8622	70.6673	-15.0643	30	249	9	1
2402	83.6217	-10.0612	-12.5238	52	5	8	3
2403	77.9867	-0.387688	-15.1776	50	24	5	2
2405	72.4509	9.11348	-17.4999	52	52	4	0
...	...	...	...	...	...	...	...

Table 3.4: Color Library Entries

3.4.3.1 Updating the Color Library

We have presented the *color library* in previous sections. Here we will define it as representing the most recent available data from characterization procedures or from scanning customer image content. A short entry sample is shown in Table 3.4. As can be seen, there exists little structure to this table; the fixed structure is handled by the 3D-LUT. The color library merely tracks any available color data presented from characterization or customer feedback sources. Prior to characterization this table is cleared out, since characterization procedures should contain enough colors to cover the entire printer gamut, even if additional holes are created in the table that were previously filled by customer image content.

The updating of the color library is a critical step in our system. We will walk through one such phase, demonstrating how colors are removed and others added.

Let Lab_{lib_i} and $CMYK_{lib_i}$ be CIELAB values and CMYK values already in the color library. The color library update process works as follows:

1. For each $Lab_{ave_i}, CMYK_{ave_i}$
 - (a) For each $Lab_{lib_i}, CMYK_{lib_i}$
 - i. If $\Delta E(Lab_{lib_i}, Lab_{ave_i}) < \max \Delta E(3)$ or $\Delta CMYK(CMYK_{lib_i}, CMYK_{ave_i}) < \max \Delta CMYK(4)$
 - A. Replace $Lab_{lib_i}, CMYK_{lib_i}$ with $Lab_{ave_i}, CMYK_{ave_i}$
 - (b) If no such replacement occurs for Lab_{ave_i} , add $Lab_{ave_i}, CMYK_{ave_i}$ to the color library.

Entry Number	L^*	a^*	b^*	C	M	Y	K
1	48	62	-16	30	225	6	2
2	38	70	-15	30	249	9	2
3	95	-2	1	0	0	0	0

Table 3.5: New Color Data Example

where, DeltaCMYK compares each device value directly without computing a distance function.

As an example let's suppose that there are three colors that could be captured in the customer image. These can be seen in Table 3.5. By following the algorithm supplied above, color library entry 2400 will be replaced by new color entry number 1. This is because their CIELAB values are similar. This captures cases where the printer is attempting to print one color and due to printer drift in a particular separation, the resulting CIELAB value is not reflective of the expected value. As a result, we must conclude that the previous mapping is no longer valid. Additionally, color library entry number 2401 will be replaced by new color entry number 2, since their CMYK values are sufficiently close in accordance with our replacement algorithm. In this case we are trying to ensure that color data does not remain in the library when drastic shifts in the printer take place. This can occur overnight if a printer sits idle. Finally, new color entry number 3 is directly added to the color library without removing any other colors assuming that neither a CIELAB match nor a CMYK match is found. This occurs to fill holes surrounding the characterization data, to ensure that all non-linearities are captured in the characterization procedure. The resulting color library can be seen in Table 3.6.

The choice of *maxDeltaE* and *maxDeltaCMYK* must be values less than *colorSimilarity*, or an equivalent value. This was determined experimentally. As the magnitude of colors thrown out as a result of color library updates interacted with those deemed similar during color segmentation and further removed during Gray Component Replacement, the coverage of the the available characterization data became much more sparse. This led to issues in computing and updating a new 3D-LUT. However, as *maxDeltaE*

Entry Number	L^*	a^*	b^*	C	M	Y	K
...	...	...	...	...	...	...	...
2400	48	62	-16	30	225	6	2
2401	38	70	-15	30	249	9	2
2402	83.6217	-10.0612	-12.5238	52	5	8	3
2403	77.9867	-0.387688	-15.1776	50	24	5	2
2405	72.4509	9.11348	-17.4999	52	52	4	0
...	...	...	...	...	...	...	...
3001	95	-2	1	0	0	0	0

Table 3.6: Color Library Entries After Update

and *maxDeltaCMYK* approach zero, colors remain in the color library, making it impossible for any optimization techniques to capture the essence of printer shift occurring. Thus, similar to the choice of *colorSimilarity*, *maxDeltaE* and *maxDeltaCMYK* represent a trade-off between sparseness of the color library and age of colors in the color library. A visual of this tradeoff is visible in Figure 3.30. This data was estimated under the assumption that the input images contain a large subset of the colors in the printer’s gamut. Actual results would be dependent on the image content being printed.

