Spectral Modeling of a Six-Color Inkjet Printer

Lawrence A. Taplin
B.S. University of Delaware
(1996)

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science in Color Science
in the Chester F. Carlson Center of Imaging Science
of the College of Science
Rochester Institute of Technology

December 2001

_____

Signature of the Author

_____

Accepted by Dr. Roy S. Berns,
Coordinator, M.S. Degree Program

CENTER FOR IMAGING SCIENCE
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

M.S DEGREE THESIS

The M.S. Degree Thesis of Lawrence A. Taplin
has been examined and approved by two member of the
Color science faculty as satisfactory of the thesis
requirement for the Master of Science degree

Dr. Roy S. Berns, Thesis Advisor

Dr. Jonathan S. Arney

CENTER FOR IMAGING SCIENCE
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

THESIS REPRODUCTION PERMISSION STATEMENT

Title of thesis: **Spectral Modeling of a Six-Color Inkjet Printer**

I, Lawrence A. Taplin, hereby **grant permission** to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date: _____ Signature of Author: _____

Spectral Modeling of a Six-Color Inkjet Printer

Lawrence A. Taplin
B.S. University of Delaware
(1996)

Acknowledgement

My sincerest thanks go out to my advisor Roy, my family and friends.

Spectral Modeling of a Six-Color Inkjet Printer

Lawrence A. Taplin

B.S. University of Delaware

(1996)

A thesis submitted in partial fulfillment of the requirements

for the degree of Master of Science in Color Science

in the Chester F. Carlson Center of Imaging Science

of the College of Science

Rochester Institute of Technology

ABSTRACT

After customizing an Epson Stylus Photo 1200 by adding a continuous-feed ink system and a cyan,magenta,yellow,black,orange and green ink set, a series of research tasks were carried out to build a full spectral model of the printers output. First, various forward printer models were tested using the fifteen two color combinations of the printer. Yule-Nielsen-spectral-Neugebauer (YNSN) was selected as the forward model and its accuracy tested throughout the colorant space. It was found to be highly accurate, performing as well as a more complex local, cellular version. Next, the performance of nonlinear optimization-routine algorithms were evaluated for their ability to efficiently invert the YNSN model. A quasi-Newton based algorithm designed by Davidon, Fletcher and Powell (DFP) was found to give the best performance when combined with starting values produced from the non-negative least squares fit of single-constant Kubelka-Munk. The accuracy of the inverse model was tested and different optimization objective functions were evaluated. A multistage objective function based on minimizing spectral RMS error and then colorimetric error was found to give highly accurate matches with low metameric potential. Finally, the relationship between the number of printing inks and the ability to eliminate metamerism was explored.

TABLE OF CONTENTS

1. INTRODUCTION

Traditional imaging systems take advantage of the fact that the human visual system is trichromatic. These systems record and reproduce a wide range of visual stimuli using just three channels of information. In most applications this works quite well. However, three-color reproductions systems are insufficient if printed reproductions from these systems are viewed under a range of lighting conditions. For several years research efforts within the Munsell Color Science Laboratory (MCSL) have been focused on the spectral reproduction of color.[1-60] If the spectral characteristics of the original are recorded and reproduced across all wavelengths of the visual spectrum, differences in the appearance of the reproduction will be eliminated even when the observer or lighting are changed. Practically, exact spectral reproduction through conventional inkjet printing cannot be obtained. However, the colorimetric redundancy within a six-color inkjet system can be exploited to minimize metamerism in the final print while maintaining a high level of colorimetric accuracy.

The current research effort described in this thesis is the direct continuation of the work begun by Di-Yuan Tzeng in creating a complete spectral color-output system. Tzeng presented his research in the form of a doctoral dissertation and a series of papers at the annual Color Imaging Conference.[23,27,37,44]

The current research initiative included several stages. First, a six-color inkjet printing system, around which the research would be conducted, was assembled. The next stage involved selecting an appropriate mathematical model to transform between digital counts and printed spectra. Next, the effectiveness of this model in predicting the six-ink output from the printing system was tested. Research into methods for inverting

the forward printer model so that reflectance spectra can be transformed into the digital counts best reproducing them were tested next. The accuracy of the inverse model was tested and optimization objective functions leading to the best spectral matches were explored. Finally, the relationship between the number of inks used in the printing system and the degree of metamerism in the matches was explored.

The completed printing system fits into the broader spectral color reproduction research being conducted at MCSL as the output stage of a full end-to-end spectral capture, archiving and output system (Figure 1-1).

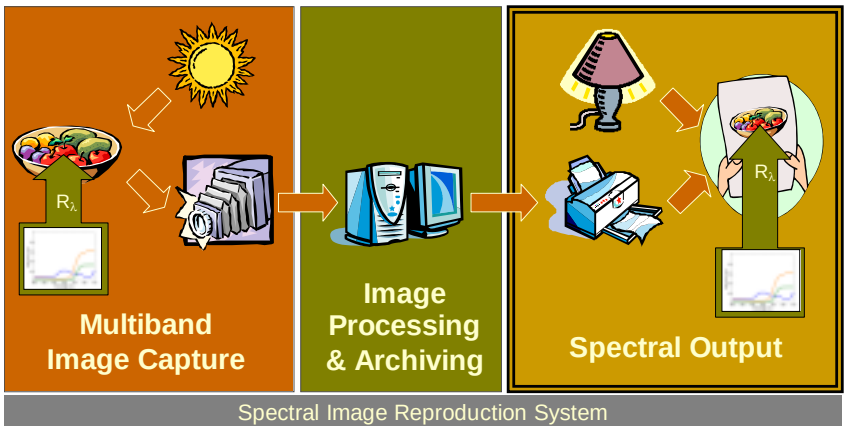


Figure 1-1 – Overview of an end-to-end spectral image reproduction system. The output stage that is the focus of this research is on the right.

2. MATERIALS

The following section details the hardware and software components that were assembled into the completed printing system.

Hardware

Epson Stylus Photo 1200

The Epson Stylus Photo 1200, a consumer class desktop inkjet printer was selected based on its high resolution, low cost and six-color capacity. The printer, shown below in Figure 2-1, was modified by changing the ink set and adding a continuous-feed ink-supply system. The printer has an advertised resolution of 1440x720dpi; however, it was only used at a resolution of 720x720dpi.

Figure 2-1 - Modified Epson Stylus Photo 1200 printer with CYMKOG inkset and continuous-feed system.

MIS Associates Continuous Feed System

A continuous feed ink supply system was installed into the printer so that a large number of prints could be made before the ink needed to be replenished. This was important in limiting the number of times the printer required characterization.

Figure 2-2 - MIS Associates continuous-feed ink supply system.

The feed system is shown in Figure 2-2. Ink drawn out through the bottom of the cartridges by the vacuum from the print head causes additional ink to be drawn in through the tubes at the top that are connected to the vent holes. The free ends of the tubes rest at the bottom of the four-ounce ink bottles.

Pigmented ink set

At the time the printing system was assembled, there were several CMYK ink sets to choose from. Archival-pigmented CMYK inks from MIS Associates were selected but additional ink colors were not yet available. Green and Orange ink from a Roland Hi-Fi jet printer was used to fill the fifth and six positions in the printer. Since the printer had already been used with regular Epson inks, it was necessary to flush out the print head before installing the new inks. This was accomplished by printing many pages of a dense test target using special cleaning cartridges also purchased from MIS Associates. The

new inks were installed in the system using the instructions included with the ink-feed system. Patches of each ink were printed and measured to produce the plot of reflectance spectra shown in Figure 2-3. By inspection, it is clear that the green and orange inks cannot be reproduced spectrally by combinations of cyan, magenta yellow and black.

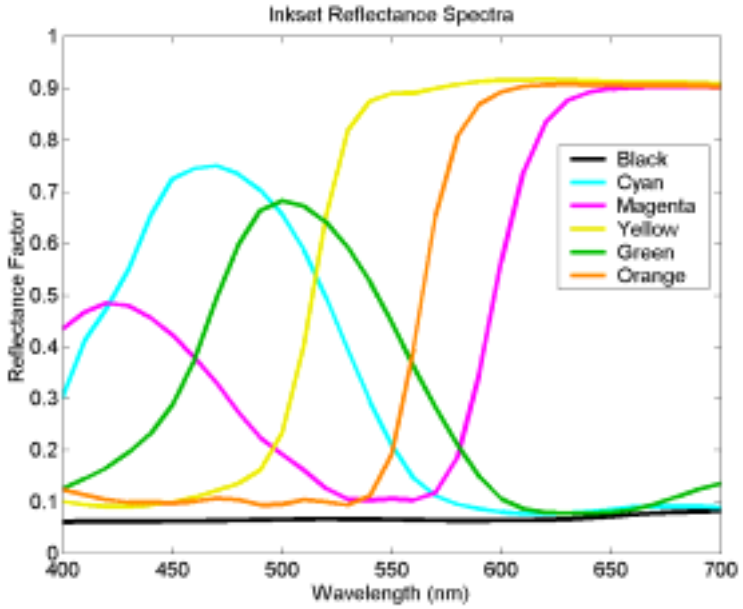


Figure 2-3 - Reflectance spectra for CMYKOG inkset.

Paper

After visually assessing output on various high quality inkjet papers, Lumijet Classic Velour was selected for use in the printer characterization and modeling reseach. The paper, manufactured by Luminos, is a heavyweight (310gsm) matte paper with a velvety finish. With only minimal bleeding, the paper was able to absorb the ink from patches printed at maximum area coverage for all six channels.

Figure 2-4 - Lumijet Preservation Series Classic Velour paper.

Spectrophotometer

A GretagMacbeth SpectroScan spectrophotometer was used to make all the spectral measurements in this research. The SpectroScan is a 45°/0° spectrophotometer with a 4mm aperture mounted on a motorized X/Y positioning table. The device was controlled using a software package called SpectraChart that helped automate the measurement of the large number of samples from each printed target. The manufacturer specifications for the instrument state an inter-instrument agreement of $0.3\Delta E^*_{ab}$ (D50,2°) based on 12 BCRA tiles with a maximum of $0.8\ \Delta E^*_{ab}$. Short-term repeatability is stated as 0.03 ΔE^*_{ab} based on 10 measurements of white spaced out ten seconds apart. Further testing of the instrument within MCSL has confirmed a high level of precision and accuracy. As shown in Figure 2-5, printed samples were measured against the dark background of the SpectroScan's top surface.

Figure 2-5 - GretagMacbeth SpectroScan spectrophotometer.

Software

Mathworks MATLAB 5.3

Initial testing and development of the various models and algorithms was conducted using MATLAB, an environment that combines mathematical computing and visualization tools with a high level programming language. The final code makes use of the image processing and optimization toolboxes. The project source code as well as descriptions for each module are included at the end of this thesis as Appendix B.

Metrowerks CodeWarrior 6

One drawback of evaluating the model in MATLAB was the slow speed of execution. To improve performance, much of the research was recoded in C using Metrowerks CodeWarrior 6 cross platform (PC/Mac) compiler. C source code for the project is included as Appendix C.

Numerical Recipes in C

Many of the algorithms needed for this research were already compiled together into one source, the book and CD, "Numerical Recipes in C".[61] Redistribution of the source code for the algorithms is strictly controlled by a license agreement and therefore the code is not included in an appendix. However, the code was used nearly unmodified and the routines need to interface to it are included in Appendix C. The only modifications made were to change the exit conditions of some routines to better handle error conditions.

Solaris based printer driver

A custom printer driver was needed to create binary files that could be copied to the printer. An Internet search uncovered a driver written for the Linux operating system by a programmer in Germany. The author, Jean-Jacques Sarton, was contacted and the driver was modified so that dot level control over the printer could be obtained.[62] Specifically, the driver takes as input an image file with one byte per pixel. Within each byte the first six bits indicate which of the inks should be printed. Because of licensing we were only provided with a binary version of the driver that runs on the Solaris platform. The driver can be invoked from other networked computers using remote execution and the results retrieved using remote file copy commands or FTP. These steps were combined together within a MATLAB script that formed the front end to the printing system.

Halftone Algorithm

Because the printer driver provided dot level control over the printer, the process of halftoning the images to obtain multilevel output was done on the host computer. For all of this research a Floyd-Steinberg error diffusion halftoning algorithm was used.[63] A SGI

routine, *ditherstiff*, part of SGI's Impressario printing and scanning server software provided a convenient implementation. The utility takes as input single channel eight bit TIFFs and outputs bi-level halftoned TIFFs. Again the needed commands were handled with a MATLAB script (see Appendix B).

System Flow Chart

The printing system requires two pieces of input, a spectral image and a dataset of measured spectra from a characterization target in order to produce spectral separations and printed output. The spectral image can take several forms. For most of this research it was created synthetically from rectangular patches of desired spectra. Later, images of complex scene images were processed. These spectral images were generated through multi-channel camera capture and spectral estimation. The spectral measurements of the characterization target are used to build the forward printer model. The model converts from printer digital counts to predicted reflectance spectra. However, since the goal of the system is to go the other direction, from reflectance spectra to digital counts, the model is inverted through nonlinear optimization. The nonlinear optimizer (in MATLAB or C) makes multiple calls to the forward model while varying the six ink levels in order to determine the levels that will minimize a difference metric (objective function) between the predicted spectra from the forward model and the original spectra from the image. The final ink levels are used to create a six-channel separation. Six multilevel images are independently halftoned using the Floyd-Steinberg algorithm and one bit per channel separations are created. These are run through the printer driver to create a binary file

compatible with the printer. Finally, the file is copied to the printer to produce the final hardcopy output. The system flow chart is depicted below as Figure 2-6.

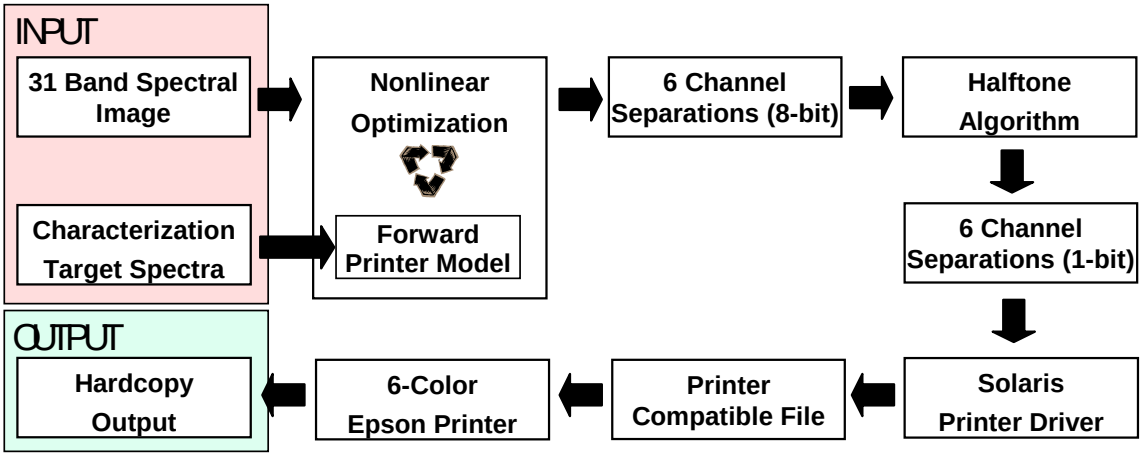


Figure 2-6 - Printing system flowchart.

3. TWO-COLOR MODEL EVALUATION

The goal of this section of research was to identify a two-color mixture model for the printing system that would be a good candidate for extension to the full six-color system.

Halftone Models

Since the printer uses discrete dots placed on the page, it makes sense to evaluate models targeted historically at this type of halftone printing system. The following is a description of several of these models. For a more detailed treatment refer to the review article by Wyble and Berns.[38]

Neugebauer Model

The Neugebauer model is the multi-ink generalization of the Murray-Davies model that predicts the reflectance of multi-ink mixtures in halftone printing. Neugebauer built up his model for the reflectance of a color print from a linear combination of the reflectance spectra of each colorant at full coverage and all possible full coverage overprints.[64,65] The set of full coverage combinations of the inks are called Neugebauer primaries. The equation for reflectance predicted by the Neugebauer model is:

$$\hat{R}_\lambda = \sum_i p_i R_{\lambda,i} \tag{3.1}$$

where, $\hat{R}_\lambda$ is the predicted reflectance vector of the mixture, p_i is a scalar weighting representing the probability that a point on the page of the color mixture is covered by the ith Neugebauer primary and $R_{\lambda,i}$ is the reflectance vector of that primary. If placement

of the ink on the page is assumed to be random then the probability scalars can be determined by the Demichel equations, shown below for a two ink system:[66]

$$\begin{aligned} p_1 &= (1-a_{ink1})(1-a_{ink2}) \\ p_2 &= a_{ink1}(1-a_{ink2}) \\ p_3 &= (1-a_{ink1})a_{ink2} \\ p_4 &= a_{ink1}a_{ink2} \end{aligned} \tag{3.2}$$

where, the "*p*"s are the probability of a point on the paper of the mixture being covered by the subscripted Neugebauer primary and the "*a"s* are the area covered by the subscripted ink. If the area coverages are express as scalars from zero to unity then the probabilities for the Neugebauer primaries also range from zero to unity. Under the further assumption that the model accurately predicts mixture reflectance, the area coverages that minimize spectral error in the predictions of model can be determined empirically for particular signals sent to the printer (digital counts). Using a regression-based approach as shown in equation (3.3) the area effectively covered by a single-ink printed on the paper can be determined.

$$\mathbf{a_{eff}} = \mathbf{R_{meas,adj}}\mathbf{R}^{\mathbf{T}}_{\mathbf{max,adj}}(\mathbf{R_{max,adj}}\mathbf{R}^{\mathbf{T}}_{\mathbf{max,adj}})^{\mathbf{-1}} \tag{3.3}$$

where, for *n* single ink samples a_{eff} is a n-dimensional of vector the estimates of the area effectively coveraged on the paper, $\mathbf{R_{meas,adj}}$ is a (31×n) matrix of the measured spectral reflectance of the printed samples minus the spectral reflectance of the paper and $\mathbf{R}^{\mathbf{T}}_{\mathbf{max,adj}}$ is the transpose of the (31×1) matrix of the spectral reflectance of the maximum possible coverage sample minus the reflectance of the paper. As shown by Yule, the Neugebauer model can be interpreted graphically as shown in the Figure 3-1.[67] The relationship between the area coverage of each ink and the area coverage of the Neugebauer primaries is demonstrated geometrically.

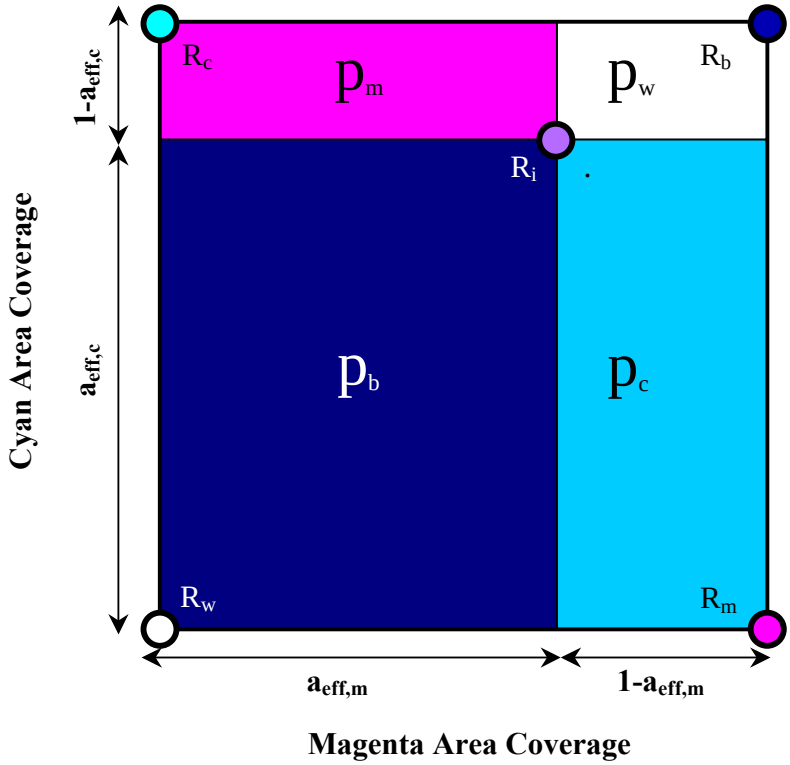


Figure 3-1 - Graphical interpretation of the Neugebauer model.

Theoretical and Effective Area Coverage

As shown in Equation (3.3), effective area coverage can be estimated with least squares regression analysis. This is particularly useful in building lookup tables (LUTs) to speed the conversion of printer digital counts to effective area coverage. The LUTs are necessary because of the nonlinear relationship between the two caused by physical spreading of the ink, optical spreading within the paper and by artifacts of the halftone implementation. This non-linearity is illustrated in Figure 3-2, made by calculating the

effective area coverages of a single ink ramp (magenta); it graphically depicts the contents of the LUT.

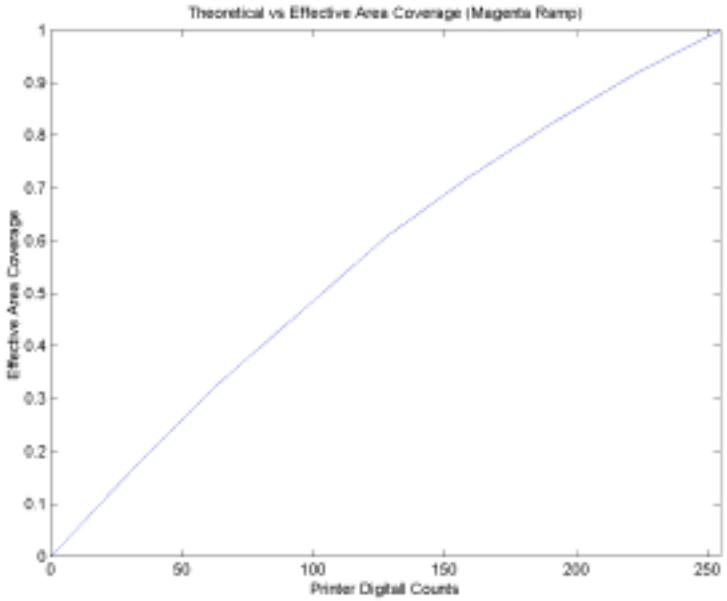


Figure 3-2 – Non-linearity in relationship between printer digital counts and effective area coverage.