One of the benefits of this strategy is that over time any holes in the characterization data set that occur due to non-linearities in the printing process will be filled. This occurs because characterization data is typically uniform over the printer’s device specific space, not a colorimetric one. Thus without knowing apriori the rate of change in a colorimetric space for a given set of device specific values, it is difficult to ensure that the printer’s gamut will have sufficient data to accomodate any non-linearities. In this strategy, since we can track colors in a colorimetric manner and obtain data during printing, any holes in colorimetric data can be filled. This further improves the characterization and can be seen in our experimental results when the first feedback mechanism is employed. In Chapter 4 we will see data that supports this improvement over static color management systems.

The number of available colors in a single image capture, with a *colorSimilarity* of 5 or less, is consistently more than those available during characterization. As the DPI of the image content increases, so too do the

Figure 3.30: MaxDeltaE/CMYK Tradeoff

acceptable areas for capture. The vast majority of images tested with this procedure returned over 2,000 colors. This is significant considering colors utilized during characterization are limited to roughly 1,700 due to time constraints and the desire to not waste printer up time and consumables (ink/toner and paper) [5]. The largest difference between the strategies is that in using the customer's image we as system designers have no control over the colors selected. This is a direct outcome of the colors printed by the customer. In Chapter 4 we will show how the quality of the characterization over time is dependent on how much of the printer's gamut is represented in the customer's images.

3.4.3.2 Updating the 3D-LUT

Once the color library is updated, each grid point in the 3D-LUT can be analyzed to determine if it has new data to compute against. This was done by saving which parts of the color library were utilized to compute the local model as defined in Section 3.3.4.5. This required a large memory allocation per node (roughly 1MB for a 15x15x15 3D-LUT) to track which colors were utilized in the previous characterization routine. This data had to then be parsed and cross-checked with the color library to determine if nodes in the 3D-LUT required recomputation. This was determined by checking to see if any colors that are in the node's color inclusion distance, as specified in Section 3.3.4.5, were updated. This filtering process is how we ensured that color shifts in one region of the printer's gamut did not poorly effect those in other regions. However, since the vast majority of image content found in practice contained large color gamuts, most node colors had to be recomputed. Thus the process of tracking which models required updates and which did not was discarded due to its cost in processing time and memory utilization. Nodes that did not receive any new color data recalculated with the same data, resulting in similar, if not identical models as the previous iteration. Thus at each iteration of segment, image, and capture, recharacterization occurs across the entire 3D LUT; the next image is then segmented for another feedback iteration.

Chapter 4

Experimental Results

In order to compare the performance of the proposed system against traditional color management work-flows, a number of experiments were run using the CR algorithm. The following sections discuss these results and compare various strategies. We found that not only is our novel characterization technique more accurate, it can also be augmented by capturing customer image content with an in-line scanner.

4.1 Characterization Performance

As mentioned, the characterization technique chosen is a cross between the moving matrix and the adaptive local linear regression techniques. In order to massively save in computational complexity of choosing k-nn, a simple inclusion distance was chosen instead. When generating a LUT it is desirable to have the fewest neighbors that locally define a concave enclosure for the grid point being calculated, as explained in Section 3.3.4.4. However, due to non-linearities, a distance suitable in one space to capture sufficient data (~ 15 data points) to define the Least Squares regression does not exist unless the inclusion distance is larger. As a result, with an inclusion distance of $\sim 30 \text{ } \delta E$, taking the pseudo-inverse of the regression matrix resulted in condition numbers close to infinity for the yellow corner. This forced the inclusion distance to $50 \text{ } \delta E$. Under these conditions, however, the mean round trip error was found to be over $3 \text{ } \delta E$. Hence, a page from the Moving Matrix Interpolation technique was utilized and weighting was

	MMI			
Data-set	mean (∇E)	std (∇E)	max (∇E)	Time(s)
FOGRA39	2.44	3.05	10.65	10
FOGRA41	1.89	2.20	9.43	10
FOGRA42	2.12	2.80	9.44	11
FOGRA43	1.89	2.61	12/09	11
FOGRA44	2.26	4.77	21.50	10
	ICI			
Data-set	mean (∇E)	std (∇E)	max (∇E)	Time(s)
FOGRA39	2.61	3.52	10.65	6
FOGRA41	2.10	2.18	9.34	7
FOGRA42	2.34	3.16	10.47	6
FOGRA43	2.09	2.72	12.09	6
FOGRA44	2.37	4.95	22.96	7
	ALLR			
Data-set	mean (∇E)	std (∇E)	max (∇E)	Time(s)
FOGRA39	2.62	3.52	11.27	6
FOGRA41	1.99	2.29	9.19	6
FOGRA42	2.25	2.83	9.32	6
FOGRA43	2.06	3.16	11.69	7
FOGRA44	2.27	4.37	26.69	5

Table 4.1: Characterization Technique Performance Summary

introduced. The hope was that while some enclosing neighborhoods contained too many data points, the weighting factor would not over-smooth the local grid point and introduce too much round trip error. This attempt succeeded, with $u = 2$ and $\varepsilon = 0.001$ as in equation 3.35.

Table 4.2 shows a summary of characterization with the proposed Continuous Recharacterization technique. Table 4.1 shows the performance of those techniques which we adapted in Section 3.3.4. Not only did our novel algorithm perform better in all regards than any of the other implemented algorithms, it did so faster. Figure 4.1 provides corner plots for the Dark-Red and Green corner points. These can be compared to those in Figures 4.2, 4.3, and 4.4. Clearly, the accuracy of the proposed algorithm is not paid for with a lack of CMYK smoothness, as evidenced by the data. In conclusion, this technique, even without its ability for feedback, outperformed the other characterization techniques tested.

(a) Round Trip Errors

(b) Corner Plot from White to Green Points

(c) Corner Plot from White to Dark-red Points

Figure 4.1: Continue Recharacterization Characterization Performance (FOGRA39)

(a) Round Trip Errors

(b) Corner Plot from White to Green Points

(c) Corner Plot from White to Dark-red Points

Figure 4.2: Adaptive Local Linear Regression Characterization Performance (FOGRA39)

(a) Round Trip Errors

(b) Corner Plot from White to Green Points

(c) Corner Plot from White to Dark-red Points

Figure 4.3: Moving Matrix Interpolation Characterization Performance (FOGRA39)

(a) Round Trip Errors

(b) Corner Plot from White to Green Points

(c) Corner Plot from White to Dark-red Points

Figure 4.4: Iteratively Clustered Interpolation Characterization Performance (FOGRA39)

Data-set	CR			
	mean (∇E)	std (∇E)	max (∇E)	Time(s)
FOGRA39	2.37	3.22	10.69	5
FOGRA41	1.87	2.34	9.12	5
FOGRA42	2.13	2.84	9.27	5
FOGRA43	1.84	2.89	12.05	6
FOGRA44	2.27	4.83	21.63	6

Table 4.2: Continuous Recharacterization Characterization Performance

4.2 Characterization Routines and Drifting Printers

While most high-end production devices have calibration routines to bring the engine back to a center point, this does not keep the engine’s characterization data consistent, especially when considering overlays. Most printer calibration techniques either look only at single separations, or the neutral axis in order to perform linearization [47, 48]. Thus there is little or no control on overlay performance. Neutrals are particularly difficult to control, especially when the black channel is not utilized to generate them, as is often the case when utilizing a Minimum-K or K-Constrained GCR strategy. Even for systems with printer calibration routines it is conceivable that drift can occur between calibration runs. The data in Table 4.3 demonstrates that if the printer’s response drifts during print between printer characterization and calibration, the quality of the reproduction degrades. Moreover Figure 4.5 shows a shift to green/cyan in the neutral axis as a result of a shift in printer dynamics, since this area is highly sensitive to variation in colorant density. Finally, Figures 4.6-4.8 represent the results at t_0 before printer drift and t_9 after the printer drift has completed. The *deltaE* images compare desired CIELAB values to those printed. Interpretation of the image can be made using Table 4.5. These results will be referenced later for comparison purposes in the following section.

Real printer color drift can occur as a result of all pigments shifting at once, or a single colorant behaving oddly. Our drifting printer model moves all colors at the same time. While this drifting printer simulation might not be perfect in mimicking any given printer’s color drift exactly, it clearly injects changes into the printer’s response to CMYK. Thus we can say that if

time	mean (∇E)	std (∇E)	max (∇E)
0	2.15	2.35	9.72
1	1.98	1.38	9.28
2	2.23	1.38	8.95
3	2.67	1.19	8.88
4	3.26	1.30	8.53
5	3.91	1.66	8.45
6	4.62	2.26	9.52
7	5.35	3.03	11.57
8	6.10	4.00	13.19
9	6.85	5.11	14.69

Table 4.3: ICI Performance with Printer Drift (FOGRA39, FOGRA43)

such shifts were to occur, the static characterization routines cannot maintain color consistency as the machine drifts.