Yule-Nielsen Spectral Neugebauer Model

To better predict reflectance of mixtures caused by the complex physical and optical interaction between the ink and paper, Yule and Nielsen proposed an empirically-fit exponent to transform the reflectance data into a space where color mixing could be performed more accurately as a linear sum of each primary's spectral reflectance raised to an exponent, n. The spectral form of the Yule-Nielsen spectral Neugebauer (YNSN) model is shown in the equation below.[68]

$$\hat{R} = \left(\sum_i p_i R_{\lambda,i,\max}^{1/n} \right)^n \tag{3.4}$$

Again the probabilities, *p*, are calculated as shown in equation (3.2) Also, when building the lookup table from digital counts to effective area coverage, the reflectance spectra must all be raised to the 1/n power.

Determination of Yule-Nielsen n-value

An optimized *n*-value was found iteratively by testing values incrementally and noting the effect on the spectral and colorimetric fit of the forward YNSN model. Figure 3-3 shows the relationship between Yule-Nielsen *n*-value, spectral RMS error and ΔE_{00}. A *n*-value of six was selected to bring the maximum values to an acceptable level. Also note that modifying *n* changes the estimated effective area coverage and is therefore not solely accounting for optical dot gain within the paper. Traditionally, an infinite *n*-value would imply a Beer-Lambert continuous tone system; however, such a system's poor predictions of the ink overprints, to be described, indicate this is not the case.

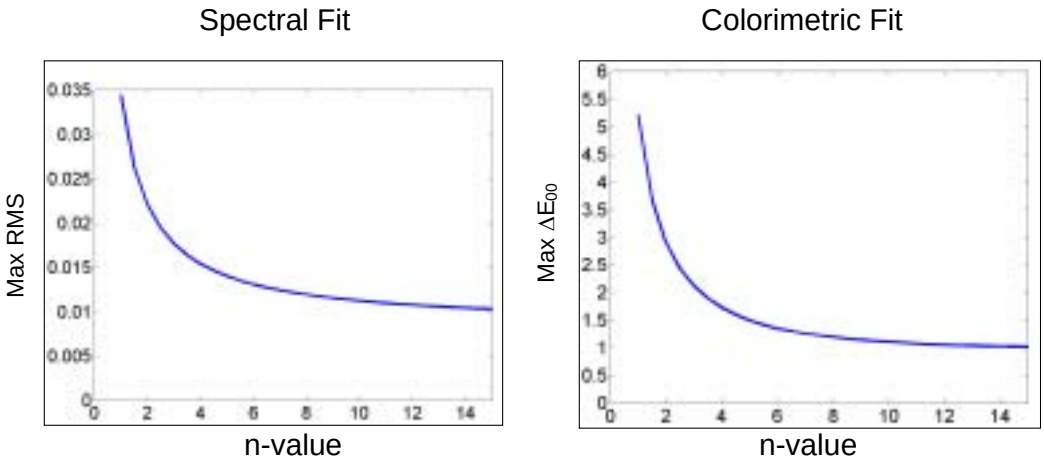


Figure 3-3 - Determination of n-value through incremental adjustment. Maximal spectral and colorimetric error for all six ink colors in comparing the single ink ramps with their forward model predictions using the indicated n-values.

Cellular Neugebauer Models

The two models described above rely on the ability to use a single set of primaries across the ink-colorant space. Heuberger, et al.[69] and later Rolleston and Balasubramanian[70] have explored a localized model where the colorant space is divided into cells and the Neugebauer model is used locally within them. This system is shown graphically in Figure 3-4. Since the area coverages for the primaries are typically expressed in the global colorant coverage coordinates, they must be normalized to the local ones using equation (3.5).

$$a'_{eff} = \frac{a_{eff} - a_{eff,lower}}{a_{eff,upper} - a_{eff,lower}} \tag{3.5}$$

where, a'_{eff}, the normalized effective area coverage is based on the upper and lower bounding area coverages of the cell. This is shown graphically in Figure 3-4.

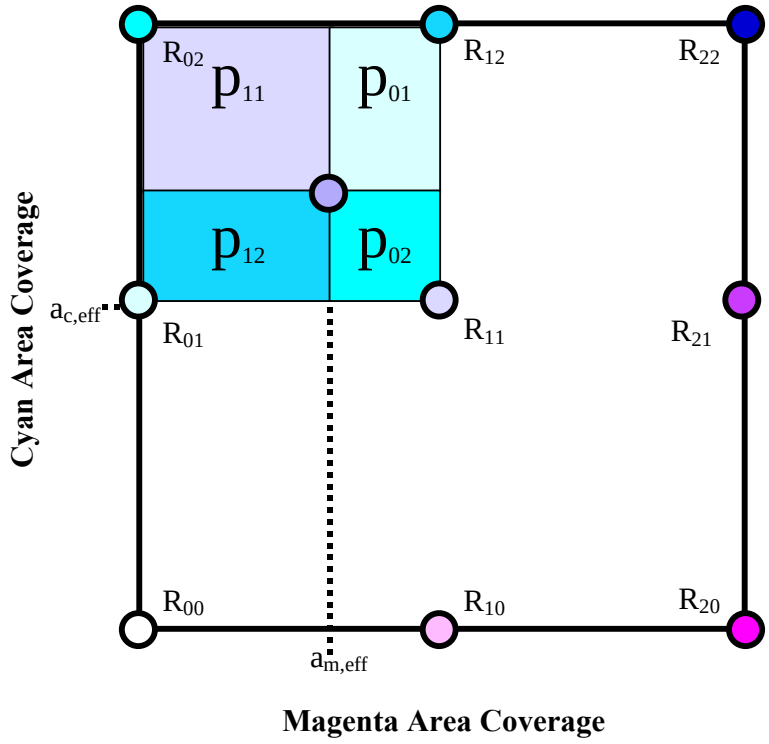


Figure 3-4 - Graphical interpretation of the cellular Neugebauer model.

Continuous Tone Models

Because the individual dots produced by the printing system are very small and the ability to invert the printer model is important, it was hypothesized that a continuous tone models might be useable, even though such a model does not provide an accurate physical representation of the system.

Single-Constant Kubelka-Munk

Kubelka and Munk developed a series of equations that can be used to predict reflectance in many types of colorant systems. Because the printing system employs a water color paper where the ink can be though of as dying the diffuse top layer, the single constant form of Kubelka-Munk was selected over the transparent form. The model relies on a

fixed ratio between the colorant mixture's scattering and absorption properties. In order to convert from reflectance to this K/S space, Equation (3.6) is used:[71]

$$(\mathrm{K/S})_\lambda = \frac{(1-R_\lambda)^2}{2R_\lambda} \tag{3.6}$$

where R_λ is the spectral reflectance of the mixture. To use this continuous tone model, the unit $(\mathrm{k/s})_\lambda$ value of each ink is first calculated by subtracting the $(\mathrm{K/S})_\lambda$ value of the paper from the $(\mathrm{K/S})_\lambda$ value of the ink at full coverage.

$$(\mathrm{k/s})_{\lambda,\mathrm{i}} = (\mathrm{K/S})_{\lambda,\mathrm{i,max}} - (\mathrm{K/S})_{\lambda,\mathrm{paper}} \tag{3.7}$$

The $(\mathrm{K/S})_\lambda$ value of a mixture of inks is calculated as a concentration-weighted sum of the unit $(\mathrm{k/s})_\lambda$ values of the inks and the $(\mathrm{K/S})_\lambda$ value of the paper, as shown in Equation (3.8).

$$(K/S)_{\lambda,mix} = (K/S)_{\lambda,paper} + \sum_i c_i (k/s)_{\lambda,i} \tag{3.8}$$

where, c_i is a concentration scalar for each ink. To convert back from $(\mathrm{K/S})_\lambda$ to reflectance space, Equation (3.6) is inverted as shown in Equation (3.9).

$$\hat{\mathrm{R}}_\lambda = 1 + (\mathrm{K/S})_{\lambda,\mathrm{mix}} - \sqrt{(\mathrm{K/S})^2_{\lambda,\mathrm{mix}} + 2(\mathrm{K/S})_{\lambda,\mathrm{mix}}} \tag{3.9}$$

Cellular Kubelka-Munk

Just as a localized version of the Neugebauer model can be constructed, so to can one for the continuous tone model. Again Equation (3.6) is used to convert all the reflectance data into the colorant mixing space but the effective area coverages are rescaled using the method used in cellular Neugebauer as shown in Equation (3.5) with the area coverages replaced with concentration scalars.

Experimental

Using the six ink printing system, the fifteen two color cases were tested with each of the printer models described above. Nine-by-nine grids for each color combination were printed using digital counts selected in approximately equal-area-coverage steps using the inverse Murray-Davies model of equation (3.3). These digital counts are shown in Table 3-I.

Table 3-I - Digital Counts Used to Print 15 9x9 Grids

Cyan	Magenta	Yellow	Black	Orange	Green
255	255	255	255	255	255
163	179	165	141	164	168
115	130	114	97	111	119
85	97	87	69	83	88
60	70	60	49	59	62
40	47	40	33	40	42
25	30	26	20	26	26
12	15	14	10	13	13
0	0	0	0	0	0

The printed patches of the grids were measured using a GretagMacbeth Spectrolino spectrophotometer. A sample of one of the four printed target pages is shown in Figure 3-5.

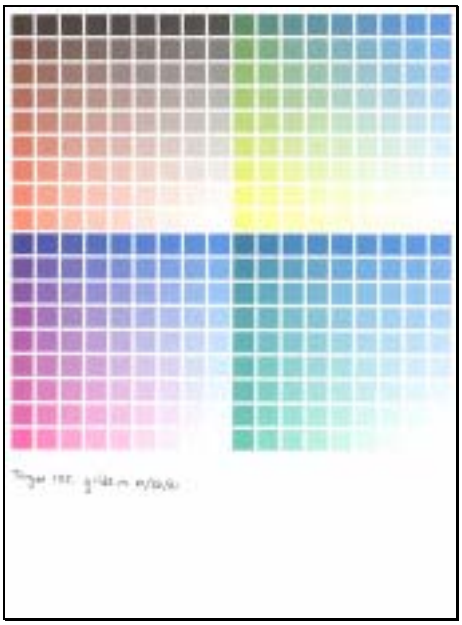


Figure 3-5 - Four of the fifteen two color overprint grid targets.

Since an objective function need not be defined to complete the *forward* models, low colorimetric error is a necessary requirement if the model will be later inverted with hopes of colorimetric accuracy. Root-mean-square (RMS) spectral error for the sample predictions was calculated as a secondary goodness metric for the model performance.

Results and Discussion

Overall performance for each model was established by pooling the eighty-one samples from each of the fifteen two-color overprint cases and computing colorimetric differences between the measured and predicted reflectance spectra. The illuminant D65 and CIE 1931 2-degree standard observer color-matching functions were used in this stage of the evaluation. Each of the Neugebauer type algorithms was tested with two transfer functions to convert between printer digital counts and effective area coverage (Figure

3-2). First the theoretical coverage suggested by dividing the digital count by the maximum digital count was used. Next the area coverages for the single ink ramps extracted from the edges of the grid were used to build lookup tables populated with regression fit area coverages using equation (3.3). The statistical summary for the pooled data is shown in Table 3-II.

Table 3-II - Overall statistical summary of algorithm predictions for all fifteen two color combinations. AC_{theo}=DC/255 and AC_{eff} represents area coverages run through the spline based lookup table. Errors shown are between the printed measured spectra and predictions from the forward printer model.

Forward Printer Model	Mean ΔE^*_{94}	StdDev ΔE^*_{94}	Max ΔE^*_{94}	Mean ΔE^*_{00}	StdDev ΔE^*_{00}	Max ΔE^*_{00}	RMS
Neugebauer (AC_{theo})	12.12	6.07	25.89	10.16	5.06	24.79	0.1702
YN-Neugebauer (n=6) (AC_{theo})	4.53	2.35	11.47	3.98	2.06	10.41	0.0640
Neugebauer (AC_{eff})	2.58	1.73	7.79	2.64	1.78	9.91	0.0285
Neugebauer (n=6) (AC_{eff})	1.32	0.89	4.96	1.31	0.90	4.88	0.0150
Cellular Neugebauer (AC_{theo})	3.39	2.88	14.83	3.12	2.72	14.13	0.0467
Cellular YN-Neugebauer (n=6) (AC_{theo})	1.40	1.38	7.08	1.31	1.31	7.07	0.0195
Cellular Neugebauer (AC_{eff})	0.84	0.72	3.18	0.83	0.73	4.59	0.0104
Cellular YN-Neugebauer (n=6) (AC_{eff})	0.55	0.48	3.04	0.53	0.47	3.10	0.0071
Continuous Tone	1.71	1.79	12.72	1.77	2.02	14.32	0.0152
Continuous Tone Cellular	0.75	0.97	9.53	0.76	1.09	10.53	0.0073

Neugebauer Model

The Neugebauer model provided poor colorimetric accuracy in all of the two color cases. For this discussion the green and orange grid will be shown in detail. Both the model based on theoretical area coverage and the one that used area coverage fit with the inverse Murray-Davies equations showed systematic error shifts. In the model based on the theoretical area coverage most of this shift can be attributed to the inaccurate prediction of the effective area coverages. This assertion is supported by the direction of the shift towards the regions of greater coverage when comparing predicted with

measured spectra as shown in the colorimetric error plots of Figure 3-8 and spectra of Figure 3-6. The improvement when the look-up-table is used for the area coverage levels is dramatic. Figure 3-9 shows the systematic error switching direction to reveal the poor modeling of the physical and optical interactions between the ink and paper inherit in the simple Neugebauer model. Plotting the spectra (Figure 3-7) shows that using effective area coverage corrects mean reflectance level error but that the spectral shape is still poorly predicted. The colorimetric summary reports in Figure 3-9 also displays a metameric-index (MI) which quantifies the spectral error of the predictions in colorimetric terms. First the measured and predicted spectra are paramerically corrected to match under illuminant D65.[72] Next the CIEDE2000 color difference is calculated under Illuminant A. An error larger than one is significant.

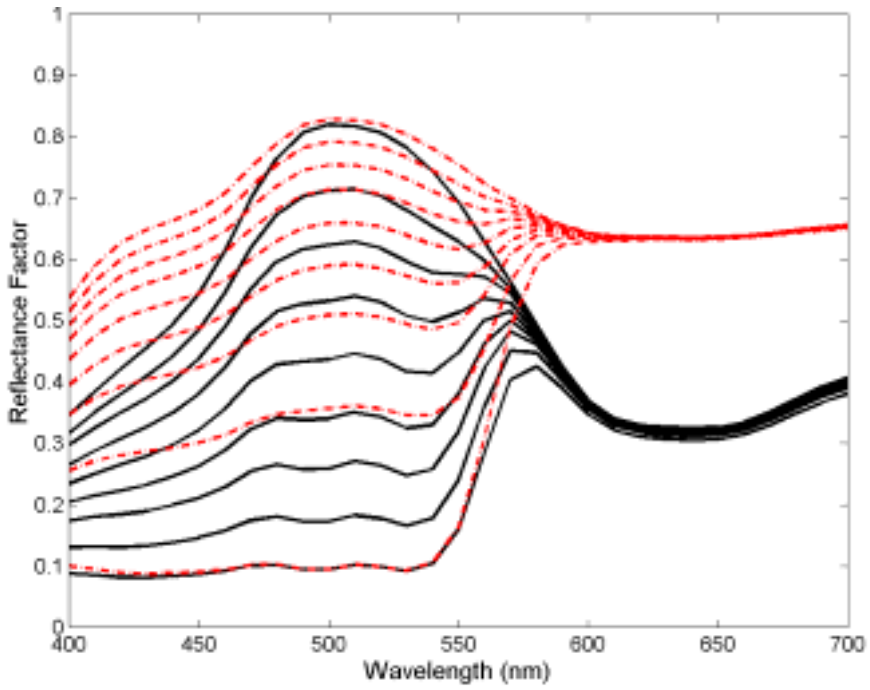


Figure 3-6 – Measured (solid black lines) and predicted (dashed red lines) sample spectra for Neugebauer model with theoretical area coverages. Green DC is fixed at 88 and Orange is varied from 0 to 255.

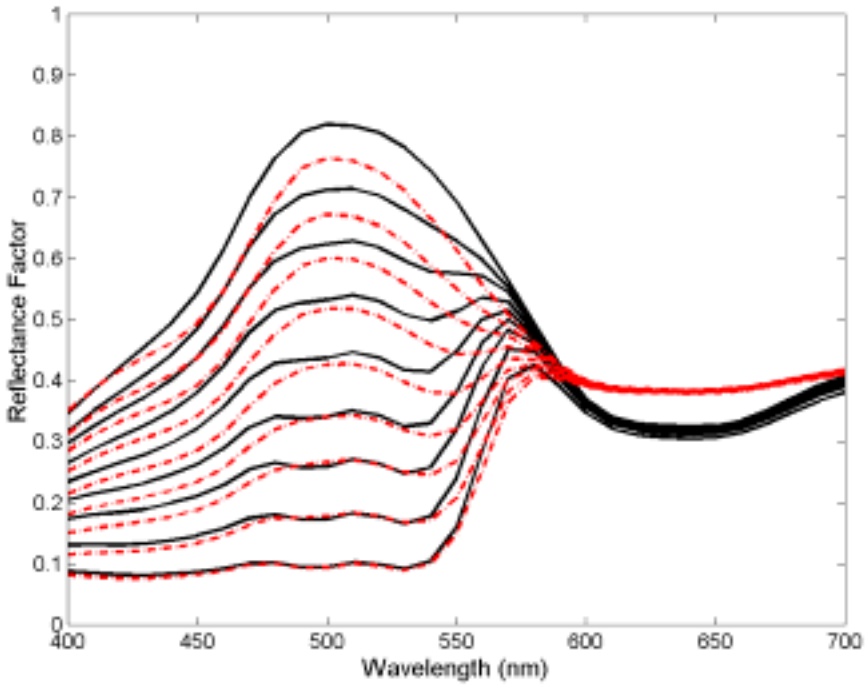


Figure 3-7 - Measured (solid black lines) and predicted (dashed red lines) sample spectra for Neugebauer model with effective area coverages. Green DC is fixed at 88 and Orange DC is varied from 0 to 255.

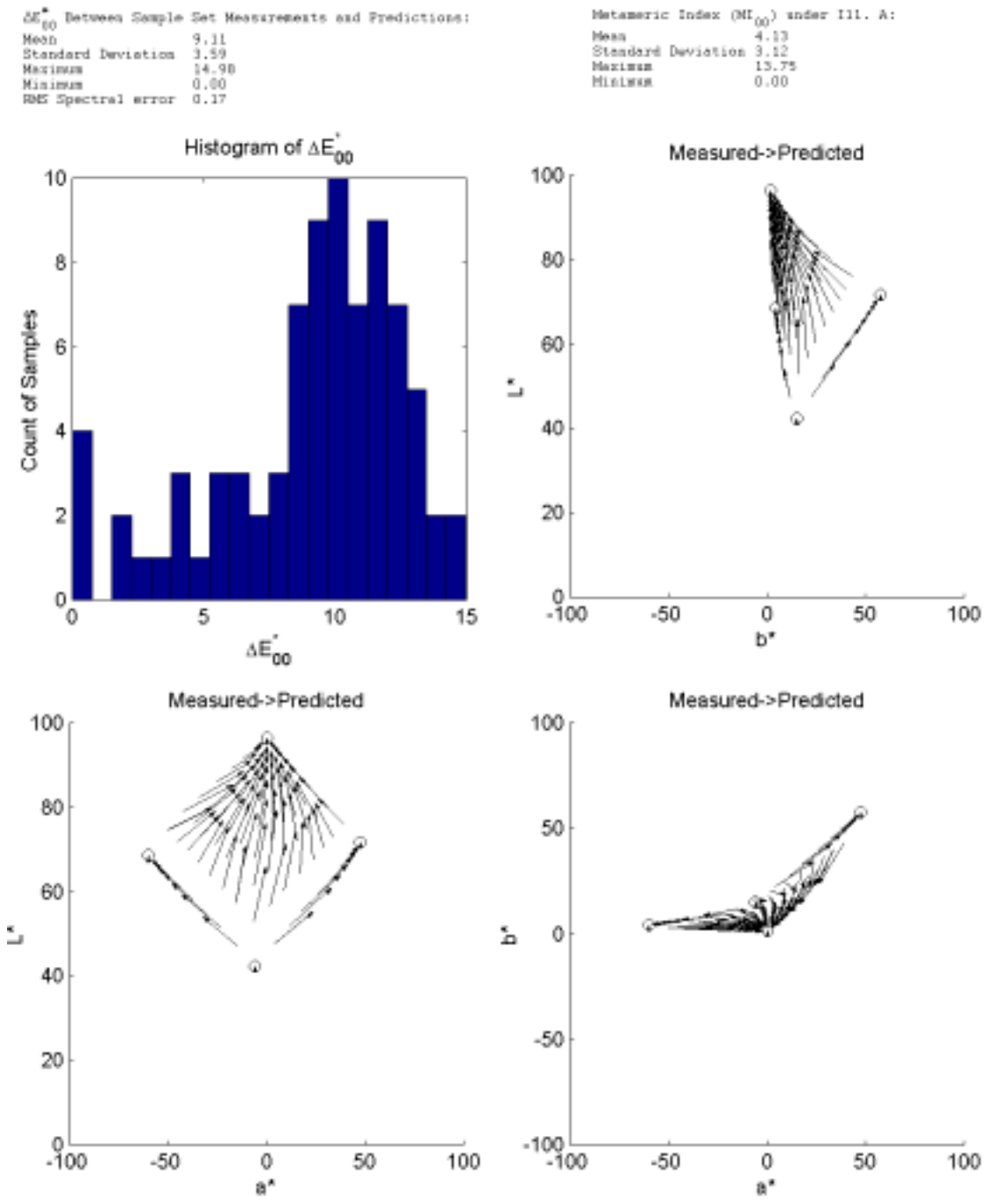


Figure 3-8 - Summary report of Neugebauer model for orange and green ink overprint grid utilizing theoretical area coverages.