4.3 Recharacterization Performance

The most complex part of performance testing the Continuous Recharacterization technique was getting used to the fact that overall performance is dependent on the image contents printed. Unlike past algorithms based entirely on a predefined set of characterization data, extra noises exist in our inputs (the image content) and thus we needed to slightly modify how we measured performance. Certainly we would like to maintain gamut-wide accuracy. However, if data for the dark purple region of the gamut does not exist in the input image (e.g. Spring image), then there is a good chance that we will not be able to properly maintain gamut-wide accuracy while printing such an image. Thus we needed to better break down how we looked at performance. First we looked at the performance of the routines with feedback from images which attempt to cover most of the printer's gamut. Here we clearly saw that with this feedback mechanism, color accuracy was maintained within characterization levels or *better* due to the availability of colors that were previously not available. We then looked at the results from feedback routines based on the Lena image, which does not cover the entire printer gamut. Here we saw that while gamut wide accuracy did not stay

Figure 4.5: Neutral Drift Using ICI Algorithm

within visual differences, the image printed continued to maintain a higher quality than static characterization routines under the same conditions. Finally we added feedback from the full gamut image set. The original images and gamuts for the Art, Lena and Spring images can be found in Figures 4.9-4.11. Notice that for the most part, the union of the Lena and Spring image gamuts covers the vast majority of the FOGRA39 gamut. Thus we showed that when the remaining printer-gamut colors were updated by the sensor system, we were able to cover the full-printer gamut with high accuracy.

As we have explained color characterization accuracy and accurate image reproduction are only two aspects that are critical to a quality image reproduction. Additionally, a quality gamut boundary descriptor and associated gamut mapping algorithm play an equally large role in a pleasing image reproduction. Thus we also monitored the gamut of the printer and output images over time.

4.3.1 Full Printer Gamut Coverage Performance Analysis

We utilized the Continuous Recharacterization technique to print images that cover a large portion of the printer gamut (Art, Ski, Spring, Lena). These images and associated gamut can be found in Figures 4.9, 3.16, 4.11,

(a) $T=0$ δE Image

(b) $T=0$ Printed Image

(c) $T=9$ δE Image

(d) $T=9$ Printed Image

Figure 4.6: Art Image Results Using ICI Algorithm with Printer Drift

(a) $T=0$ δE Image

(b) $T=0$ Printed Image

(c) $T=9$ δE Image

(d) $T=9$ Printed Image

Figure 4.7: Lena Image Results Using ICI Algorithm with Printer Drift

(a) $T=0$ ΔE Image

(b) $T=0$ Printed Image

(c) $T=9$ ΔE Image

(b) $T=0$ Printed Image

Figure 4.8: Spring Image Results Using ICI Algorithm with Printer Drift

(a) Original Art Image

(b) Original Art Image Gamut

Figure 4.9: Art Image and Gamut

a) Original Lena Image

b) Original Lena Image Gamut

Figure 4.10: Lena Image and Gamut

a) Original Spring Image

b) Original Spring Image Gamut

Figure 4.11: Spring Image and Gamut

and 4.10, respectively.

The results after each time block are summarized in Table 4.4. While the global mean δE for the characterization data set performed at high levels, we also measured the performance of the Art, Lena, and Spring Images at t_0 and t_9 image. These can be seen in Figures 4.12-4.14, which can be decoded using the information in Table 4.5. Not only did the performance of CR drastically improve the quality of the global mean, as compared to the results in 4.3, but it also maintained high quality rendering for the image providing the image content as compared with that in Figures 4.6-4.8. Figure 4.15 contains dark red corner plots for various times during the printer drift experiment. As can be seen, the smoothness of the CMYK response continued to be maintained throughout the transient period. Finally, Figures 4.16 and 4.17 visualize the transient printer gamut and the output gamut of the Art Image, respectively. While the size of the printer gamut appeared somewhat constant, the specific gamut boundaries were shifting. It is the subtle shifts in gamut that cause traditional algorithms to fail. Moreover, notice in the Art Image Gamut, that as the printer's gamut shifted, the output Art Image was able to utilize the newly found printer gamut. Clearly we have demonstrated that in the face of gamut movement, we can maintain high quality reproduction and perhaps even improve on the gamut utilization. These shifts in gamut mapping may not be desirable, since the customer desires a consistent image reproduction. More real-world studies would be required in order to determine if utilizing the newly found printer gamut is a feature that is pleasing and desirable. However, previously, without the CR strategy, the gamut boundary descriptor would remain constant, and as such gamut mapping could improperly map in-gamut colors to out of gamut colors.

In conclusion, the CR algorithm is capable of maintaining high color reproduction quality during printer drifts when the customer is printing in large portions of the printer's gamut.

time	mean (∇E)	std (∇E)	max (∇E)	Images Printed
0	1.64	1.89	7.13	Art, Lena, Ski, Spring
1	1.56	1.96	7.41	Art
2	1.65	2.03	7.79	Art
3	1.82	2.16	8.36	Art
4	1.84	2.48	9.44	Art, Spring
5	1.78	2.7	9.06	Art, Ski
6	1.96	2.94	9.22	Art, Spring
7	2.18	3.28	9.88	Art, Ski
8	2.29	3.37	10.6	Art, Spring
9	2.38	3.94	11.10	Art, Ski, Spring, Lena

Table 4.4: CR Performance with Printer Drift (FOGRA39, FOGRA43)

Color	δE
Black	< 1
Blue	< 3
Cyan	< 5
Green	< 7
Yellow	< 10
Red	otherwise

Table 4.5: DeltaE Image Decoding Values

T=0 δE Image

(b) T=0 Printed Image

(c) T=9 δE Image

(d) T=9 Printed Image

Figure 4.12: Art Image Results Using CR Algorithm with Printer Drift

T=0 δE Image

(b) T=0 Printed Image

(c) T=9 δE Image

(d) T=0 Printed Image

Figure 4.13: Lena Image Results Using CR Algorithm with Printer Drift

T=0 δE Image

(b) T=0 Printed Image

(c) T=9 δE Image

(d) T=9 Printed Image

Figure 4.14: Art Image Results Using CR Algorithm with Printer Drift

(a) T=0 Dark Red Corner Plot

(b) T=5 Dark Red Corner Plot

(c) T=9 Dark Red Corner Plot

Figure 4.15: CR Algorithm Dark Red Corner Plot with Printer Drift

(a) T=0 Printer Gamut

(b) T=5 Printer Gamut

(c) T=9 Printer Gamut

Figure 4.16: Printer Gamut with Printer Drift

(a) T=0 Art Gamut

(b) T=5 Art Gamut

(c) T=9 Art Gamut

Figure 4.17: Art Output Image Gamut with Printer Drift

4.3.2 Partial Printer Gamut Coverage Performance Analysis

After completing a full gamut analysis, we again utilized the Continuous Recharacterization method, but instead, we intentionally provided image content that covered only part of the printer's gamut. The Lena image's gamut can be seen in Figure 4.10. When compared to that of the FOGRA39 and FOGRA43 gamuts, as in (a), (c) of Figure 4.16, it is clear that Lena only covers part of the overall printer's gamut. We therefore chose Lena as a test target to print during printer drift in order to demonstrate that while the overall performance of characterization data round-trip-error does not stay within visible delta, the current image content being printed continues to be of high quality.

Figures 4.18 and 4.19 show a few snap-shots of the image reproduction of Lena through time. Through t_9 it is clear that very little error was introduced into the image content. Some of the dark regions show poorer performance, but this can be explained by a shift in gamut, as seen in 4.16, since this experiment was run with the same printer model. On the other hand, Table 4.6 shows round-trip errors over time during printer drift. It is clear that while the full printer gamut accuracy was slightly better than having no recharacterization technique, it was not maintained to below visual differences. However, after taking data for iteration #9, we went on to print additional content, starting with the Spring image, then the Art image, and concluding with the Ski image. The overall round-trip-errors continued to improve until stabilizing around 2.37 deltaE after the Ski image. This data can be seen in Table 4.6, t_9^* . Printing more of the output gamut clearly facilitates higher reproduction performance. When printing only a single target with a small gamut, performance of that image's gamut is maintained, until such time that full printer gamut is printed, allowing the entire system to converge towards t_0 quality. Furthermore, after printing the same targets as in Table 4.4, t_9 performance was almost identical, indicating that no matter where a system may start, the color information available is utilized similarly regardless of the steps and image content sampled to arrive at that moment.