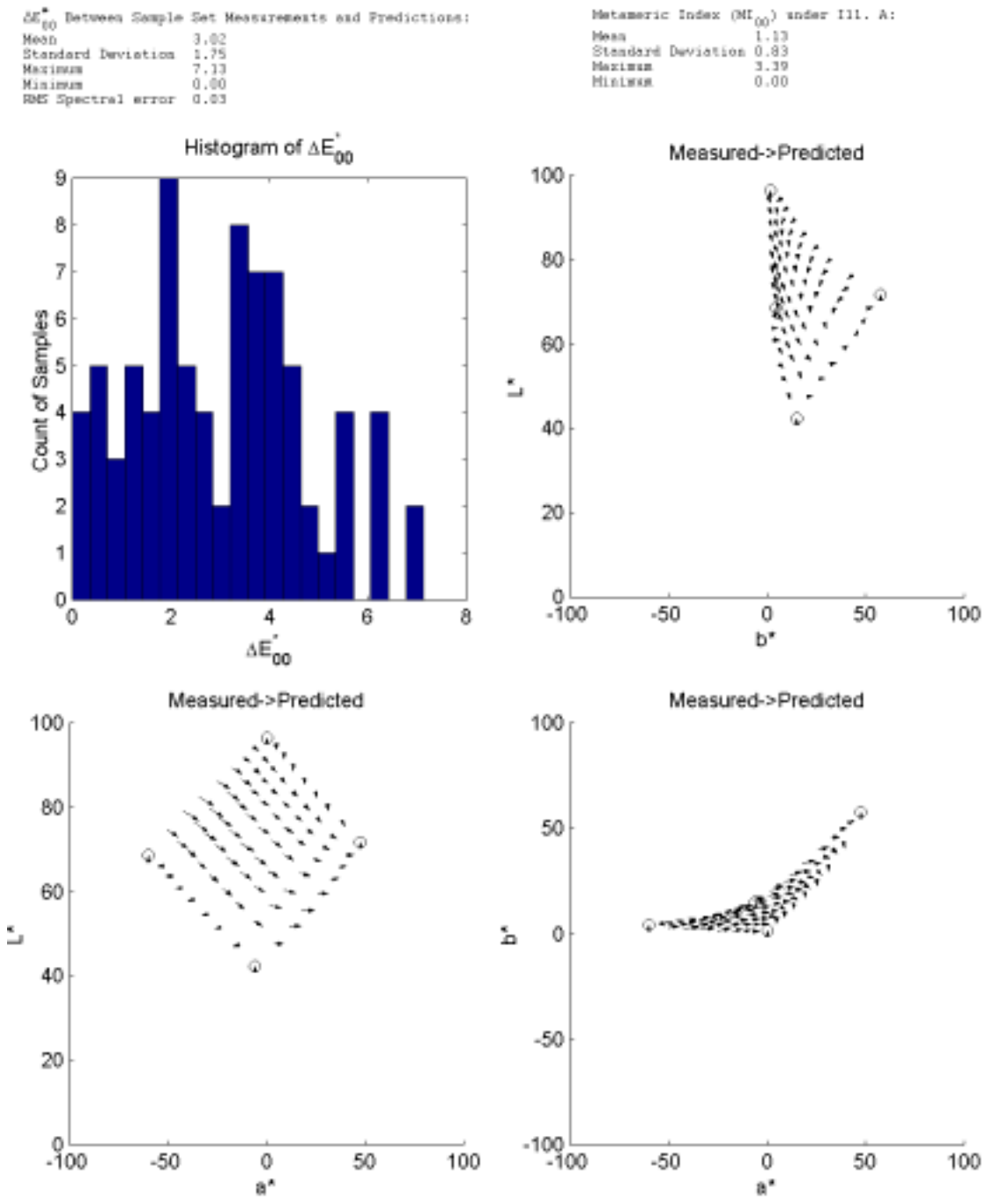


Figure 3-9 – Summary report of Neugebauer model for orange and green ink overprint grid utilizing effective area coverages.

Yule-Nielsen Spectral Neugebauer Model

The improvement in colorimetric accuracy with the addition of the Yule-Nielsen n-value is clearly visible in Figures 3-15 and 3-16. The systematic error with the two different transfer functions appears to be in the same direction as with the previous model but the magnitude is significantly reduced throughout the color space.

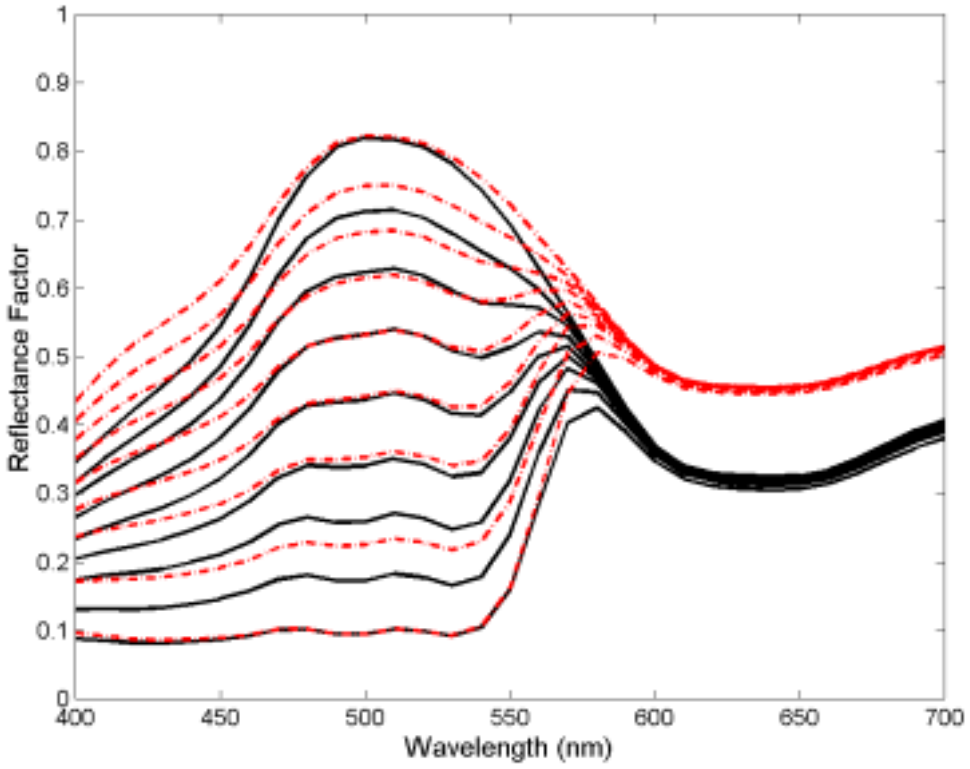


Figure 3-10 - Measured (solid black lines) and predicted (dashed red lines) sample spectra for YNSN model with theoretical area coverages. Green DC is fixed at 88 and Orange is varied from 0 to 255.

Looking at the spectra reveals the improvement to the spectral curve shapes imparted by the Yule-Nielsen correction. However, in Figure 3-10 the same shift in reflectance is visible and in Figure 3-11, a systematic error is still visible. As shown in the summary

report the YNSN model is acceptably small with an average of CIEDE2000 1.55. The maximum error has also been reduced to 4.46.

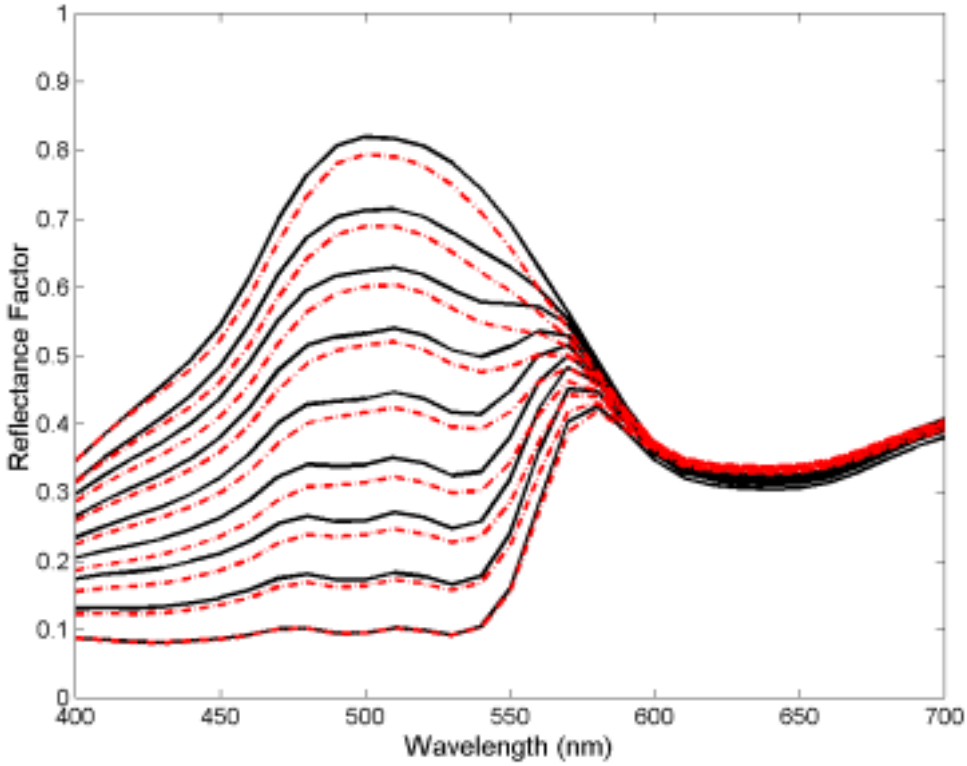


Figure 3-11 - Measured (solid black lines) and predicted (dashed red lines) sample spectra for YNSN model with effective area coverages. Green DC is fixed at 88 and Orange is varied from 0 to 255.

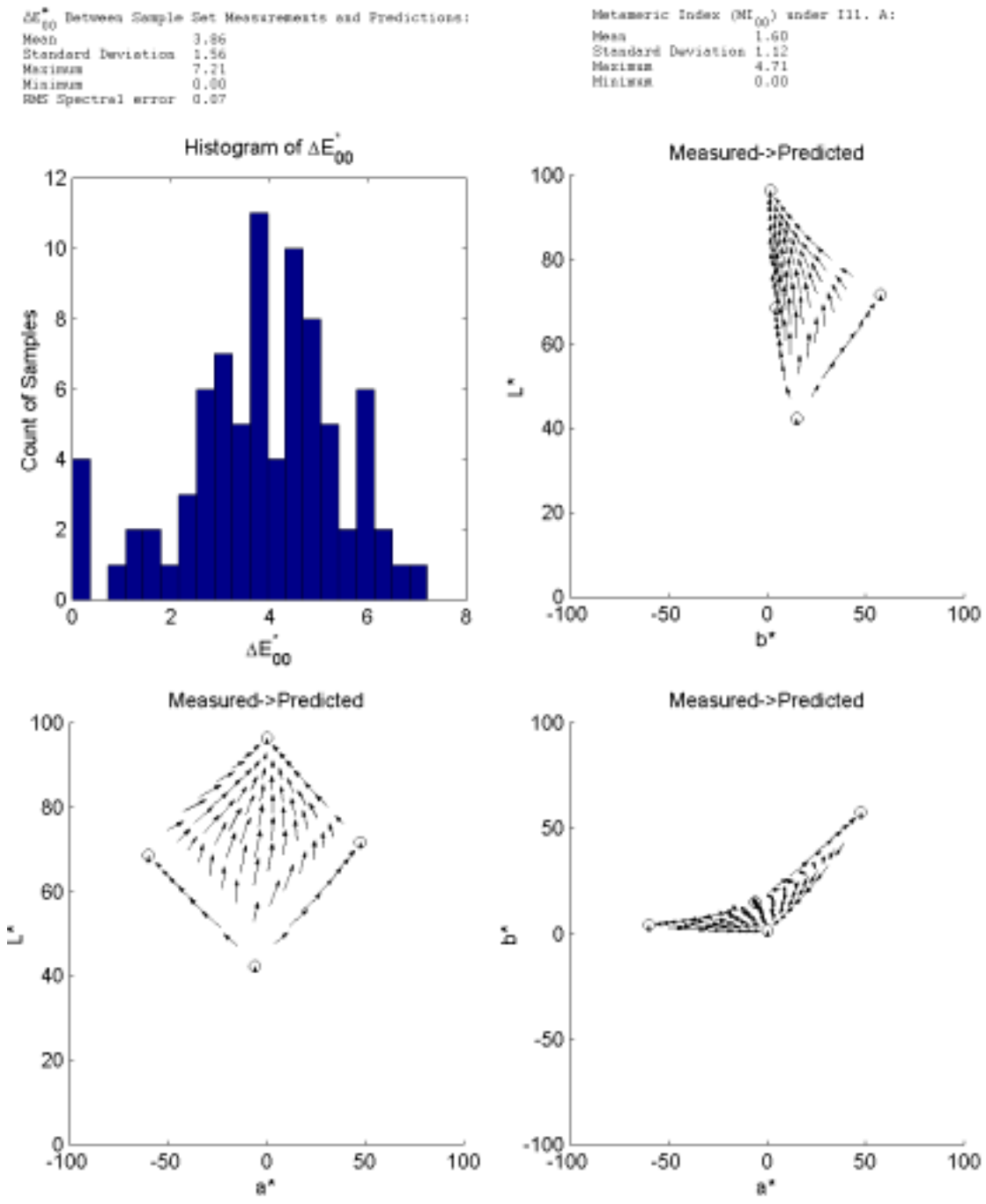


Figure 3-12 – Summary report of YNSN model (n=6) for orange and green ink overprint grid utilizing theoretical area coverages.

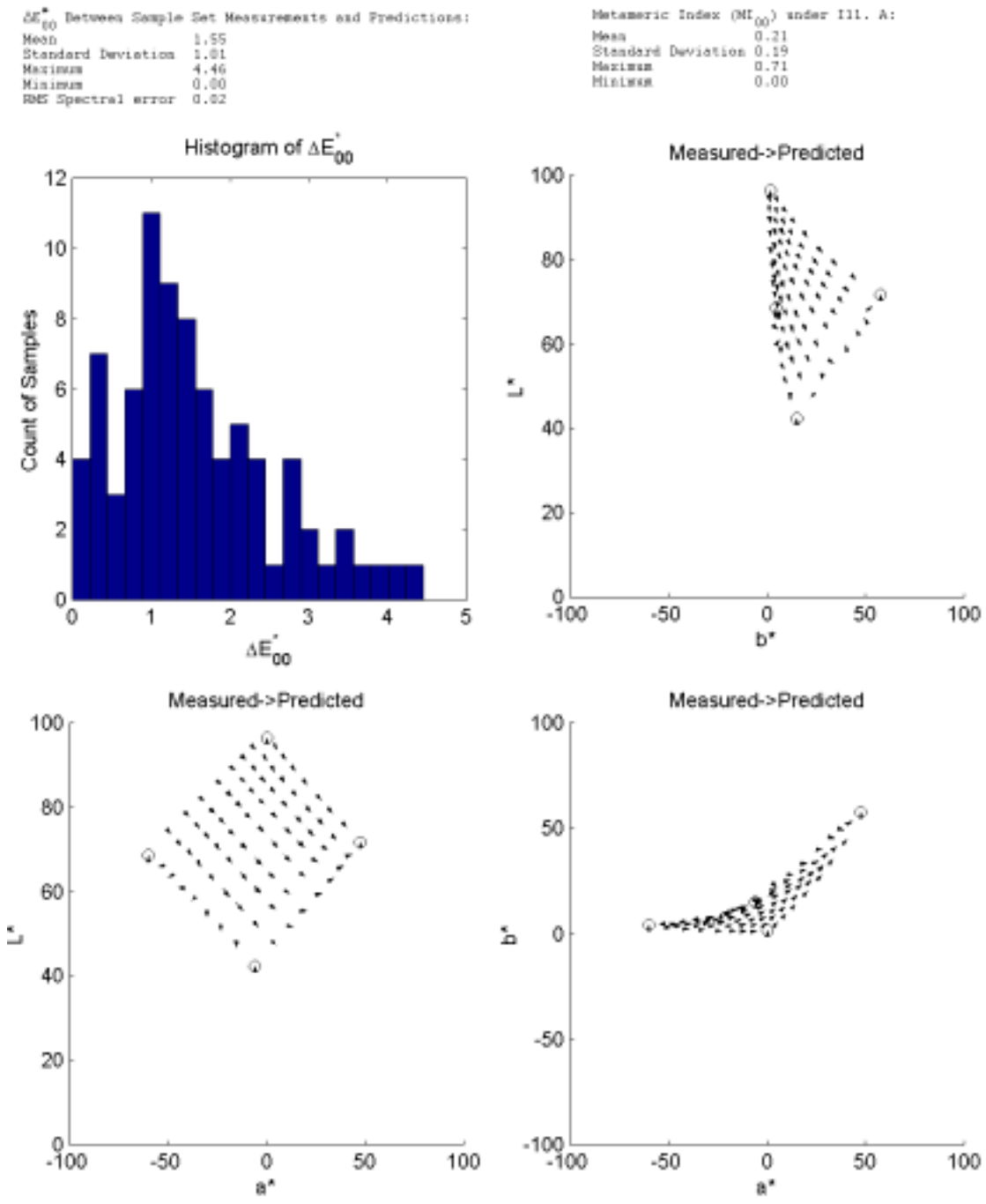


Figure 3-13 – Summary report of YNSN model (n=6) for orange and green ink overprint grid utilizing effective area coverages.

Cellular Neugebauer Model

By dividing the colorant space up into cells and applying the Neugebauer model in a local fashion the colorimetric error was further reduced. Comparing Figure 3-8 with Figure 3-16 shows the extent of this improvement for even the model based on theoretical area coverage. Unfortunately because the error for this model is still unacceptably large the benefit of working in the digital count space directly cannot be realized. When comparing the error histograms for the cellular and non-cellular models it is important to note the number of samples falling into the zero error bins. We expect that all samples that are Neugebauer primaries will be predicted without error since their measured values are used in the equation. However, because there are more of these samples in the cellular model, the mean value is artificially lower than for the global model. In a real-world system few samples would fall exactly on the Neugebauer primaries. The close fit of the predicted to measured spectra is illustrated in Figure 3-15.

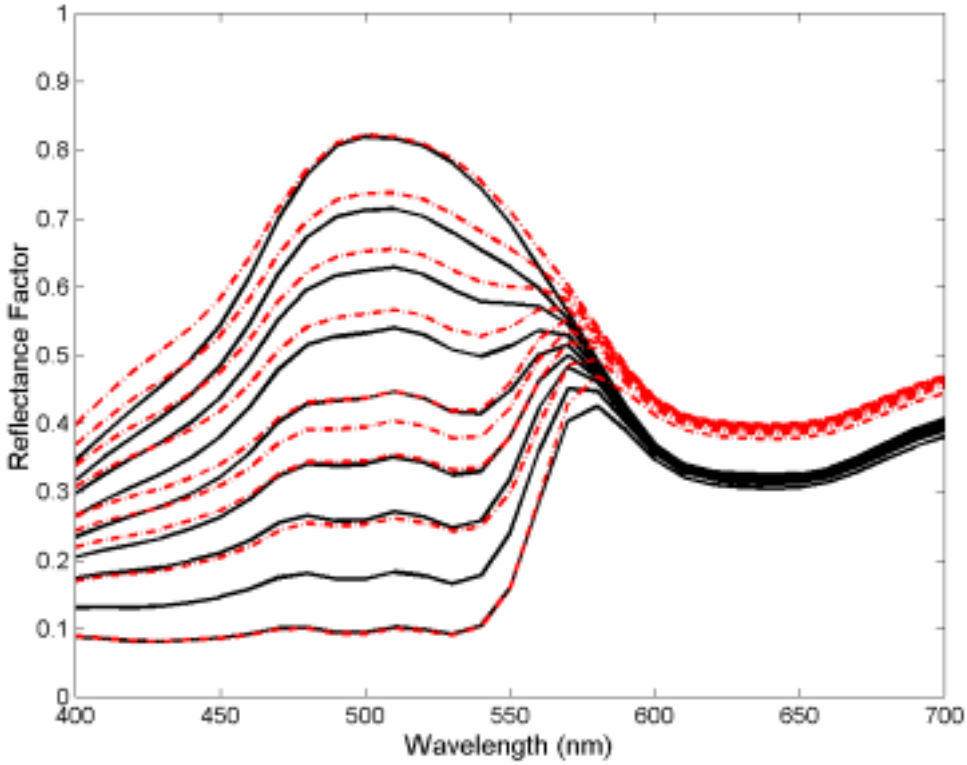


Figure 3-14 - Measured (solid black lines) and predicted (dashed red lines) sample spectra for Cellular Neugebauer model with theoretical area coverages. Green DC is fixed at 88 and Orange is varied from 0 to 255. Note the gross area coverage mismatch of the samples within the cell boundries.

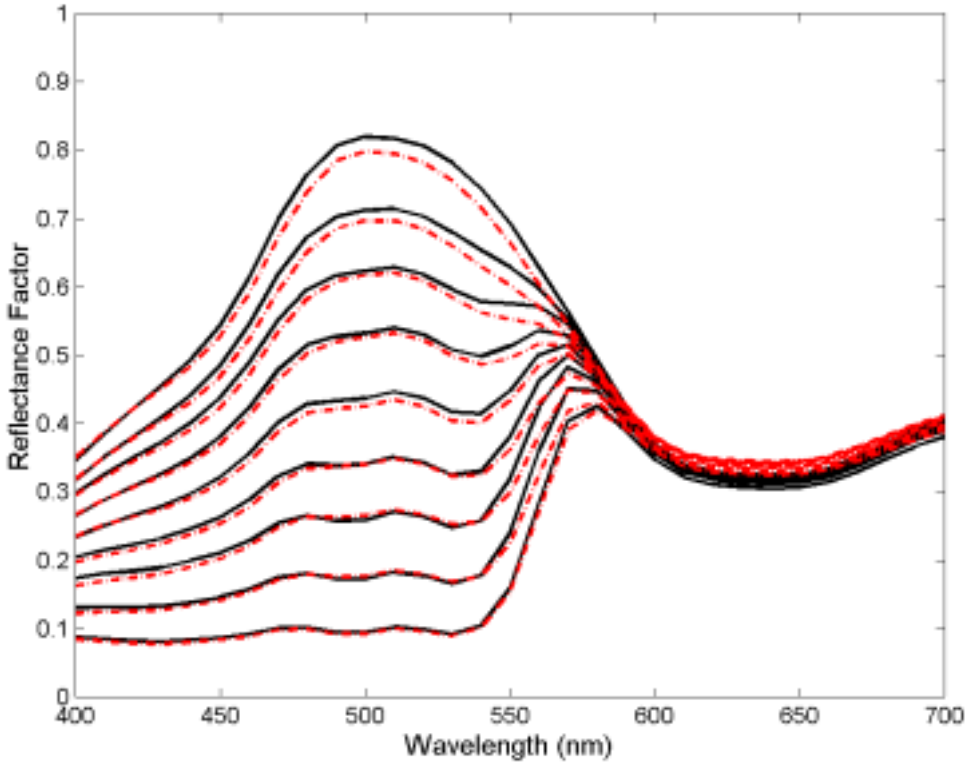


Figure 3-15 - Measured (solid black lines) and predicted (dashed red lines) sample spectra for Cellular Neugebauer model with effective area coverages. Green DC is fixed at 88 and Orange is varied from 0 to 255.

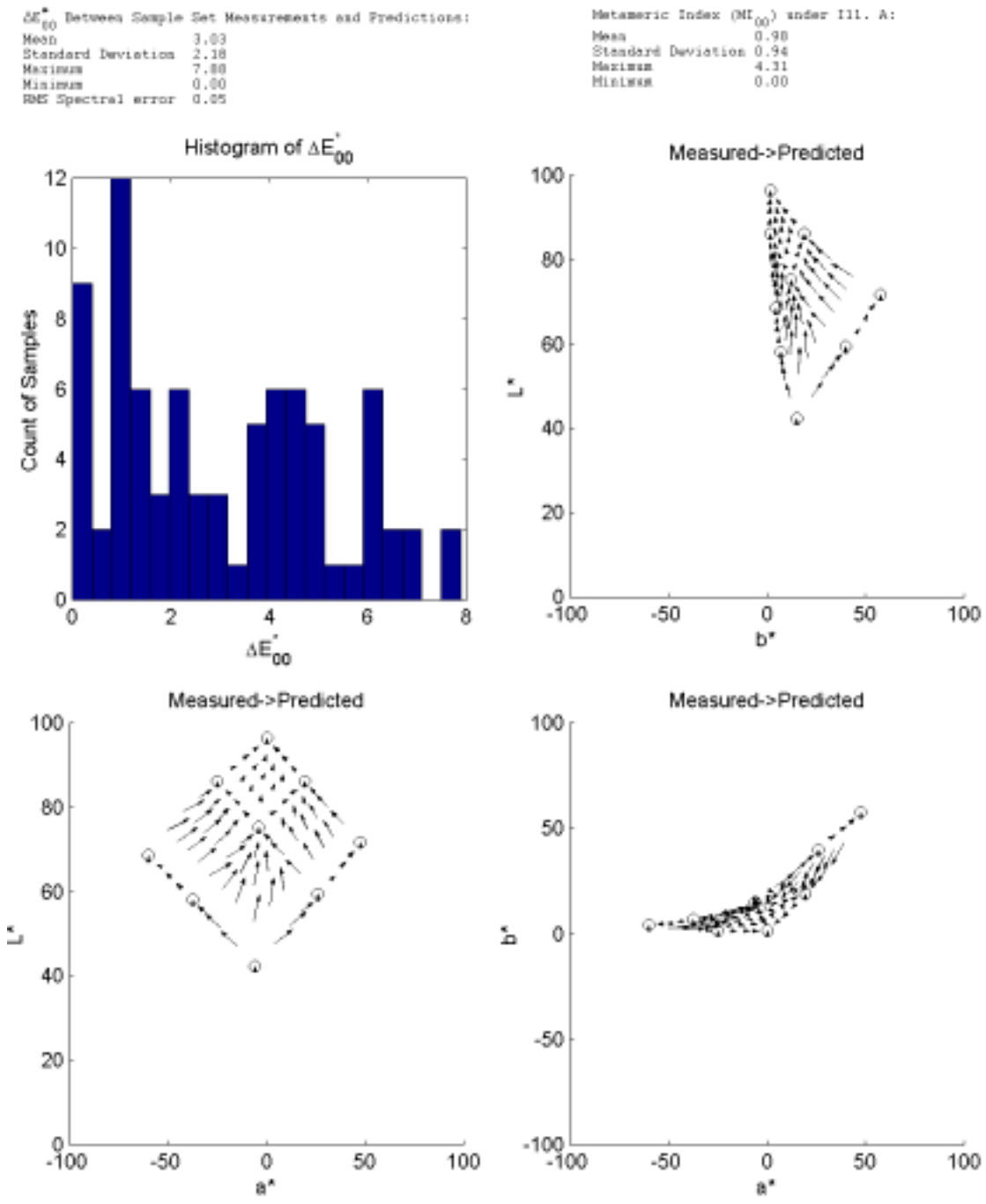


Figure 3-16 – Summary report of Cellular Neugebauer model for orange and green ink overprint grid utilizing theoretical area coverages.

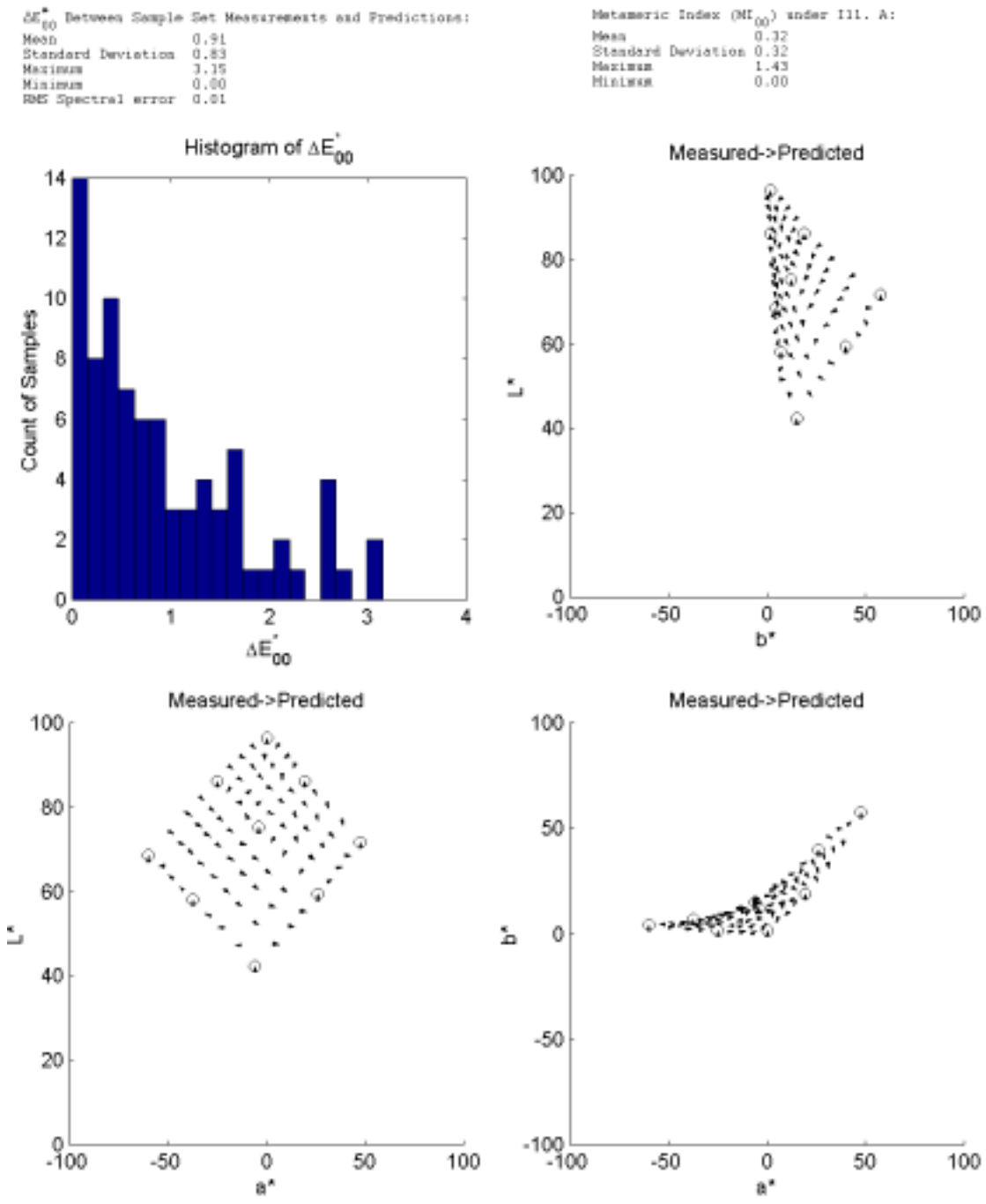


Figure 3-17 - Summary report of Cellular Neugebauer model for orange and green ink overprint grid utilizing effective area coverages.

Cellular Yule-Nielsen Spectral Neugebauer Model

The most complex two color models tested were the ones where the colorant space was divided into cells and the Yule-Nielsen n-value was used. Here too, the advantage in using the effective instead of theoretical area coverage was demonstrated.

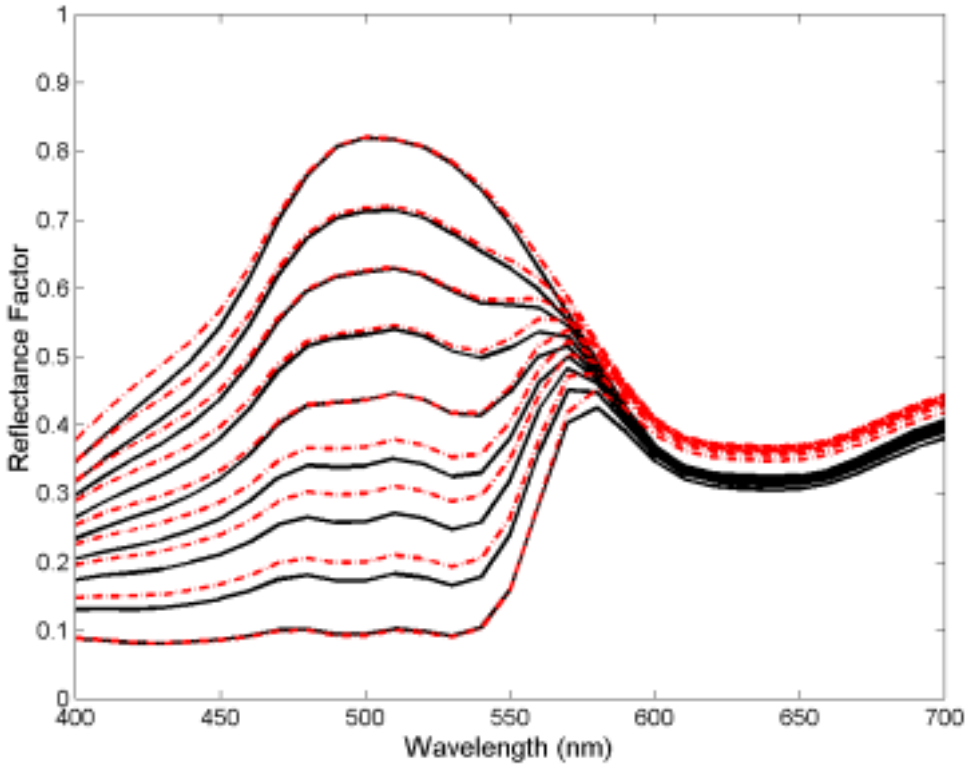


Figure 3-18 - Measured (solid black lines) and predicted (dashed red lines) sample spectra for Cellular YNSN model (n=6) with theoretical area coverages. Green DC is fixed at 88 and Orange is varied from 0 to 255.

As demonstrated by the spectral plots, excellent results were obtained when effective area coverage was used with the cellular model and the Yule-Nielsen correction.

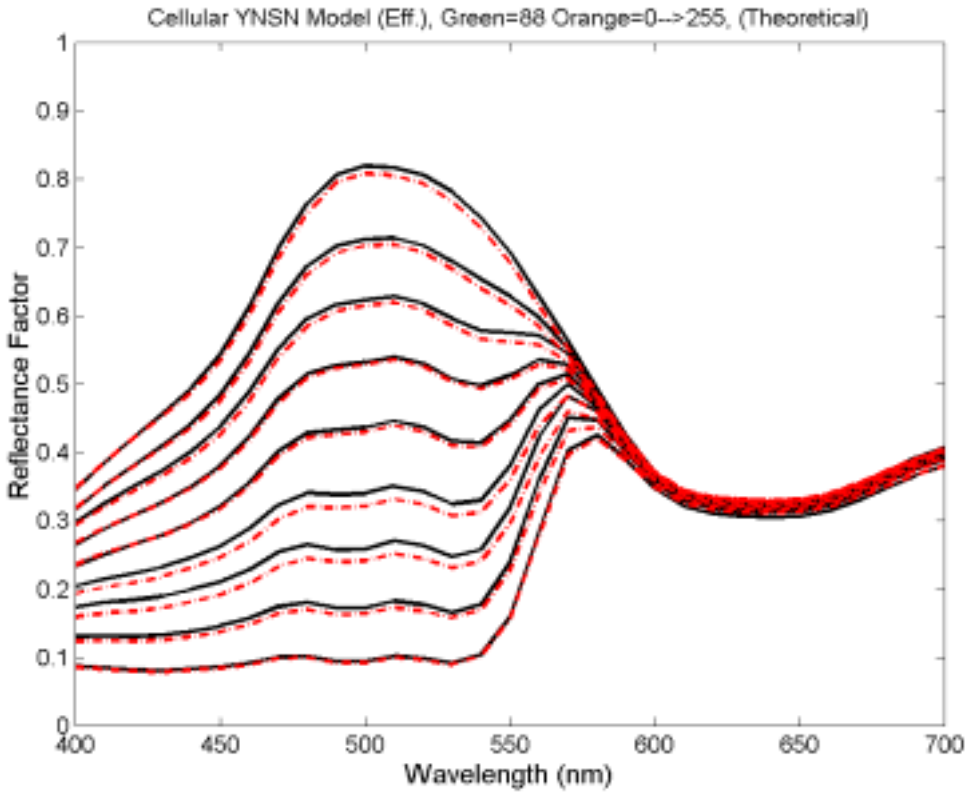


Figure 3-19 - Measured (solid black lines) and predicted (dashed red lines) sample spectra for Cellular YNSN model (n=6) with effective area coverages. Green DC is fixed at 88 and Orange is varied from 0 to 255.

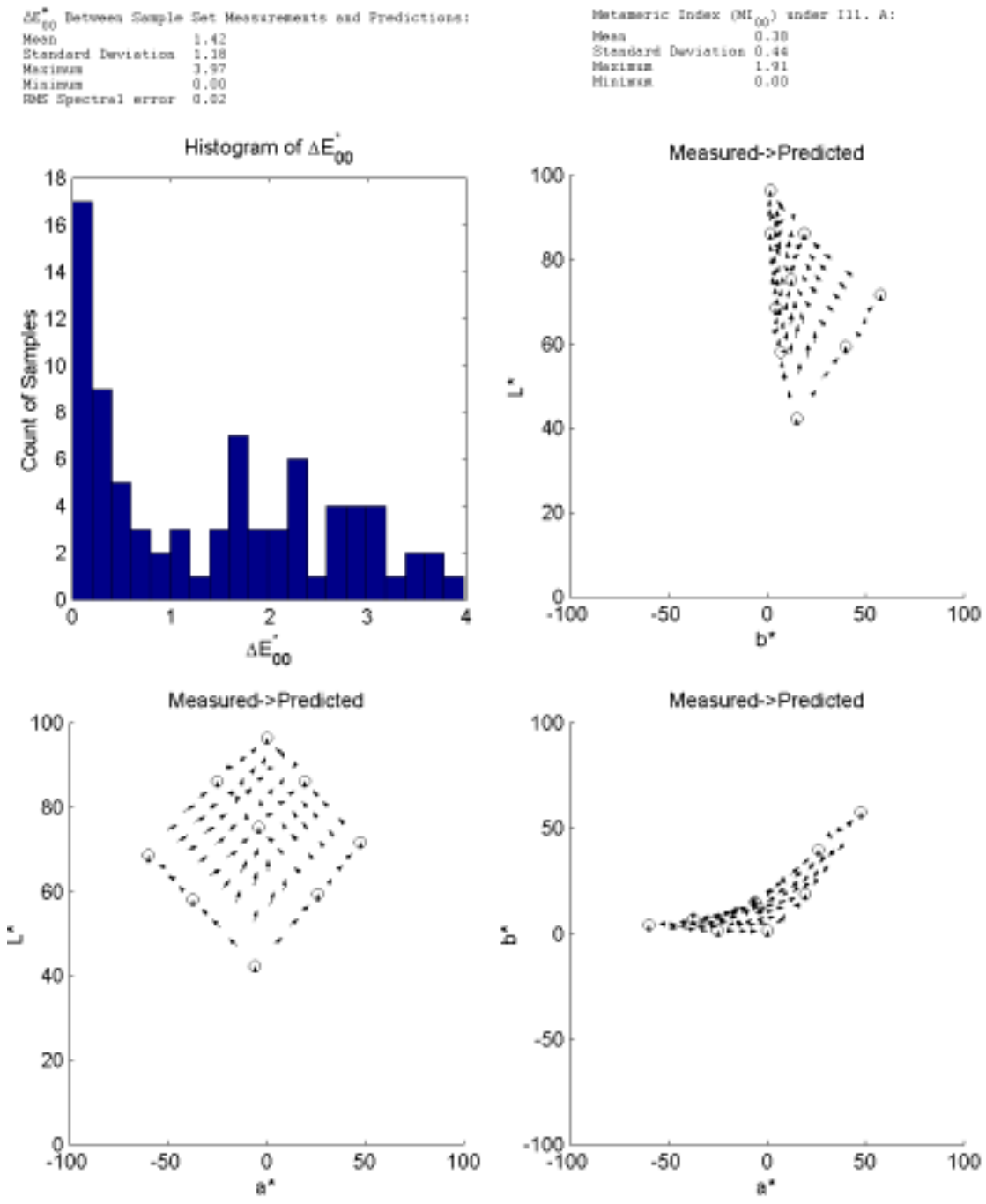


Figure 3-20 – Summary report of Cellular YNSN (n=6) model for orange and green ink overprint grid utilizing theoretical area coverages.

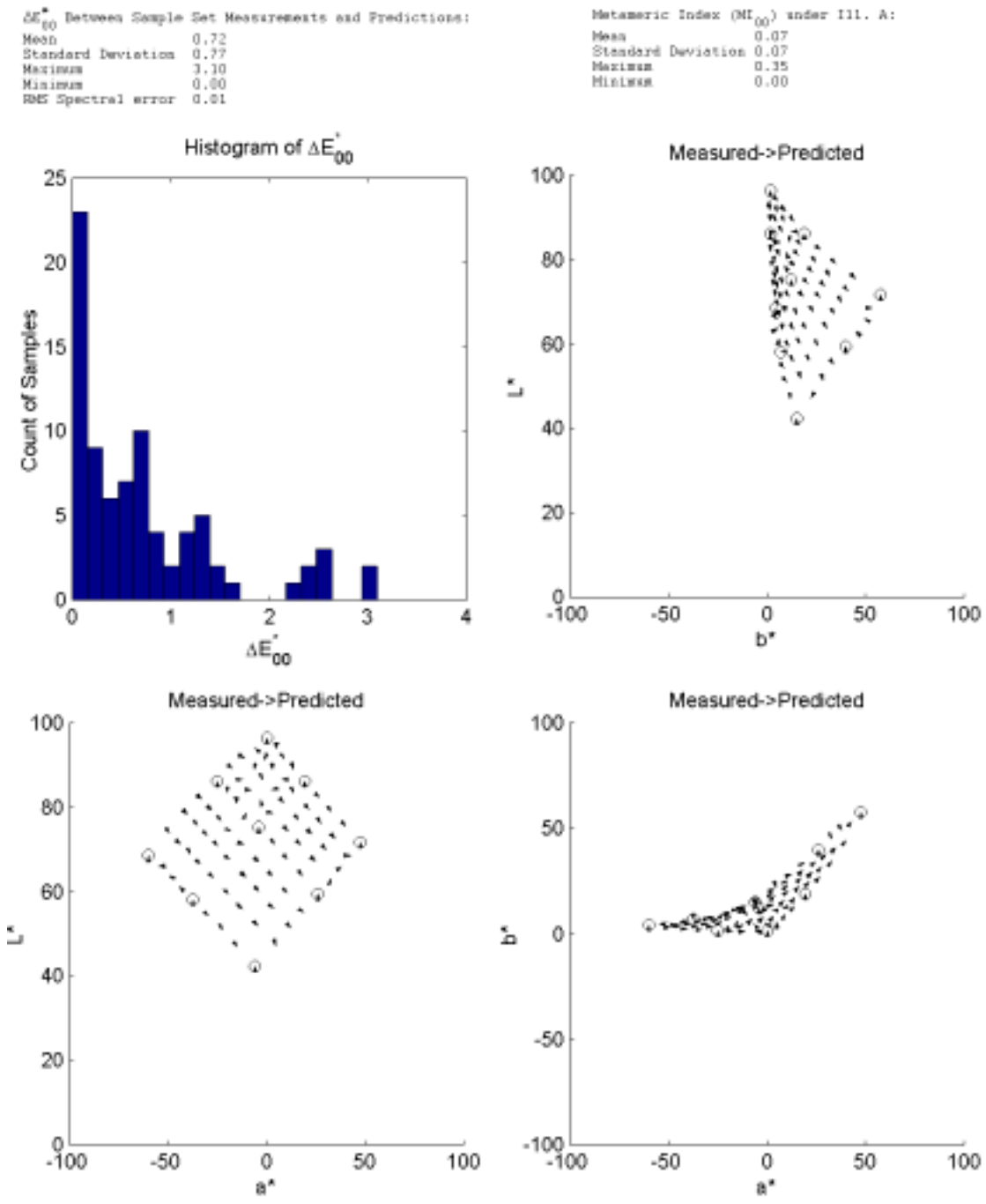


Figure 3-21 – Summary report of Cellular YNSN (n=6) model for orange and green ink overprint grid utilizing effective area coverages.

Single Constant Kubelka-Munk Model

In the continuous tone model, systematic error appeared as a shift towards the full overprint region. This demonstrates the model's inability to accurately predict the two-ink overprint. The mean error for the model was quite small, however the maximum error was still unacceptably large. Looking at the Figure 3-22 spectra might lead one to believe that the model predictions are very close to the measured values. However, the matches for the dark overprint regions have much less accuracy (Figure 3-23).