(a) $T=0$ δE Image

(b) $T=5$ δE Image

(c) $T=9$ δE Image

(d) $T=9^*$ δE Image

Figure 4.18: Lena Image δE Feedback Results Using CR Algorithm with Printer Drift

(a) T=0 Printed Image

(b) T=5 Printed Image

(c) T=9 Printed Image

(d) T=9* Printed Image

Figure 4.19: Lena Image *Printed Image* Results Using CR Algorithm with Printer Drift

time	mean (∇E)	std (∇E)	max (∇E)	Images Printed
0	1.82	1.93	7.42	Lena
1	1.73	1.91	7.57	Lena
2	1.97	1.92	8.45	Lena
3	2.39	2.09	9.65	Lena
4	2.91	2.42	10.81	Lena
5	3.45	3.06	12.29	Lena
6	4.05	3.83	13.62	Lena
7	4.64	4.94	15.16	Lena
8	5.22	6.32	16.37	Lena
9	5.58	7.69	17.59	Lena
9*	2.37	4.04	12.80	Spring, Art, Ski, Lena

Table 4.6: Lena Image Feedback Results Using CR Algorithm with Printer Drift

4.3.3 Characterization for Free

One of the claims made regarding the CR system is that printing a test-pattern covering the printer's gamut results in a characterized printer. This data must be in the form of a CMYK test-pattern. We attempted to generate a situation to properly demonstrate this performance improvement. As a result we intentionally drifted the simulated printer without feedback. We then simulated the printing of a set of CMYK patches, and processed them through the image content data path, including the scanner system. For comparison a formal characterization routine was then run. The resulting data is seen in Table 4.7. The data indicates that while a drastic and acceptable performance improvement is made when printing a characterization data set, a formal characterization significantly outperforms it. This is due to the extra pre-processing performed on the image content feedback, which is first merged with the existing color library. If this merging process fails to remove all of the old data, it is likely that the performance of the overall gamut will be impacted. This artifact might imply that we have not fully optimized the feedback system. Alternatively, the sensor system adds noise due to its own numerical and sensing imperfections. As such, it is difficult to say which is the major contributor, perhaps and likely it is both the algorithm and the sensing system contributing a portion of the additional error seen in characterization performance. Thus, while the performance of a real

Description	mean	std	max
Pre-drift	1.91	1.98	7.42
Post-drift	6.54	4.35	16.37
Post-Characterization Print	2.08	2.92	13.82
Format Characterization	1.54	2.10	9.70

Table 4.7: Performance of Printing Full Gamut Patches Versus Formal Characterization

characterization routine may outperform the continuous recharacterization method, continuous recharacterization provides the unique opportunity to never stop printing and offers print line management new work-flows not previously available.

4.3.4 Neutral Drift Experiment

One of the very hard problems in color controls is drifting neutral and memory tones. In Figure 4.5 there are a series of images that represent one such shift, simulated by using the ICI algorithm. A similar experiment using the CR algorithm was run. However, content feedback was utilized during the process, then the neutral image was printed. The result in Figure 4.20, shows that with the CR algorithm, such defects are avoided and the printer's response is properly tracked to ensure neutral and memory tone reproduction, even in sensitive areas of the printer's gamut.

Figure 4.20: Neutral Drift Using CR Algorithm

Chapter 5

Conclusions and Future Work

In this thesis we have claimed, demonstrated, and assessed the benefits of utilizing customer image content for continuous characterization. Our experiments show that using the customer image for continuous recharacterization improves performance considerably and if sufficient gamut coverage exists in the image content printed, the color control system will converge to below noticeable color differences. We have also outlined state of the art Color Control features and printing technologies and have demonstrated how this work fits directly into and extends the state of the art.

By utilizing a simulated environment we have been able to focus on the Characterization and feedback aspects and demonstrate their success. The next logical step is to perform these routines on a real printer. To keep the test practical and simple, it would require the creation of ICC profiles or 3D-LUTs based on characterization data from a t_0 point on a machine. These would then be utilized to analyze the performance of the base characterization and color features implemented as part of this thesis. We could then drift Magenta by directly controlling the engine targets and making Magenta considerably darker. Finally, we would print a simple test pattern, scan it, and perform our feedback loop manually. The updated ICC profiles could then be placed on the machine, in hopes that the quality returns with the updated profile. This could be done by comparing neutral prints from before and after updating the printer's profiles.

Taking advantage of customer image content for feedback represents a major improvement in the page to page and day to day color performance of digital color reproduction devices. The technologies explored in this thesis

advance the state of the art, and while real world tuning and experimentation is still to be completed, the concept is worth exploring further.