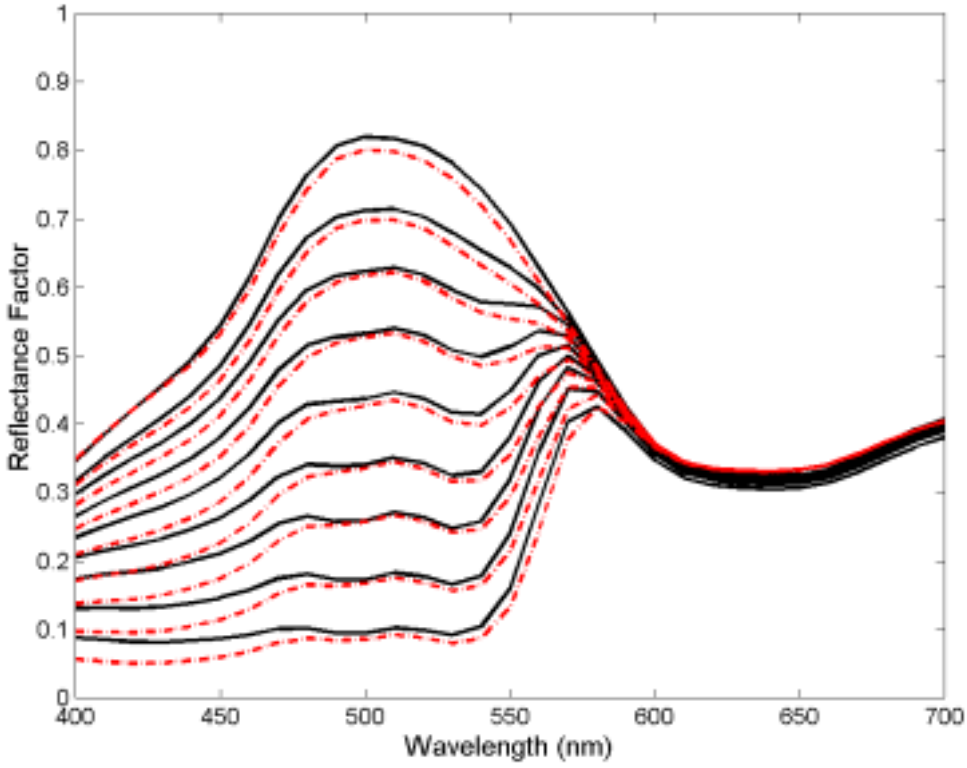


Figure 3-22 - Measured (solid black lines) and predicted (dashed red lines) sample spectra for single constant KM model. Green DC is fixed at 88 and Orange is varied from 0 to 255.

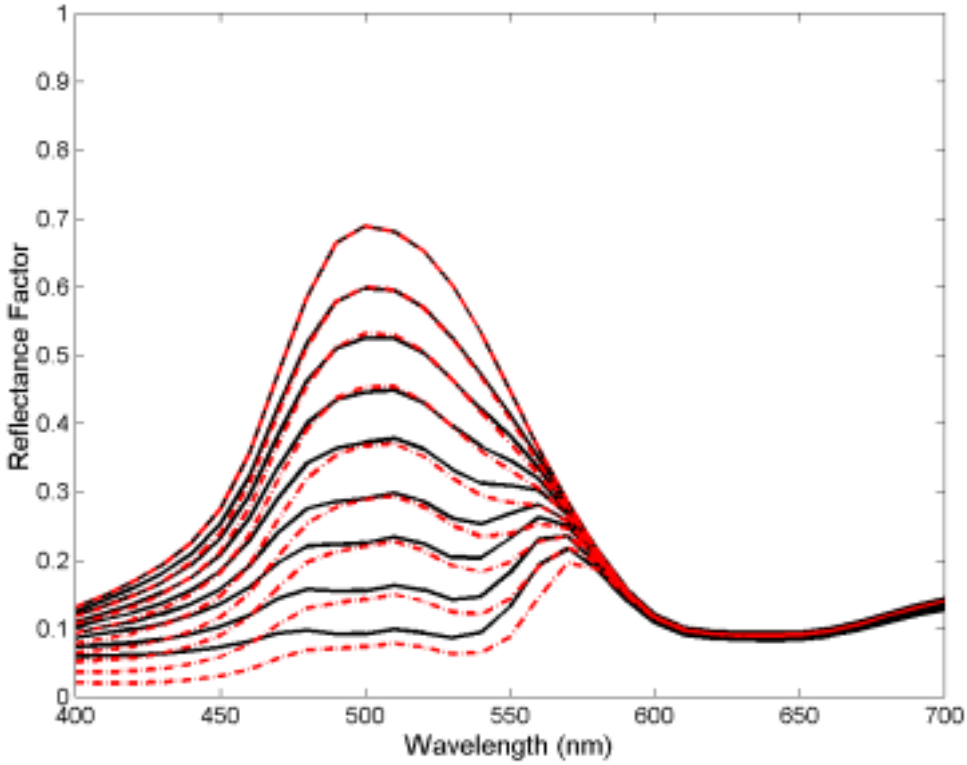


Figure 3-23 - Measured (solid black lines) and predicted (dashed red lines) sample spectra for single constant KM model. Green DC is fixed at 255 and Orange is varied from 0 to 255.

The systematic nature of the spectral error in the above plots translates into systematic colorimetric error as well. This is shown in summary reports below, where the error vector arrows all point towards the two-color overprint region.

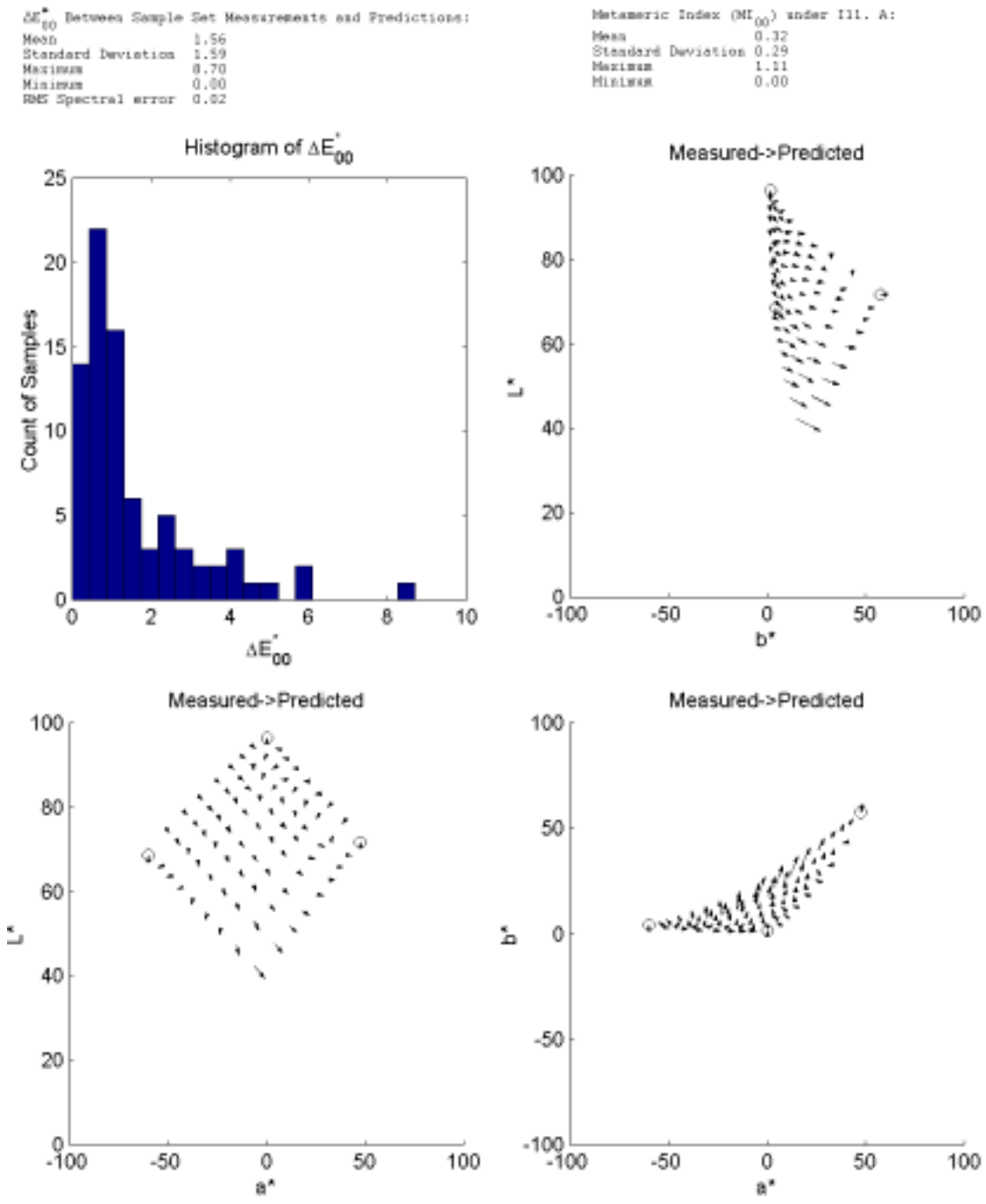


Figure 3-24 – Summary report of single constant KM model for orange and green ink overprint grid.

Cellular Kubelka-Munk Model

Because it is simpler to invert, it was hoped that the continuous tone model would provide accurate predictions of the two color mixtures. However, the large maximum colorimetric error ruled out this class of models as a final predictor of mixture reflectance. Even the local, cellular model was unable to accurately predict the two color overprints. In the case of the Green and Orange inks CIEDE2000 was 6.03 representing only a small improvement over the global version. For the cells containing the lighter mixtures colorimetric error was smaller and less systematic.

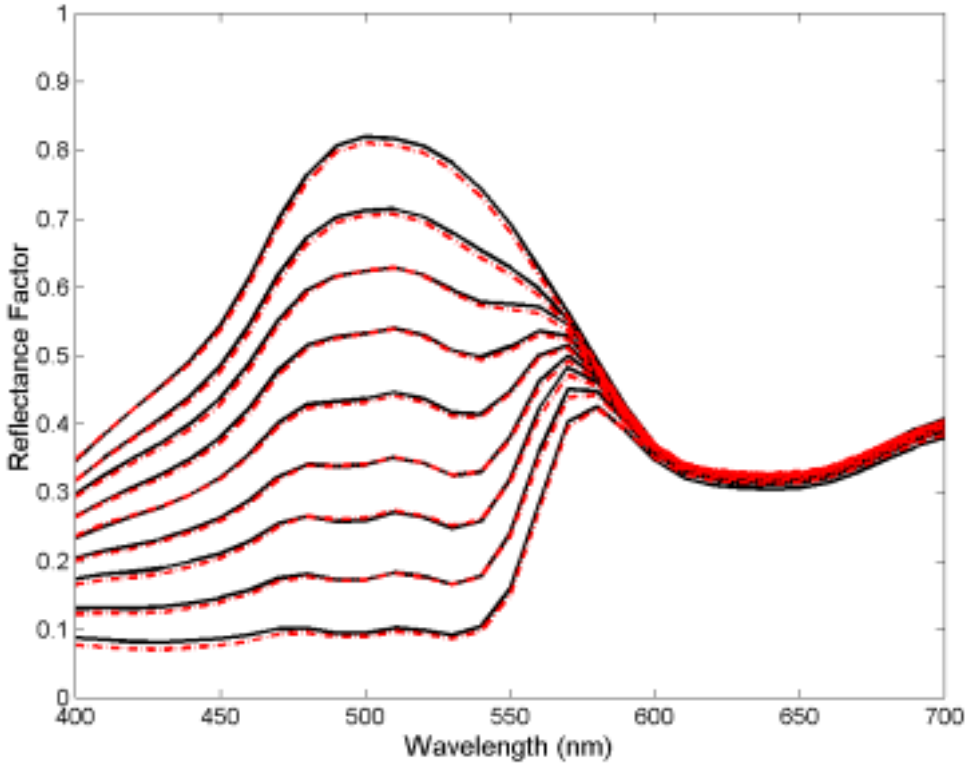


Figure 3-25 - Measured (solid black lines) and predicted (dashed red lines) sample spectra for cellular single constant KM model. Green DC is fixed at 88 and Orange is varied from 0 to 255.

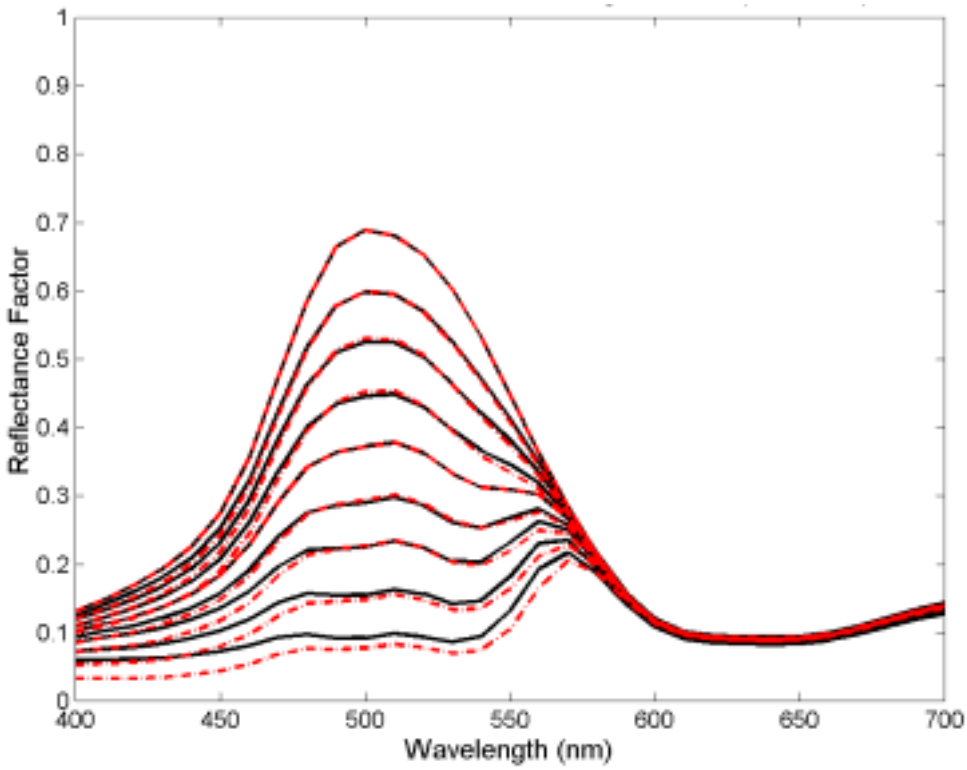


Figure 3-26 - Measured (solid black lines) and predicted (dashed red lines) sample spectra for cellular single constant KM model. Green DC is fixed at 255 and Orange is varied from 0 to 255.

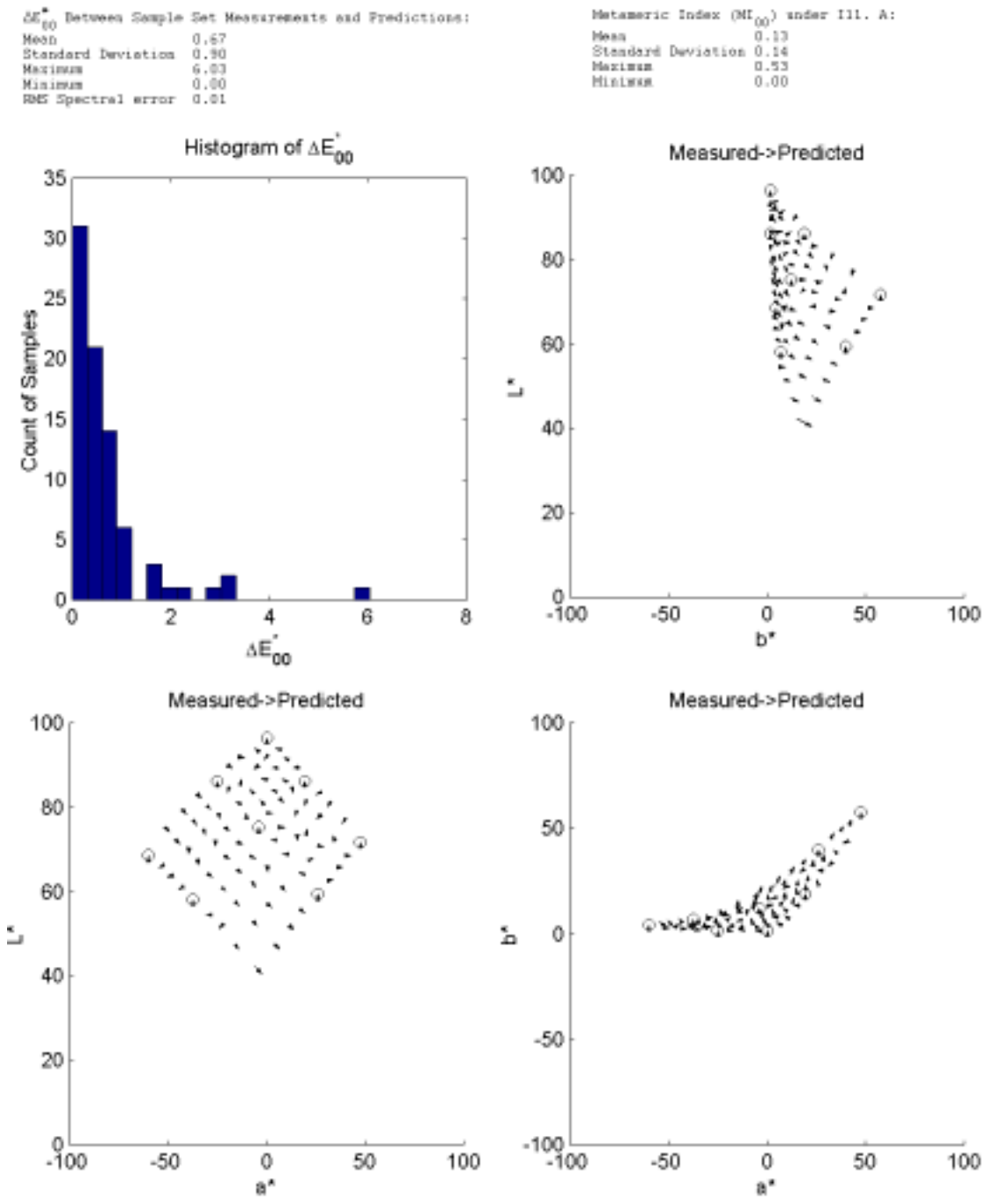


Figure 3-27 – Summary report of the cellular continuous-tone model for the orange and green inks.

Conclusions

The best performing model, Cellular YNSN, was not surprisingly the one that used the largest number of primaries to predict the mixture spectra. Additionally, the importance of using effective rather than theoretical area coverage was graphically demonstrated by the large and systematic errors in the models that used them. It was unfortunate that the continuous-tone based models had such large maximal errors, as the inversion of this type of model is much simpler. Ultimately the YNSN model with the effective area coverage LUT was selected for further testing and expansion to the full six-color system. The decision to not use the cellular version was made based on the smaller number of samples needed to characterize the printer, simpler implementation, and the reasoning that the cellular partitioning could always be added later if it was needed.

4. SIX-COLOR FORWARD MODEL

Evaluation of different forward predictive models using the two color overprints indicated that the Yule-Nielsen-spectral-Neugebauer model was a good candidate for extension to the six-color model. In this section the accuracy of the forward model will be evaluated for the full six color case. Additionally, the results from the YNSN model will be compared to those obtained using a local cellular version with an expanded number of primaries.

Six-Color Yule-Nielsen-Spectral-Neugebauer (YNSN)

The Yule-Nielsen modified spectral Neugebauer equation (YNSN) used in this research is defined in Equations (4.1) and (4.2) for the general case of K inks under the Demichel constraints:[64-68,73]

$$\hat{R}_\lambda = \left(\sum_{i=1\rightarrow 2^K} p_i R_{\lambda,i,\max}^{1/n} \right)^n \tag{4.1}$$

$$p_i = \prod_{j=1\rightarrow K} \begin{pmatrix} \text{If ink } j \text{ is in Neugebauer Primary } i\text{, then } a_j \\ \text{Else, } (1-a_j) \end{pmatrix} \tag{4.2}$$

$$a_i = \text{LUT}(dc_i) \tag{4.3}$$

where, $\hat{R}_\lambda$, the estimated reflectance is based on the weighted sum of the reflectances of the possible full area coverage overprint combinations (Neugebauer primaries). Each primary is weighted by a scalar, p_i, whose values is based on the product of ink area

coverages making up the primary. The area coverage is determined from the digital count (dc_i) through a lookup-up-table like the one shown in Figure 4-3. With six inks there are 64 possible combinations. These were printed as patches and measured spectrally using the calibration target shown in Figure 4-1.

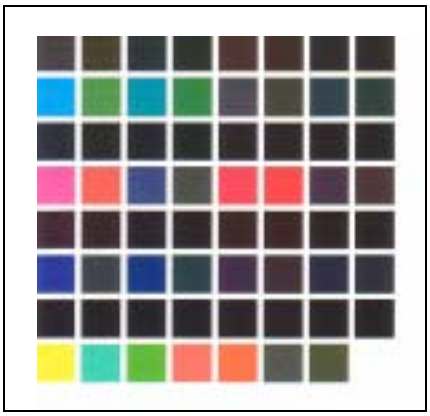


Figure 4-1 - Calibration target of 64 Six-Ink Neugebauer Primaries

As visible in Figures 4-5 and 4-6, many of the Neugebauer primaries, particularly the half containing black ink have very low reflectance factors across the visible spectrum. This means that to some degree the model accuracy is dependent on the spectrophotometer's precision and accuracy in measuring high-density samples.

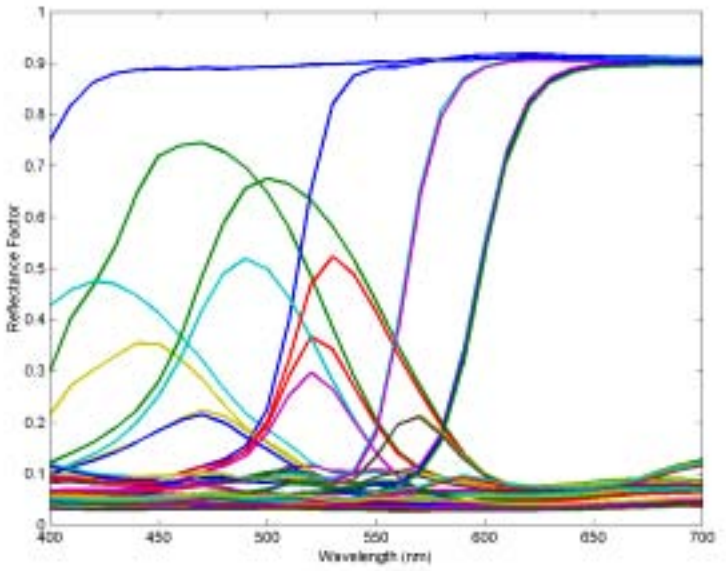


Figure 4-2 - Reflectance spectra of 64 Neugebauer primaries from the YNSN model.

Conversion from digital count to effective area coverage

The nonlinear relationship between printer digital counts and effective area coverage was demonstrated in the previous section. The same spline based lookup tables were used again without modification, based only on single-ink ramps and applied directly to the six-color model. Figure 4-3 shows the relationship for the magenta ink; the other inks behaved in a similar fashion.

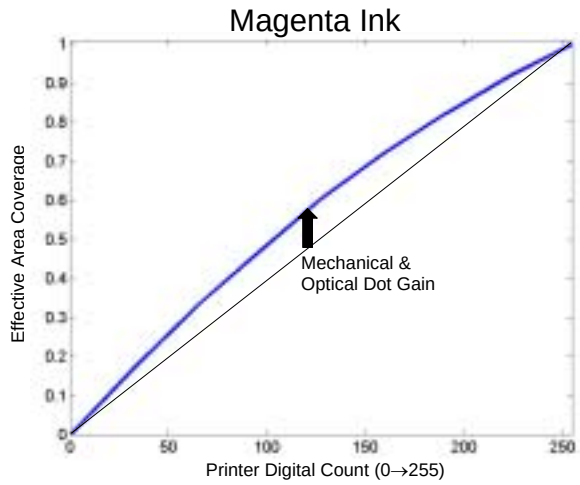


Figure 4-3 - Relationship between printer digital counts and effective area coverage for magenta ink.

Yule-Nielsen n-value

The Yule-Nielsen n-value of six, selected in the two-color model section was used for six-color model as well.

Six-Color Cellular YNSN

The benefit of partitioning the colorant space and using localized models was demonstrated in the two-color model evaluation section. Equations (4.1) and (4.2) were used to define a cellular model by adding in a modified definition for effective area coverage based on additional Neugebauer primaries that more closely surround the sample in colorant space. Specifically, the primaries with the next highest and lowest area coverages are used. The effective area coverages of each ink is used in the renormalization:

$$a_{eff,cellular} = \frac{a_{eff} - a_{eff,lower}}{a_{eff,upper} - a_{eff,lower}} \tag{4.4}$$

In the YNSN model, the Neugebauer primaries are printed with only with full or no coverage. The first round of testing for the cellular version added an additional coverage level near fifty percent effective area coverage. With three coverage levels the colorant space is divided into sixty-four cells and 729 primaries must be printed and measured for calibration. These samples are shown in Figure 4-4 and the wide range of spectra are shown in Figure 4-5.

Figure 4-4 - Targets displaying 729 Neugebauer primaries of Cellular YNSN model.

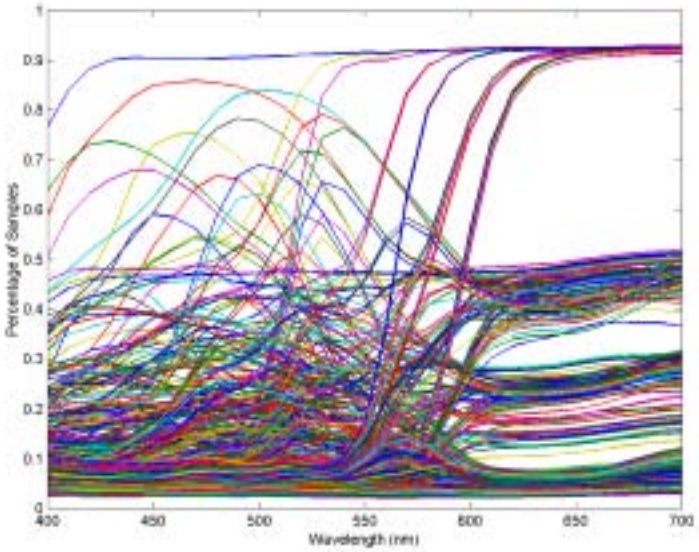


Figure 4-5 - Reflectance spectra of 729 Neugebauer primaries used in cellular YNSN model.

Experimental

The accuracy of the forward model was tested using two datasets. The first was a full six-way factorial division of the colorant space (5^6 = 15,625 samples) and the second the midpoints of that division (4^6 = 4,906 samples). The starting point for the testing was printer digital counts. The digital counts were converted to effective area coverage using the spline based lookup tables and fed through the forward models (YNSN and Cellular-YNSN) to produce predicted reflectance spectra for each sample in the dataset. The digital counts used for the five-way factorial are shown in Table 4-I, and those for the midpoints in Table 4-II.

Table 4-I - Digital counts used in 5^6 factorial sampling of colorant space (15,625 samples).

Black	Cyan	Magenta	Yellow	Green	Orange
0	0	0	0	0	0
20	23	28	25	26	24
50	58	66	59	62	59
96	111	127	114	117	112
255	255	255	255	255	255

Table 4-II - Digital counts used in 4^6 factorial midpoint sampling of colorant space (4,096 samples).

Black	Cyan	Magenta	Yellow	Green	Orange
9	11	13	11	12	11
33	38	46	40	41	40
69	81	93	83	87	81
144	158	178	160	166	159

Although the samples are well spaced in colorant space they do not evenly span CIELAB color space. As shown in Figure 4-6 the dark regions of the color space are heavily over sampled corresponding to samples with high area coverage of multiple inks. Because so many samples fall into the dark region of color space, where necessary, CIE ΔE_{00} was used as a colorimetric error metric since it more accurately estimates perceptual color differences in this region than previous color-difference equations.

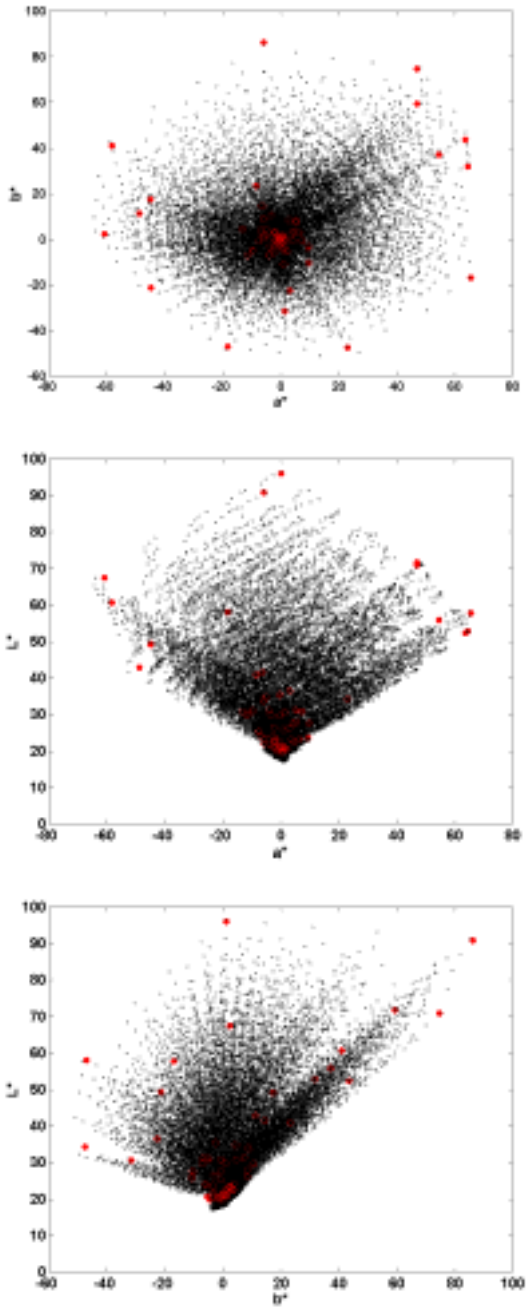


Figure 4-6 - Colorimetric distribution of 5^6 factorial samples. Red markers indicate locations of 64 Neugebauer primaries.

Printed Samples

Since the objective at this stage was to evaluate the accuracy of the forward model it was necessary to physically print all of the sample digital counts within the two datasets and then measure them spectrally. Each of the printed targets also included a small number of identical patches to monitor print-to-print and spatial uniformity of the printing system. Mean print-to-print variability was found to be near 0.2 CIE2000 units with a maximum color difference of 0.6. The verification patches were rather light with L* values ranging from 76 to 91. The uniformity is likely lower for darker samples. A total of thirty-two pages and with over twenty four thousand patches were printed and measured.

Results and Discussion

After printing and measuring all of the samples, spectral RMS error was calculated between the predicted and measured values for all of the samples. For both the YNSN and Cellular-YNSN models the mean RMS spectral error was less than one percent.

Table 4-III - RMS Spectal error between model estimations and measured reflectance spectra.

Model	Sample Set	Mean RMS	Max RMS	StdDev RMS
YNSN	5^6 Factorial	8.53e-03	7.03e-02	5.36e-03
YNSN	4^6 Midpoint	4.60e-03	2.32e-02	2.94e-03
Cellular YNSN*	5^6 Factorial	4.39e-03	4.83e-02	4.19e-03
Cellular YNSN*	4^6 Midpoint	4.56e-03	1.91e-02	2.78e-03

***Colorant space divided into 64 cells, leading to 729 Neugebauer primaries being used.**

A colorimetric comparison of the measured and predicted samples was also conducted. The DE2000 (2-deg, ill D65) mean, maximum and standard deviation for the two models and sample sets are shown in Table 4-IV. The accuracy of the YNSN model is excellent and certainly acceptable for most image reproduction applications.

Table 4-IV - Colorimetric error between model estimations and measured reflectance spectra.

Model	Sample Set	Mean ΔE_{00}	Max ΔE_{00}	StdDev ΔE_{00}
YNSN	5^6 Factorial	1.60	5.32	0.66
YNSN	4^6 Midpoint	0.97	3.22	0.43
Cellular YNSN*	5^6 Factorial	0.83	3.81	0.55
Cellular YNSN*	4^6 Midpoint	0.94	2.81	0.41

***Colorant space divided into 64 cells, leading to 729 Neugebauer primaries being used.**

Since so many samples were evaluated, the distribution of RMS spectral error was plotted as a histogram to roughly gauge its distribution. Next, the samples associated with the tail of the distribution, the samples with errors of 0.02 or greater and 0.03 or greater were examined more closely to see if they could be clustered together by a common feature. Figure 4-9 shows the colorimetric projection of all of the 5^6 samples. The red points indicate the location of the RMS spectral errors greater than 0.02 and the green points those greater than 0.03. Most of the high error samples cluster along a line in color space that was later determined to be associated with high concentrations of magenta, yellow and orange inks.

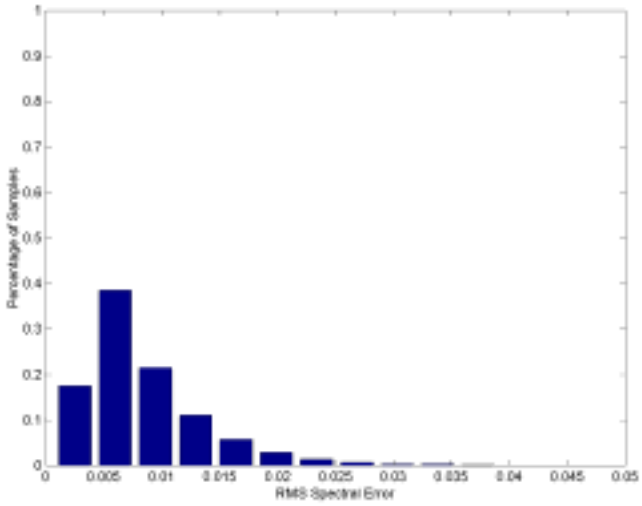


Figure 4-7 - Histogram of RMS spectral error distribution for five-way division of colorant space as predicted using the six color YNSN model (15,626 Samples).

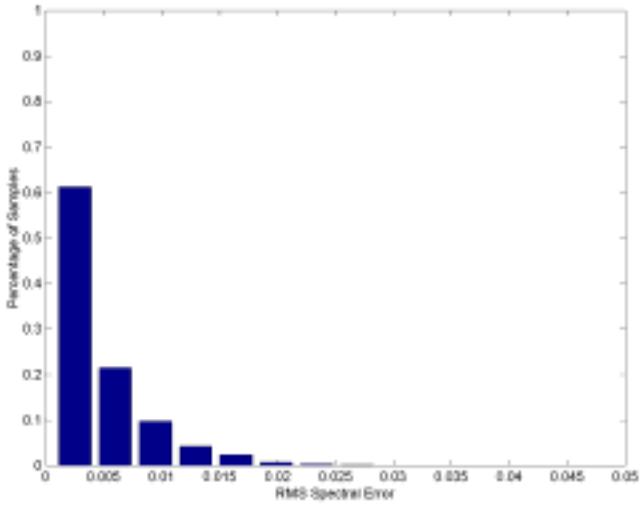


Figure 4-8 - Histogram of RMS spectral error distribution for five-way division of colorant space as predicted using the six-color cellular YNSN model with 729 neugebauer primaries (15,626 Samples).

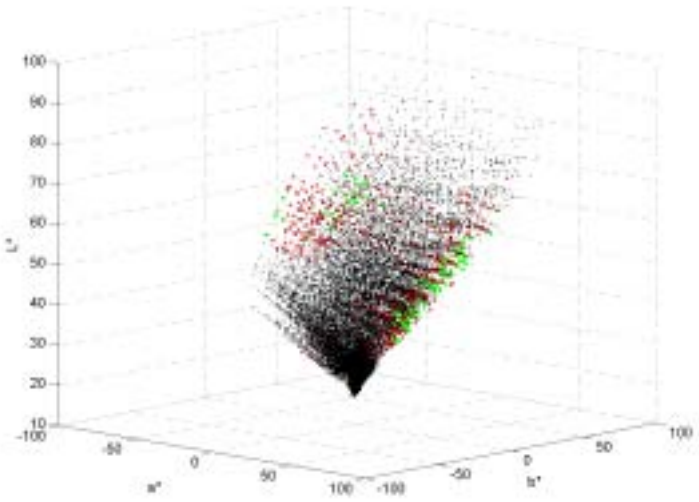


Figure 4-9 - RMS Spectral error distribution of 5^6 Factorial Sampling of colorant space as predicted with six-color YNSN model. Red markers indicate error >0.02 and green markers >0.03.

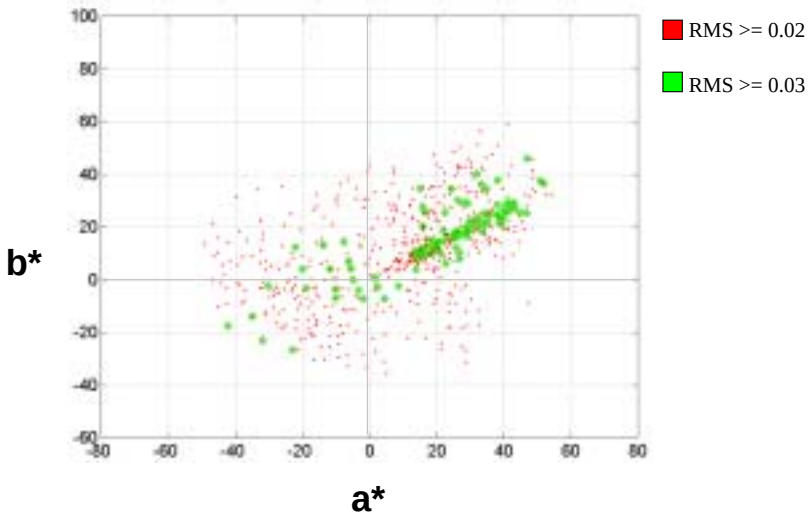


Figure 4-10 - Colorimetric distrbution of RMS spectral error for 5^6 sampling of colorant space predicted by six-color YNSN model.

By looking at the difference between the predicted and measured spectra of Figure 4-11, the systematic nature of the error becomes apparent. The effective area coverage

associated with the larger cluster of samples seems to indicate a problem closely tied to the Magenta-Yellow-Orange primary. This region of high error also appears in the Cellular-YNSN model where the primaries are located closer to the samples and local linearity of the color-mixing model would be expected to lead to more accurate estimates.

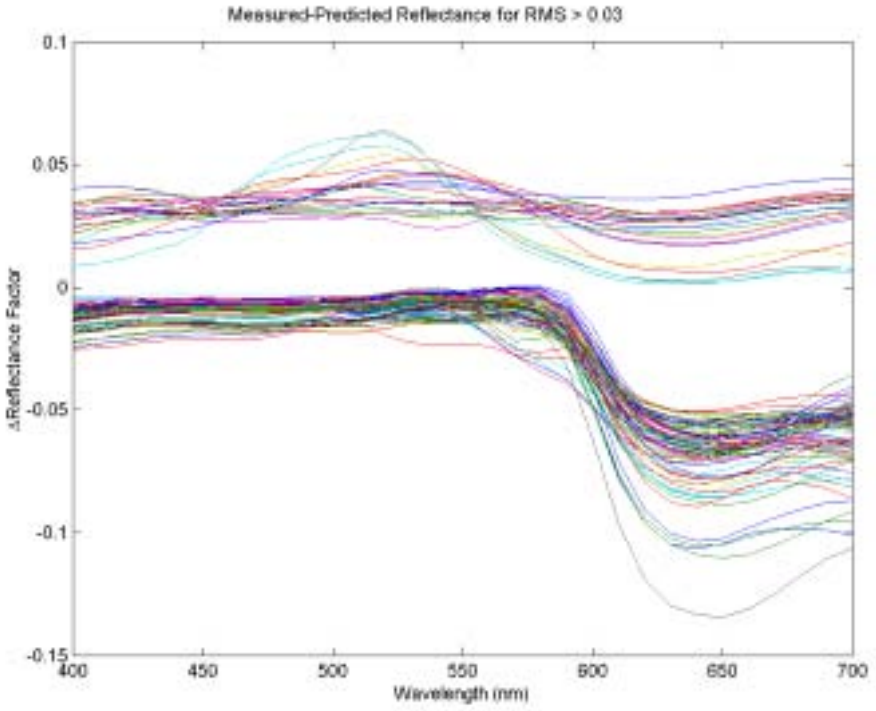


Figure 4-11 - Difference in measured and predicted spectral for the 100 samples with RMS spectral error greater than 0.03.

By using the larger five-way factorial dataset as primary data for the cellular model, the colorant space was divisible into 4,096 cells. The midpoints of those cells, the second dataset, were predicted using the highly localized cellular model. The colorimetric (Table 4-V) and spectral (Table 4-VI) accuracy gained by using fifteen thousand calibration

samples is very small. It seems hard to justify the adoption of this algorithm when the additional time needed to characterize changes in ink or media is so considerable.

Table 4-V - Colorimetric and metatmeric accuracy of 4^6 midpoint estimated with YNSN model and Cellular-YNSN model with 4096 cells (15,625 Neugebauer primaries).

	ΔE_{00} (Ill D65, 2-Deg)				MI_{00} (Ill A, 2-Deg)				Spectral RMS
Model	**Mean**	**StdDev**	**Max**	**Min**	**Mean**	**StdDev**	**Max**	**Min**	
YNSN	0.97	0.43	3.22	0.044	0.24	0.20	1.51	0.001	0.0055
C-YNSN	0.78	0.38	2.66	0.055	0.22	0.20	1.42	0.001	0.0044

Table 4-VI - RMS Spectal error between model estimations and measured reflectance spectra.

Model	Primaries	Mean RMS	Max RMS	StdDev RMS
YNSN	64	4.60e-03	2.32e-02	2.94e-03
Cellular YNSN	729	4.56e-03	1.91e-02	2.78e-03
Cellular YNSN	15625	3.74e-03	1.64e-02	2.35e-03

Conclusions

For this system, the YNSN model is highly accurate and requires a small number of calibration measurements. The computational complexity and additional calibration measurements of the cellular model do not appear to be justified by the small increase in accuracy they bring. A small region of systematic error remains in the model and seems to be highly correlated to a single overprint combination. Further investigation into this area may be warranted. To improve the computational efficiency of the model it may be possible to combine together the dark primaries of high spectral similarity.

5. OPTIMIZING MODEL INVERSION

Introduction

The Yule-Nielsen-Spectral-Neugebauer model (YNSN) is not directly invertible. Nonlinear optimization is one approach for reaching a solution iteratively. The goal of this section of research was to identify optimization algorithms that were appropriate for this specific problem and to select the one that was most efficient in accurately inverting the YNSN model.

Evaluation Targets

For the algorithm evaluation two sample targets were used. The first was a series of four panels, painted with patches of art restoration pigments and titanium white by a conservator at the National Gallery Art in Washington DC. The panels contained a total of 219 pigment samples. Many of these pigments' reflectance spectra fall outside the colorimetric and spectral gamut of the printing system. A second target was created in such a matter as to insure that all its samples fall completely within gamut. This was accomplished by printing the samples from the system itself. The printed sample dataset contained 384 samples of randomly selected digital counts, constrained in such a way as to ensure a broad distribution through the colorant space. A reproduction of the pigment targets is included below as Figure 5-1 and the printed target as Figure 5-2.

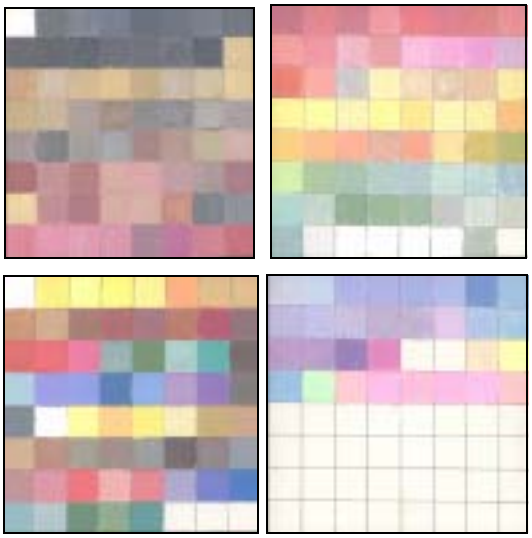


Figure 5-1 – 219-sample painted-pigment targets.

Figure 5-2 – 384-sample printed-patch target.

Objective Function

Nonlinear optimization deals well with multidimensional problems but requires a monotonic objective function. Testing of the algorithms was conducted with the objective function set to spectral RMS error between the measured and model predicted spectra as given in Equation (5.1).

$$RMS_{obj} = \sqrt{\frac{\sum_{\lambda=400,410...700} (R_\lambda - \hat{R}_\lambda)^2}{31}} \tag{5.1}$$

The selection of this objective function was done with the expectation that observations made will be relevant to other objective functions, for example color difference.

Optimization in MATLAB

All of the initial work done for this research was conducted in MATLAB. Use of the language is advantageous because of the ease in prototyping solutions, plotting results and the large number of prewritten routines. Initially the YNSN model was inverted in MATLAB using the "Optimization Toolbox" routine *fmincon*. The routine allows for the constrained minimization of nearly any function and was readily applied to the YNSN model. Based on the constraints provided and the scale of the problem the *fmincon* function selects an appropriate algorithm to use for the inversion. For YNSN with upper and lower bounding constraints on effective area coverage, a Quasi-Newton approach was selected. The Quasi-Newton algorithm observes the behavior of the model and its gradient (using a finite difference method) to build up curvature information and an approximation of the Hessian matrix.[74] Processing speed in MATLAB was unacceptably slow, therefore the problem was recoded in C.