Bibliography

- [1] L. K. Mestha and S. A. Dianat, *Control of Color Imaging Systems: Analysis and Design*. CRC Press, 2009.
- [2] E. S. Hamby and E. M. Gross, “A control-oriented survey of xerographic systems: Basic concepts to new frontiers,” *IEEE*, vol. Proceeding of the 2004 American Control Conference, pp. 2615–2629, 2004.
- [3] J. G. Elliot, “Tc-tribo interview.” From a series of interviews both in person and on the phone. John Elliot was a major contributor to the understanding of how toner and carrier and development system function. He led design teams for numerous active and significant products during his 35 years at the Xerox Corporation, Webster, NY., 2009.
- [4] D. C. Craig, “Tough color control problem interview.” Notes from a series of interviews. In his twenty plus years at Xerox, David Craig has been a major contributor in a broad range of products, as an engineer, product manager and as a competency center leader. He has focused on image quality and color controls for the vast majority of his career., July 2010.
- [5] R. Bala, *Digital Color Imaging Handbook (Chapter 5)*. CRC Press, 2003.
- [6] e. a. Alan C. Bovik, *The Essential Guide To Image Processing*. Academic Press, Elsevier Inc., 2009.
- [7] I. C. Consortium, “Image technology colour management; architecture, profile format, and data structure,” 2004.
- [8] S. K. Shevell, *Science of Color*. Optical Society of America, 2003.

- [9] J. Kaiser, "Inkjet printing: Color accuracy," *IEEE*, 2009.
- [10] M. Anderson, "Digital camera calibration for color measurements on prints," *SPIE*, vol. 64930T, pp. 1–9, 2007.
- [11] R. E. Burger, "Device independent color scanning," *SPIE*, vol. 1909, pp. 70–74, 1993.
- [12] A. G. Yifeng Wu, "Printer color calibration using an embedded sensor," *Color Imaging XII: Processing, Hardcopy and Applications*, vol. 64930P, pp. 1–8, 2007.
- [13] B. M. Nengsheng Zhang and W. Lan, "Next generation color management process modeling for digital printing," *IEEE*, vol. International Conference on Industrial Informatics, pp. 932–937, 2006.
- [14] G. Pawle and L. Borg, "Evolution of the icc profile connection space," *SPIE*, vol. 4421, pp. 446–450, 2002.
- [15] D. Littlewood and G. Subbarayan, "Maintaining an accurate printer characterization," *Twelfth Color Imaging Conference: Color Science and Engineering Systems, Technologies, and Applications*, vol. 12, pp. 203–210, 2004.
- [16] N. Y. H. Kotera, T. Morimoto and R. Saito, "Adaptive color correction based on objects color classification," *SPIE*, vol. 3409, pp. 278–285, 1998.
- [17] R. Balasubramanian, "Reducing the cost of lookup table based color transformations," *Seventh Color Imaging Conference: Color Science, Systems and Applications*, vol. 7, pp. 65–68, 1999.
- [18] M. A. Kriss, "Modeling an electro-photographic printer: Part i, monochrome systems," *Color Imaging XII: Processing, Hardcopy and Applications*, vol. 64930N, pp. 1–14, 2007.
- [19] M. A. Kriss, "Modeling an electro-photographic printer: Part ii, color systems," *Color Imaging XII: Processing, Hardcopy and Applications*, vol. 64930O, pp. 1–10, 2007.

- [20] H. Zeng, “3-d color separation maximizing the printer gamut,” *SPIE*, vol. 5008, pp. 260–267, 2003.
- [21] H. Zeng, “Gray component replacement by direct colorimetric mapping,” *SPIE*, vol. 3963, pp. 317–322, 2000.
- [22] K. M. Hitoshi Ogatsu and S. Kita, “A flexible gcr based on cielab,” *SPIE*, vol. 2414, pp. 123–133, 1995.
- [23] R. Balasubramanian and E. Dalal, “A method for quantifying the color gamut of an output device,” *Color Imaging: Device-Independent Color, Color Hard Copy, and Graphic Arts II*, vol. 3018, pp. 110–116, 1997.
- [24] M. Mahy, “Gamut calculation of color reproduction devices,” *The Fourth Color Imaging Conference: Color Science, Systems, and Applications*, vol. 4, pp. 145–150, 1996.
- [25] G. J. Braun and M. D. Fairchild, “Techniques for gamut surface definition and visualization,” *Fifth Color Imaging Conference: Color Science, Systems and Applications*, vol. 5, pp. 147–152, 1997.
- [26] M. R. L. Ján Morovi, “Calculating medium and image gamut boundaries for gamut mapping,” *Color Research & Application*, vol. 25, pp. 394 – 401, 2000.
- [27] P.-L. S. Jan Morovic and P. Morivic, “The gamuts of input and output colour imaging media,” *Color Imaging: Device-Independent Color, Color Hardcopy, and Graphic Arts VI*, vol. 4300, pp. 114–125, 2001.
- [28] D. Poole, *Linear Algebra: A Modern Introduction*. Thomson, 2006.
- [29] J. Stewart, *Calculus Early Transcendentals*. Thomson, 2003.
- [30] P. J. Olver and C. Shakiban, *Applied Linear Algebra*. Pearson/Prentice Hall, 2006.
- [31] W. T. V. William H. Press, Saul A. Teukolsky and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2002.

- [32] D. L. Lee and G. Marcu, "Performance of media-specific and device-specific output profiles," *SPIE*, vol. 2963, pp. 119–130, 2000.
- [33] E. W. R.S. Gentile and J. P. Allebach, "A comparison of techniques for color gamut mismatch compensation," *Journal of Imaging Technology*, vol. 16, pp. 176–181, 1990.
- [34] E. D. Montag and M. D. Fairchild, "Gamut mapping: Evaluation of chroma clipping techniques for three destination gamuts," *Sixth Color Imaging Conference: Color Science, Systems and Applications*, vol. 6, pp. 57–61, 1998.
- [35] G. Marcu, "Gamut mapping: An overview of the problem," *PICS 1999: Image Processing, Image Quality, Image Capture, Systems Conference*, vol. 2, pp. 192–199, 1999.
- [36] R. B. Karen M. Braun and R. Eschbach, "Development and evaluation of six gamut-mapping algorithms for pictorial images," *Seventh Color Imaging Conference: Color Science, Systems and Applications*, vol. 7, pp. 144–148, 1999.
- [37] G. Marcu, "Some gamut mapping issues and solution," *Color Imaging: Device-Independent Color, Color Hardcopy, and Graphic Arts VI*, vol. 4300, pp. 131–138, 2001.
- [38] H. Zeng, "Spring-primary mapping: a fast color mapping method for primary adjustment and gamut mapping," *Color Imaging XI: Processing, Hardcopy, and Application*, vol. 6058, pp. 1–12, 2005.
- [39] S. J. Cao Congjun, "Study on color space conversion between cmyk and cielab based on generalized regression neural networks," *IEEE*, vol. International Conference on Computer Science and Software Engineering, pp. 275–277, 2008.
- [40] I. W. Sandberg, "Global implicit function theorems," *IEEE*, vol. CAS-28, pp. 145–149, 1981.
- [41] R. Balasubramanian and M. S. Maltz, "Refinement of printer transformations using weight regression," *SPIE*, vol. 2658, pp. 334–340, 1996.

- [42] S. A. D. Daniel E. Viassolo and L. Mestha, "Practical algorithm for the inversion of an experimental input-output color map for color correction," *Journal of Optical Engineering*, vol. 42(3), pp. 625–631, 2003.
- [43] E. K. G. Maya R. Gupta and E. Chin, "Adaptive local linear regression with application to printer color management," *IEEE Transactions on Image Processing*, vol. 17 NO.6, pp. 936–945, 2008.
- [44] S. Wang, "Color image segmentation based on color similarity," *IEEE*, vol. CiSE 2009. International Conference on Computational Intelligence and Software Engineering, 2009.
- [45] A.-R. Gomez, Sanches-Yanez and Correa-Tome, "Natural image segmentation using the cielab space," *IEEE*, vol. CONIELECOMP 2009. International Conference on Electrical, Communications, and Computers., pp. 107–110, 2009.
- [46] M. Elliot, "Scheduling system for placing test patches in a printing apparatus," 2009.
- [47] G. S. Raja Bala, Vishal Monga and J.-P. V. de Capelle, "Two-dimensional transforms for device color calibration," *SPIE*, vol. 5293, pp. 250–261, 2004.
- [48] Y. Wu, "Combine 1d and 3d color calibration methods for ensuring consistent color reproduction," *SPIE*, vol. 5293, pp. 242–249, 2004.
- [49] R. Bala, "Challenges in color reproduction: Towards higher dimensions," *SPIE*, vol. 5667, pp. 162–169, 2005.
- [50] R. Chung, "Process control and color management implementation," *SPIE*, vol. 5293, pp. 269–276, 2004.
- [51] P. J. Green, "A reference printer and color management architecture," *Color Imaging XII: Processing, Hardcopy and Applications*, vol. 64930R, pp. 1–6, 2007.
- [52] H. Zeng and J. Humet, "Inter-printer color calibration using constrained printer gamut," *Color Imaging X: Processing, Hardcopy and Applications*, vol. SPIE-IS&T/Vol. 5667, pp. 393–400, 2005.