Numerical Recipes in C

Many of the algorithms developed to perform this type of optimization have been compiled in the book "Numerical Recipes in C."[61] The book is accompanied by C source code implementations of the algorithms it discusses. Prewritten source code was particularity important as the complexity of the algorithms lies mostly in the subtly of

their implementation. For example, careful attention must be paid to the handling of round off error. Therefore, to save time and ensure proper implementation of the algorithms the code was used largely unmodified. The performance of the following algorithms was compared; full details of their history and inner workings can be found in the Numerical Recipes book:

- *AMOEBA* – Downhill simplex method. Requires only function evaluations.
- *POWELL* – Direction Set method, multi-dimensional search.
- *FRPR* – (Fletcher-Reeves-Polak-Ribiere) Conjugate gradient method.
- *DFP* – (Davidon-Fletcher-Powell) Variable-Metric/Quasi-Newton method.

Starting Value Sensitivity

The sensitivity of each algorithm to the starting point used was tested with several different approaches. First, fixed values near the corners of the colorant space were used. All inks were set at 10%, or 90% effective area coverage. Next single constant Kubelka-Munk (KM) was used to determine the starting points. This was done under the assumption that concentration is equivalent to area coverage. The equations for single-constant KM are given in Section 3. The concentrations were determined using a non-negative least squares (NNLS) algorithm and concentrations above unity were clipped.[75]

Results and Discussion

The five algorithms were used to match the spectra of two evaluation targets using several different starting points. The number of calls to the forward model needed to converge the inverse model was recorded. Additionally, RMS error and colorimetric statistics were gathered as measures of goodness for each model prediction of the sample

spectra. A summary of the results for the various algorithms is shown in Table 5-I with colorimetric accuracy shown in Table 5-II.

Downhill Simplex - AMOEBA

The first algorithm tested was the downhill simplex *AMOEBA*. The algorithm was started with the simplex vertices at the corners of the colorant space. Unfortunately, for the printed in-gamut samples the algorithm did not always converge to a solution. In a practical system these samples would have to be reprocessed using a different algorithm. Different starting points were tested but resulted in more samples not converging.

Single constant Kubelka Munk - KS

If the single constant Kubleka-Munk algorithm is used on its own it is clearly the quickest means of reaching a solution but the accuracy of the matches is low, both colorimetrically and spectrally. The poor accuracy is not surprising given that the physics of the system do not match those of the model. However, because of the models speed, it was used as a first-order approximation or starting point for the other, more accurate models. This was done with the expectation that if the other algorithms were started near the optimal solution they would converge to it faster.

Multidimensional Search - POWELL

The next algorithm tested was the multidimensional search algorithm, *POWELL*. The algorithm requires a starting point and a matrix of search directions. In all cases the search directions were set to the unit vectors. On the other hand the starting points were varied. In the first trials of the algorithm, the ink area coverages were set to 10% and the sample spectra were matched by minimizing spectral error. Next all the colorants were

set to 90% area coverage and the samples rematched. The average number of iterations and consequently the number of calls to the forward model increased slightly indicating a sensititity to starting value. Finally the starting points were set to the colorant coverages suggested by the non-negative least squares fit with the Kubelka-Munk model for each sample. These starting points led to faster convergence with less calls to the forward model and similar spectral and colorimetric accuracy.

Conjugate gradient method – FRPR

The conjugate gradient method, FRPR, outperformed the previous iterative algorithms in terms of processing speed. The increased performance was due to more efficient calls to the forward model through the use of the gradient function. As with *POWELL* using light starting points was better than using dark ones but the difference in performance was more significant in this case. When the single constant Kubelka-Munk model was used to provide starting points, the speed of the algorithm further increased.

Variable-Metric/Quasi-Newton method - DFP

The final algorithm tested, DFP, provided accurate results with the greatest speed (~150 pixels/sec). The actual algorithm used the BFGS (Broyden-Fletcher-Goldfarb-Shanno) variation of DFP which contains some subtle improvements to the way roundoff error and tolerances are handled. It was noted that DFP will not converge if started in very dark region because the numerically derived gradient gets stuck at zero. This was true for both the in-gamut printed samples and the pigment datasets. It is interesting to note that the metameric pigment samples were consistently processed faster even across the different

algorithms (Figure 5-3). This was probably due to a steep gradient in the optimization function near the optimal answers.

Table 5-I – Summary of algorithm performance and Spectral RMS error of solutions set.

				Spectral RMS			Function Calls			
Target	Algorithm	Start Value	(Pix/Sec)	Mean	Std Dev	Max	Mean	Std Dev	Max	No Converge
384 Random	AMOEBA	0/1	18.4	0.004	0.004	0.021	1492	1625	5014	65
Pigments	AMOEBA	0/1	55.0	0.040	0.038	0.194	505	259	1585	0
384 Random	DFP	All 0.1	83.9	0.005	0.010	0.184	325	113	758	0
Pigments	DFP	All 0.1	86.4	0.040	0.038	0.194	320	128	732	0
384 Random	DFP	All 0.9	30.0	0.028	0.066	0.682	945	581	1671	120
Pigments	DFP	All 0.9	42.5	0.055	0.077	0.648	671	464	1662	37
384 Random	DFP	KS	125.3	0.004	0.004	0.021	215	93	713	0
Pigments	DFP	KS	152.9	0.040	0.038	0.194	175	127	1061	0
384 Random	FRPR	All 0.1	31.4	0.004	0.004	0.021	850	447	3042	0
Pigments	FRPR	All 0.1	41.9	0.040	0.038	0.194	671	347	1651	0
384 Random	FRPR	All 0.9	27.0	0.004	0.004	0.021	1029	486	3478	0
Pigments	FRPR	All 0.9	35.5	0.040	0.038	0.194	789	377	2350	0
384 Random	FRPR	KS	47.2	0.004	0.004	0.021	581	395	2649	0
Pigments	FRPR	KS	73.1	0.041	0.037	0.194	378	337	1605	0
384 Random	KS	KS	4266.7	0.122	0.061	0.275	n/a	n/a	n/a	0
Pigments	KS	KS	5475.0	0.158	0.056	0.274	n/a	n/a	n/a	0
384 Random	POWELL	All 0.1	5.8	0.004	0.004	0.021	4759	2912	15558	0
Pigments	POWELL	All 0.1	9.8	0.040	0.038	0.194	2858	1914	9118	0
384 Random	POWELL	All 0.9	4.9	0.004	0.004	0.021	5615	3518	23616	0
Pigments	POWELL	All 0.9	8.0	0.040	0.038	0.194	3460	2504	13638	0
384 Random	POWELL	KS	6.4	0.004	0.004	0.021	4312	2637	14604	0
Pigments	POWELL	KS	10.3	0.040	0.038	0.194	2718	1651	8478	0

Table 5-II - Summary of algorithm performance and Colorimetric accuracy.

				ΔE_{00}			Function Calls			
Target	Algorithm	Start Value	(Pix/Sec)	Mean	Std Dev	Max	Mean	Std Dev	Max	No Converge
384 Random	AMOEBA	0/1	18.4	0.30	0.31	1.87	1492	1625	5014	65
Pigments	AMOEBA	0/1	55.0	2.57	2.80	13.47	505	259	1585	0
384 Random	DFP	All 0.1	83.9	0.39	1.17	21.62	325	113	758	0
Pigments	DFP	All 0.1	86.4	2.56	2.82	13.45	320	128	732	0
384 Random	DFP	All 0.9	30.0	3.88	8.14	41.75	945	581	1671	120
Pigments	DFP	All 0.9	42.5	4.16	6.52	42.11	671	464	1662	37
384 Random	DFP	KS	125.3	0.36	0.36	2.26	215	93	713	0
Pigments	DFP	KS	152.9	2.64	2.81	13.47	175	127	1061	0
384 Random	FRPR	All 0.1	31.4	0.34	0.30	1.86	850	447	3042	0
Pigments	FRPR	All 0.1	41.9	2.57	2.82	13.46	671	347	1651	0
384 Random	FRPR	All 0.9	27.0	0.35	0.35	2.49	1029	486	3478	0
Pigments	FRPR	All 0.9	35.5	2.59	2.81	13.45	789	377	2350	0
384 Random	FRPR	KS	47.2	0.42	0.45	4.95	581	395	2649	0
Pigments	FRPR	KS	73.1	2.68	2.79	13.47	378	337	1605	0
384 Random	KS	KS	4266.7	9.73	3.69	18.58	n/a	n/a	n/a	0
Pigments	KS	KS	5475.0	10.32	3.71	18.94	n/a	n/a	n/a	0
384 Random	POWELL	All 0.1	5.8	0.30	0.31	1.87	4759	2912	15558	0
Pigments	POWELL	All 0.1	9.8	2.57	2.81	13.46	2858	1914	9118	0
384 Random	POWELL	All 0.9	4.9	0.31	0.31	1.87	5615	3518	23616	0
Pigments	POWELL	All 0.9	8.0	2.57	2.81	13.45	3460	2504	13638	0
384 Random	POWELL	KS	6.4	0.29	0.31	1.87	4312	2637	14604	0
Pigments	POWELL	KS	10.3	2.57	2.81	13.46	2718	1651	8478	0

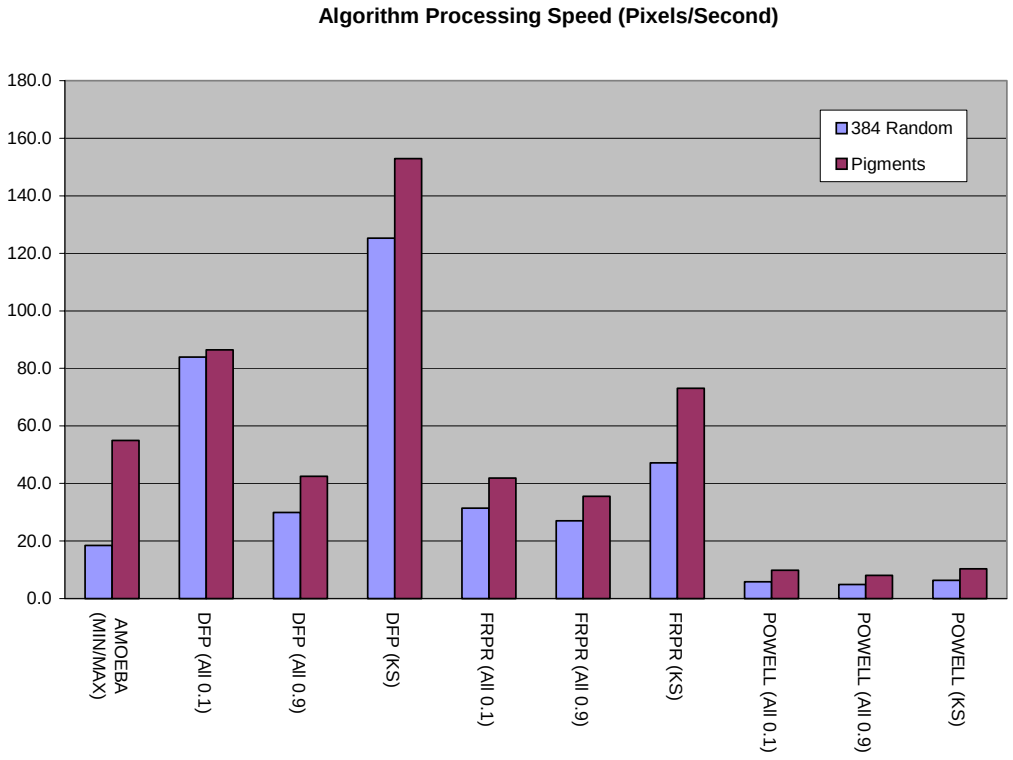


Figure 5-3 - Algorithm performance (pixels/second) for two sample targets and different algorithm starting points.

Conclusions

Algorithm performance with fixed starting points in the light corner of colorant space converged faster than the same algorithm with starting points in the dark corner of the colorant space. Using the single constant Kubelka-Munk and non-negative least squares estimation of the colorant amounts provided the best starting point for most of the models. All of the models exhibited similar spectral and colorimetric precision and accuracy once they converged to a solution. The DFP algorithm was selected as the best choice and will

be closely examined in the following section. Further improvements to the models speed might be obtained by rewriting the linear search routine in a manner specific to this system.

6. SIX-COLOR INVERSE MODEL EVALUATION

After selecting the Davidson-Fletcher-Powell (DFP) quasi-Newton algorithm, end-to-end system performance in inverting the Yule-Nielsen-Spectral-Neugebauer model was conducted. The measured spectra from five evaluation targets were matched using mixtures of the six printer inks and the mixtures were printed and measured spectrally. Forward model predictions of the measured output spectra were compared to the original spectra to assess the accuracy of the end-to-end spectral printing system.

Objective Functions

Three different objective functions were tested in conjunction with the DFP based inversion of the YNSN model.

Spectral RMS Error

The first objective function was RMS spectral error between the original and model predicted spectra, computed as shown in equation (6.1).

$$RMS_{obj} = \sqrt{\frac{\sum_{\lambda=400,410...700} (R_\lambda - \hat{R}_\lambda)^2}{31}} \tag{6.1}$$

where, R_λ is the original reflectance spectra and $\hat{R}_\lambda$ is the YNSN model prediction for the printed spectra based on the effective area coverages of the six inks. This objective function minimizes spectral error but does not necessarily lead to the best colorimetric match.

Spectral RMS Error with Spectral Weighting

The second objective function included a spectral weighting function to take into account the eyes spectral sensitivity and a reference illuminant under which the printed reproduction should match the original (D65). The weighting function was computed using the diagonal of Matrix-R computed using equation (6.2) presented by Fairman.[72]

$$\mathbf{R} = \mathbf{A}(\mathbf{A}'\mathbf{A})^{-1}\mathbf{A}' \qquad (6.2)$$

where, **A** is a (31x3) matrix of ASTM tristimulus weights for the reference illuminant and observer.

Multistage Objective Function

For the third objective function the optimizer converged to a final solution in two stages. First, the objective function was set to spectral RMS error and a solution minimizing spectral error was found. Next, through small adjustments (±%5) to each of the six ink effective area coverages from the first solution a colorimetric match was made by minimizing ΔE^*_{ab}.

Evaluation Targets

NGA Pigment Target

The NGA Pigment target is made up of 219 samples painted by a conservator at the National Gallery of Art in Washington, DC using art restoration pigments mixed with titanium white. The samples were arranged on four panels but compiled into a single target for printing and measurement.

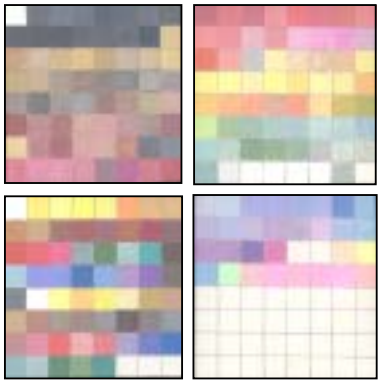


Figure 6-1 - 219 Painted pigment samples from National Gallery of Art.

GretagMacbeth ColorChecker

The ColorChecker contains 24 samples and was included in this evaluation so that the results could be compared with those obtained in Di-Yuan Tzeng's previous research.

Figure 6-2 - GretagMacbeth ColorChecker

GretagMacbeth ColorChecker DC

GretagMacbeth recently introduced a color target designed to aid in the characterization of digital imaging equipment. The new target contains a grid of 24x10 patches. The patches around the edge and center of the target are achromatic and meant to assess spatial uniformity in the imaging device. All 240 samples were included with the knowledge that the mean result is skewed by the distribution of these samples.

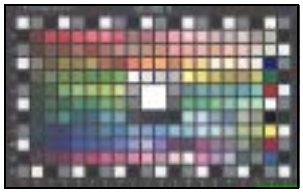


Figure 6-3 - GretagMacbeth ColorChecker DC target.

Vrhel Object Colors

The collection of 170 object colors collected by Vrhel, *et. al.*[76] was used as an indication of the results that could be expected from the printing system if real world objects were imaged and printed.

Random Printed Samples

The final target was made up of 384 random printed samples produced as output from the system itself. Production of the samples in this manner ensured that they fall completely within the spectral and colorimetric gamut of the printer. The samples were distributed through out the colorant space.

Figure 6-4 - Target of 384 Random printed samples.

Results and Discussion

Each of the three objective functions was used within the nonlinear optimization to match the spectra of the five targets. Colorimetric, metameric and spectral comparisons between the original target spectra and the forward model predicted spectra were compiled into Table 6-I. The fifteen sets of optimized area coverages were printed and measured, with the comparisons to the original spectra compiled in Table 6-II. Colorimetric performance was computed under D65 and the 1931 two-degree observer color matching functions. The Metameric index was computed by paramerically correcting[72] the spectra sets to match under illuminant D65 with the two-degree observer and then computing the color difference under CIE illuminant A.

Table 6-I - Colorimetric, metameric and spectral accuracy summary of original and predicted spectra produced through YNSN model inversion using three objective functions.

		Colorimetric - DE2000			Metameric – MI_{00}			Spectral RMS		
Objective	**Target**	**Mean**	**Std. Dev.**	**Max**	**Mean**	**Std. Dev.**	**Max**	**Mean**	**Std. Dev.**	**Max**
Spectral RMS	**Random**	0.58	0.95	9.89	0.15	0.26	3.52	0.004	0.004	0.036
	Object Spectra	3.38	3.20	18.44	0.68	0.84	6.04	0.027	0.028	0.261
	Pigment	2.45	2.68	14.03	0.72	0.72	3.37	0.039	0.037	0.192
	ColorChecker	2.50	1.89	7.60	0.59	0.55	1.89	0.026	0.018	0.063
	ColorCheckerDC	2.62	2.60	15.38	0.51	0.72	9.73	0.024	0.020	0.124
Overall Spectral RMS		1.95	2.52	18.44	0.45	0.66	9.73	0.020	0.026	0.261
Weighted Spectral RMS	**Random**	0.39	0.92	10.69	0.17	0.20	2.38	0.005	0.005	0.081
	Object Spectra	2.45	3.32	18.41	0.97	0.90	6.04	0.036	0.036	0.261
	Pigment	1.64	2.53	14.19	0.87	0.84	4.25	0.049	0.048	0.264
	ColorChecker	1.61	1.44	4.79	0.78	0.68	2.50	0.038	0.033	0.129
	ColorCheckerDC	1.56	2.34	15.38	0.71	0.87	9.73	0.032	0.032	0.225
Overall Weigthed RMS		1.29	2.31	18.41	0.59	0.77	9.73	0.026	0.035	0.264
Multistage (RMS→ΔE^*_{ab})	**Random**	0.13	0.54	6.05	0.19	0.31	4.19	0.005	0.004	0.036
	Object Spectra	1.87	3.37	18.45	0.72	0.83	6.04	0.031	0.030	0.261
	Pigment	0.88	2.08	10.83	0.73	0.73	3.69	0.044	0.042	0.203
	ColorChecker	0.96	1.55	4.96	0.60	0.49	1.89	0.033	0.026	0.102
	ColorCheckerDC	1.24	2.65	15.38	0.60	0.73	9.73	0.028	0.023	0.149
Overall Multistage		0.85	2.23	18.45	0.49	0.67	9.73	0.024	0.030	0.261

Table 6-II – Colorimetric, metameric and spectral accuracy summary of original and measured printed spectra produced through YNSN model inversion using three objective functions.

		Colorimetric - DE2000			Metameric – MI_{00}			Spectral RMS		
Objective	**Target**	**Mean**	**Std. Dev.**	**Max**	**Mean**	**Std. Dev.**	**Max**	**Mean**	**Std. Dev.**	**Max**
RMS	**Random**	2.10	1.00	8.23	0.36	0.38	3.29	0.018	0.009	0.047
	Object Spectra	3.60	3.04	18.44	0.78	0.80	6.04	0.033	0.027	0.260
	Pigment	3.17	2.30	13.77	1.05	0.82	3.74	0.048	0.031	0.186
	ColorChecker	2.85	1.39	6.01	0.78	0.52	1.85	0.033	0.022	0.102
	ColorCheckerDC	3.32	2.39	15.80	0.60	0.72	8.97	0.029	0.020	0.129
Overall RMS		2.87	2.17	18.44	0.64	0.70	8.97	0.030	0.024	0.260
Weighted RMS	**Random**	1.95	0.98	10.02	0.39	0.37	2.20	0.018	0.009	0.061
	Vrhel	3.35	2.97	18.69	1.12	0.99	6.56	0.041	0.037	0.261
	Pigment	2.59	2.29	13.78	1.06	0.90	3.69	0.052	0.043	0.248
	ColorChecker	2.34	1.67	7.64	0.99	1.08	4.69	0.045	0.042	0.193
	ColorCheckerDC	2.28	2.29	15.71	0.86	1.10	8.91	0.038	0.036	0.280
Overall Weighted RMS		2.40	2.10	18.69	0.78	0.88	8.91	0.034	0.034	0.280
Multistage (RMS→ΔE^*_{ab})	**Random**	1.96	0.83	6.53	0.39	0.42	3.71	0.018	0.009	0.049
	Object Spectra	3.01	2.77	18.64	0.89	0.82	6.06	0.038	0.031	0.261
	Pigment	2.30	1.58	9.99	1.04	0.83	3.92	0.051	0.035	0.194
	ColorChecker	1.87	1.36	5.64	0.77	0.75	3.41	0.039	0.034	0.154
	ColorCheckerDC	2.48	2.24	16.09	0.76	0.77	8.63	0.035	0.025	0.196
Overall Multistage		2.32	1.83	18.64	0.70	0.73	8.63	0.033	0.028	0.261

Looking at the system's overall predicted performance, the spectral RMS objective function is the worst performer. However when the actual printed spectra are compared to the originals we see that the mean color differences are all about the same. Small improvements in the standard deviation and maximum colorimetric error are seen by using the weighted spectral RMS error function or multistage objective function. The maximum error of the predicted matches for the in-gamut, random-printed-samples, indicates that a small number of samples are not converging to a solution of minimal error, even when one exists. The samples from all the targets except the random in-gamut target were pooled together and the mean spectral error between the original and printed spectra calculated at each wavelength. The resulting plot is shown in Figure 6-5. It would be interesting to see if the indicated peaks hold some physical correspondence with an

optimized ink set or are purely an artifact of the current ink set and printer model. The graphs upswept tail at the long wavelength end does coincide with the location where several of the pigments spectra increase but the printed mixtures can not.

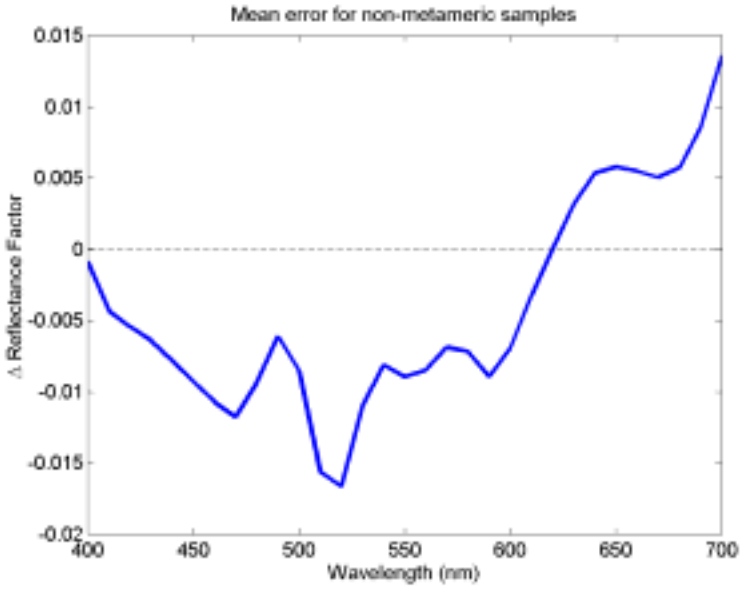


Figure 6-5 - Mean spectral reflectance error by wavelength for pooled dataset from ColorChecker, ColorChecker DC, Vrhel, and Pigment targets.

Comparison with Di-Yuan Tzeng's results

Tzeng made use of ten four color models in his final six-color end-to-end reproduction system. He also used a different printer, the DuPont Waterproof system. Comparison of his results with those of the current research was carried out for the GretagMacbeth ColorChecker target and each of the three objective functions. Tzeng used ΔE^*_{94} and the D50 illuminant so those will be used here too rather than ΔE_{00} and D65. The results are summarized in Table 6-III. For this target, the overall performance of Tzeng's Dupont

Waterproof based system is nearly identical to the Epson Inkjet based system when the multistage objective function is used. The magenta sample provides an interesting look at the advantage of using the multistage objective function. When spectral RMS error is used as the objective minimization function colorimetric error suffers and a match of

Table 6-III - End-to-End performance comparison of Tzeng's ten four-Color YNSN model using the DuPont Waterproof system with CMYKOG inks with the modified Epson 1200 from this research using the full six color YNSN model and three different optimization objective functions. Metameric colorimetric and spectral comparisons are between original spectra and measurements of printed reproductions.

	Tzang's System			Epson 1200 w/RMS			Epson 1200 w/weighted RMS			Epson 1200 w/Multistage		
Color Name	ΔE^*_{94}	MI_{94}	RMS	ΔE^*_{94}	MI_{94}	RMS	ΔE^*_{94}	MI_{94}	RMS	ΔE^*_{94}	MI_{94}	RMS
Dark Skin	1.64	0.51	0.013	2.51	0.68	0.016	1.72	0.11	0.020	1.44	0.16	0.021
Light Skin	1.57	0.14	0.038	1.65	0.52	0.033	1.67	0.70	0.042	2.05	0.74	0.045
Blue Sky	2.09	0.68	0.038	2.43	1.04	0.027	1.42	1.63	0.039	0.77	1.23	0.030
Foliage	1.70	0.49	0.016	3.52	0.07	0.023	2.99	0.17	0.024	1.16	0.16	0.025
Blue Flower	2.21	0.20	0.056	3.10	0.75	0.058	2.37	0.12	0.075	2.36	0.31	0.075
Blue Green	1.26	0.21	0.041	1.57	0.20	0.022	1.70	0.16	0.024	1.67	0.13	0.022
Orange	1.76	0.33	0.019	4.14	0.44	0.048	4.84	0.57	0.060	5.50	0.67	0.072
Purplish Blue	2.53	1.51	0.039	0.41	0.35	0.011	6.23	0.82	0.042	0.75	0.36	0.013
Moderate Red	1.54	0.12	0.028	0.76	0.22	0.013	0.44	0.02	0.011	0.56	0.20	0.012
Purple	1.87	1.08	0.083	5.79	1.19	0.040	3.42	1.17	0.068	3.20	0.51	0.046
Yellow Green	1.50	0.52	0.022	2.70	0.49	0.037	2.02	0.37	0.030	2.03	0.41	0.030
Orange Yellow	2.81	1.03	0.046	0.87	0.04	0.026	0.82	0.03	0.028	0.77	0.07	0.028
Blue	5.80	1.39	0.028	4.40	0.97	0.023	3.40	0.97	0.021	4.35	1.02	0.022
Green	1.05	0.29	0.020	3.89	0.28	0.027	1.72	0.13	0.018	2.60	0.20	0.020
Red	0.91	0.56	0.078	5.95	1.36	0.102	8.17	3.32	0.193	5.58	2.49	0.154
Yellow	1.52	0.32	0.028	2.40	0.29	0.042	2.74	0.18	0.037	1.91	0.21	0.043
Magenta	0.96	0.80	0.123	4.07	0.32	0.062	3.20	1.05	0.132	0.85	0.74	0.106
Cyan	1.55	1.74	0.034	2.30	0.23	0.017	2.57	0.32	0.018	1.26	0.42	0.014
White	1.45	0.10	0.059	2.93	0.20	0.059	0.28	0.07	0.065	0.23	0.05	0.064
N8	1.44	0.16	0.054	2.37	0.46	0.036	1.07	0.68	0.042	0.89	0.62	0.040
N6.5	2.89	0.25	0.050	2.22	0.81	0.033	2.12	0.97	0.041	1.64	0.71	0.031
N5	3.08	0.48	0.033	2.20	0.65	0.020	2.97	1.23	0.037	1.73	0.62	0.021
N3.5	2.00	0.11	0.014	1.66	0.15	0.007	0.89	0.54	0.012	0.92	0.06	0.009
Black	1.55	0.63	0.005	2.76	0.35	0.005	1.30	0.04	0.004	2.08	0.12	0.004
Mean	1.94	0.58	0.040	2.78	0.50	0.033	2.50	0.64	0.045	1.93	0.51	0.039
Stdev	1.01	0.46	0.026	1.41	0.36	0.022	1.82	0.73	0.042	1.44	0.53	0.034
Max	5.80	1.74	0.123	5.95	1.36	0.102	8.17	3.32	0.193	5.58	2.49	0.154
Min	0.91	0.11	0.005	0.41	0.04	0.005	0.28	0.02	0.004	0.23	0.05	0.004

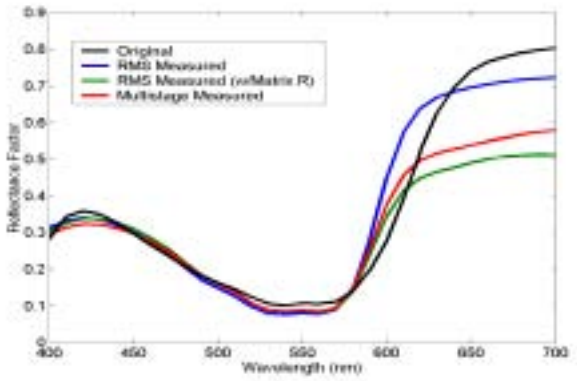


Figure 6-6 - Original and printed reproductions of magenta ColorChecker sample reflectance spectra using three different objective functions.

4.07 ΔE^*_{94} under D50 for the 2-degree observer is obtained. Switching to the weighted RMS error doesn't improve the match significantly. However, this may have been caused by the construction of Matrix-R for illuminant D65 instead of D50. The multistage objective function achieves a colorimetric match of 0.85 ΔE^*_{94}. This highly accurate match was obtained despite the fact that the second stage of the objective function was minimizing ΔE^*_{ab} for D65. Ideally the optimization should be carried out again using the D50 illuminant in the objective function.

Conclusions

Inversion of the YNSN model through the DFP algorithm using single-constant Kubelka-Munk starting points and a multistage objective function provided good end-to-end colorimetric and spectral accuracy for five sample targets. The colorimetric, metameric and spectral results for the GretagMacbeth ColorChecker target compare well with those from Tzeng's DuPont Waterproof based system. The advantage of using the full six-color YNSN model instead of ten four color models was immediately obvious. However this feature of the current system may become more important if an optimized ink set is

developed and the complex interactions between more than four inks at a time can be exploited. Additionally, when dealing with pictorial images rather than large solid patches the ability to model mixtures of more than six inks may become significant when the interaction between adjacent pixels is considered.

7. METAMERIC POTENTIAL

Introduction

The relationship between the number of inks used in the printing system and metameric index of the matches was investigated through a simple computational experiment. A single target of 219 painted-pigment samples mixed with titanium white was used for the evaluation. The target is described in Chapter 6. Three ink sets made up of CMY, CMYK, and CMYKOG inks were evaluated. The Yule-Nielsen-spectral-Neugebauer model was used to match the pigment spectra with a predicted mixture of the inks; it was inverted using the DFP algorithm discussed in the previous sections. The multistage objective function in which a spectral match based on minimizing RMS error, followed by colorimetric matching was used during the inversion. In the colorimetric stage, adjustments of plus and minus five percent effective area coverage were allowed to improve colorimetric accuracy. Eleven of the pigments were highly fluorescent and the evaluation was conducted twice, once with and once without these samples.

Results and Discussion

Colorimetric, metameric and spectral comparisons of the three, four and six ink matches were conducted. The results for the entire set of 219 pigments are shown in Table 7-I. The metameric index represents the colorimetric difference between the original and predicted spectra under illuminant A after a parameric correction to match the samples under illuminant D65. As the number of inks was increased, the metameric index was seen to decrease. The spectral RMS error was also reduced as the number of inks increased. However the mean colorimetric error increased slightly when the black ink

was added to the cyan, magenta, and yellow ink set. As shown in Table 7-II this increase almost disappears when the highly fluorescent samples (Figure 7-1) are removed, revealing a cause for this behavior. The small increase that remained may be due to round-off error and the fact that a different colorimetric metric, ΔE^*_{ab} was used in the model inversion objective.

Table 7-I – Colorimetric, metameric and spectral comparison of three, four and six ink matches.

	Colorimetric - DE2000			Metameric – MI_{00}			Spectral RMS		
Inks	**Mean**	**Std. Dev.**	**Max**	**Mean**	**Std. Dev.**	**Max**	**Mean**	**Std. Dev.**	**Max**
CMY	0.95	1.96	9.74	2.16	1.45	5.42	0.068	0.049	0.290
CMYK	1.02	2.16	14.57	1.43	0.83	4.63	0.061	0.053	0.291
CMYKOG	0.52	1.38	7.62	0.79	0.86	5.17	0.048	0.051	0.297

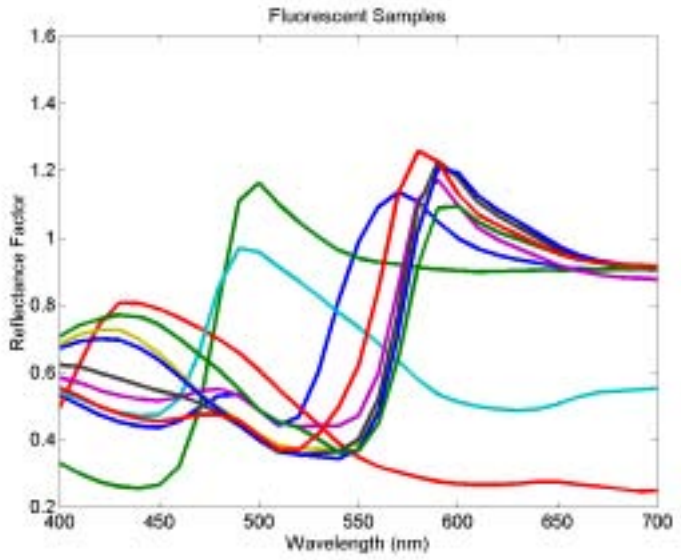


Figure 7-1 - Reflectance spectra of highly fluorescent pigments.

Table 7-II - Colorimetric, metameric and spectral comparison of three, four and six ink matches without highly fluorescent samples.

	Colorimetric - DE2000			Metameric – MI_{00}			Spectral RMS		
Inks	**Mean**	**Std. Dev.**	**Max**	**Mean**	**Std. Dev.**	**Max**	**Mean**	**Std. Dev.**	**Max**
CMY	0.65	1.35	6.44	2.18	1.45	5.42	0.062	0.039	0.258
CMYK	0.67	1.37	6.44	1.41	0.79	3.74	0.055	0.043	0.258
CMYKOG	0.31	0.87	5.56	0.76	0.81	4.07	0.043	0.043	0.253

For the several of the pigments, colorimetric matches were possible with all three ink sets. The multistage objective function was successful in utilizing the additional inks to reduce metamerism in these cases. For example, looking at the manganese gray pigment sample and the predictions of the three matches (Figure 7-2) we see that all the matches are metameric. However the match from the six ink match is less susceptible to illuminant metamerism as shown in Table 7-III.

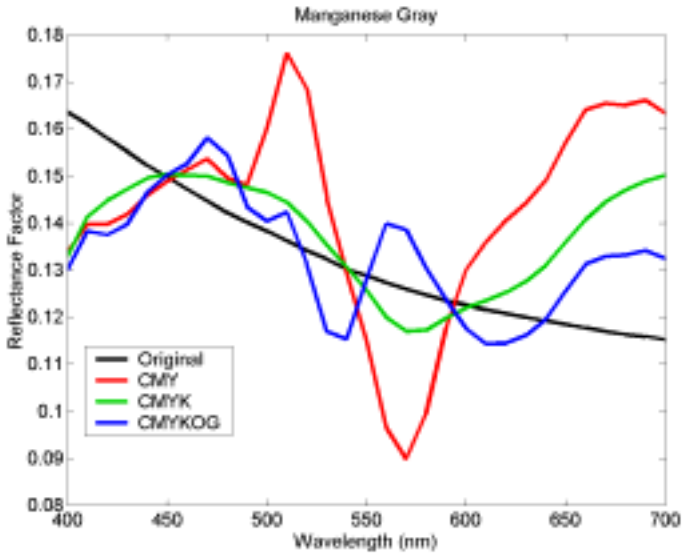


Figure 7-2 - Reflectance spectra for metameric Manganese Gray sample predictions using three, four and six inks.

Table 7-III - Colorimetric and metameric qualities of matches for three inksets.

Inkset	ΔE_{00} (2°,D65)	MI_{00} (2°,D65➔2°,A)
CMY	0.02	5.22
CMYK	0.01	1.77
CMYKOG	0.05	0.23

Plotting the twenty-eight pigments where the CMY and CMYK matches had metameric indexes that were nearly identical but both greater than the CMYKOG match, it is seen

that the advantage of adding orange and green inks correlates well with the orange and green reflectance spectra (Figure 7-3).

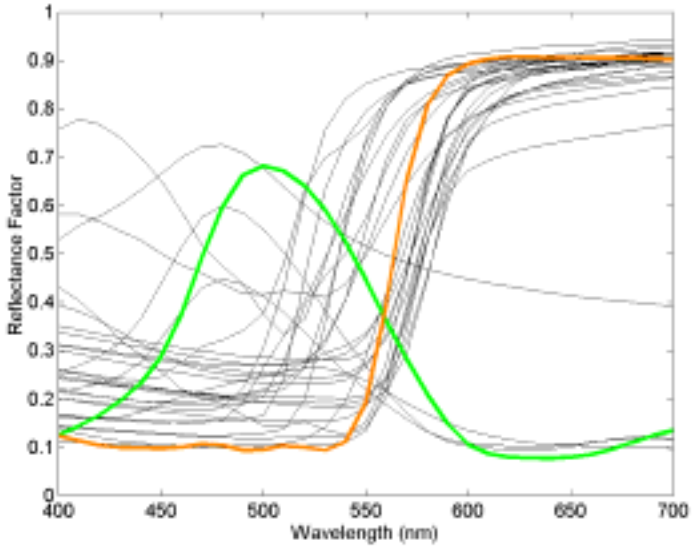


Figure 7-3 - Reflectance spectra of 28 pigments where MI_{00} for CMY ≈ CMYK and > CMYKOG.

Conclusions

As expected, the metameric index was reduced by using additional inks. The significant maximum colorimetric error from the six ink matches indicates that for the reproduction of these pigments the ink set is not optimal. Selection of more optimized inks could be carried out using the framework of this experiment if the characterization data required by the Neugebauer model could be synthesized or measured for a database of available inks. This process was described by Tzeng.[23,27,37,43] The multistage objective function was able to successfully reduce metamerism without decreasing colorimetric accuracy. The second stage of the objective function, where colorimetric matching is conducted, could be further enhanced by adding color appearance matching or gamut mapping.

8. CONCLUSIONS

Each phase of this research thesis provided valuable experience both in general printer modeling and the specific task of creating a spectral printer model for the custom six-color inkjet printer.

Two-Color Model Evaluation

The two-color model evaluation led to the identification of the Yule-Nielsen-spectral Neugebauer model (YNSN) as providing sufficient accuracy in predicting printed spectra from input digital counts. While the cellular YNSN model provided only slightly higher accuracy, the increased number of primaries needed to characterize the printer made its use less desirable. Unfortunately, the easily invertible continuous tone approximation of the printing system through single-constant Kubelka-Munk was not sufficiently accurate. However, the effort to implement it was not lost as during a later research phase as it provided good starting points that improved speed of the YNSN based model.

Testing of the Six-Color Forward Model

By coarsely sampling the entire colorant space with more than fifteen thousand samples, the performance of the YNSN model and Cellular-YNSN model was evaluated. Overall accuracy of both models was excellent. The mean colorimetric and spectral error of the forward model predictions was slightly lower for a second set of points located away from the outer boundaries of the colorant space. A possible cause for this is the accuracy of the Demichel weightings of the Neugebauer primary area coverages. If dot placement is not completely random then the weightings could be wrong. Physical inspection of the samples revealed that the small ink drops do not fully cover the paper even at the highest

digital counts which might also impact dot placement and the accuracy of the models. These observations are consistent with the higher error near the colorant space boundaries where individual primary weights are at their highest levels and most able to cause error. Further investigation into the validity of the Demichel weightings would probably lead to a more accurate forward model. Also, the small systematic error associated with the magenta-yellow-orange Neugebauer primary should receive further attention. In the future, the variable dot-size feature of the printer may be used; this will lead to higher coverage and smoother gradations but also a more complex model.

Optimizing Model Inversion

The task of selecting and implementing a optimization routine to invert the YNSN forward model proved to be the most computationally difficult part of the research. The final selection of the Quasi-Newton Davidson-Fletcher-Powell (DFP) algorithm was made solely on its performance compared with the other tested models. This means that better algorithms might exist and that subtle changes to the implementation of the algorithm to tailor it to this problem might also lead to improved performance. Use of the single-constant Kubelka-Munk model to provide starting points for the optimization routine was found to significantly improve performance and eliminate cases where a convergent solution could not be reached.

Six-Color Inverse Model Evaluation

Testing of the optimization driven inverse model was carried out with the goal of identifying a suitable objective function and measuring end-to-end accuracy of the printing system. A multistage objective function based on both spectral and colorimetric

error was found to give the most accurate results. Spectra measured from a printed reproduction of the GretagMacbeth ColorChecker were found to be as accurate as in the previous research conducted by Tzeng using the DuPont WaterProof system and a spectral printer model based on ten separate four-color YNSN models. An advantage of the current system is that it allows for simultaneous placement of all six inks within a halftone cell. This feature may become significant with the processing of pictorial images or with optimized ink sets.

Metameric Potential

Using CMY, CMYK and CMYKOG subsets of the printer's inks the potential reduction in metamerism when matching 219 artists' pigments was explored. Matches made with the multistage objective function were able to use added inks to minimize metamerism. Since the analysis was carried out theoretically, the additional error between the forward printer model and actually printing the samples was not imposed on the results; if it had it may have reduced the significance of the results but the fact that it worked theoretically shows that an optimized six-color ink set, where the potential is larger, could be well utilized to reduce metamerism.

Closing Thoughts on Future Research

The current system is computationally fast enough to process entire images but at hours per image, only in a research setting. Since the printing system was successfully modeled spectrally the model can be applied to many other real world problems. For example building lookup tables for specific applications of the printer could be conducted using the model rather than having to print and measure all the samples that make up the

lookup table. Intuitively, research into the areas of ink selection, spectral gamut mapping, minimizing starting value sensitivity and improving the overall speed of the algorithm seem worthwhile. Conducting this research was an amazingly rewarding experience. Hopefully, the knowledge gained can be applied to future research efforts in the field.

REFERENCES

1. R. S. Berns, Spectral modeling of a dye diffusion thermal transfer printer, *J. Electronic Imaging* **2**, 359-370 (1993).
2. T. Kohler and R. S. Berns, Reducing metamerism and increasing gamut using five or more colored inks, Proc. of IS&T Third Technical Symposium on Prepress, Proofing and Printing, 24-28 (1993).
3. D. S. S. Vent, Multichannel analysis of object-color spectra, Master Degree Thesis, R.I.T., Rochester, NY, 1994.
4. J. Arney, P. Engeldrum and H. Zeng, An expanded Murray-Davies model of tone reproduction in halftone imaging, *J. Imaging Sci. Technol*. **39**, 502-508 (1995).
5. R. Berns, M. Shyu, Colorimetric characterization of a desktop drum scanner using a spectral model, *J. Electronic Imaging* **4**, 360-372 (1995).
6. R. S. Berns and K. Iino, Spectral Modeling of an ink jet printer, in *Electronic Imaging SPIE/IS&T's International Technical Working Group Newsletter Special Issue on Color* **5**, 1995, p. 3.
7. S. Arney, C. D. Arney and P. G. Engeldrum, Modeling the Yule-Nielsen effect, *J. Imag. Sci. and Technol.* **40**, 233-238 (1996).
8. P. D. Burns and R. S. Berns, Analysis of multispectral image capture, in *Proc. of the IS&T/SID Fourth Color Imaging Conference Color Science, Systems, and Applications*, IS&T, Springfield, VA, 1996, pp. 19-22.
9. P. D. Burns, Analysis of image noise in multi-spectral color acquisition, Ph.D. Thesis, R. I. T., Rochester, NY, 1997.
10. J. S. Arney and M. Katsube, A probability description of the Yule-Nielsen effect II: The impact of halftone geometry, *J. Imag. Sci. and Technol.* **41**, 633-636 (1997).
11. J. S. Arney, A probability description of the Yule-Nielsen effect, *J. Imag. Sci. and Technol.* **41**, 637-642 (1997).
12. P. Burns, R. Berns, Error propagation in color signal transformations, *Color Res. Appl.* **22**, 280-289 (1997).
13. J. S. Arney, T. Wu and C. Blehm, Modeling the Yule-Nielsen effect on color halftones, in *Proc. of the IS&T/SID Fifth Color Imaging Conference Color Science, Systems, and Applications*, IS&T, Springfield, VA, 1997, pp. 62-65.
14. K. Iino, R. Berns, A spectral based model of color printing that compensates for optical interactions of multiple inks, in *Proc. of AIC Color 97*, 1997, pp. 610-613.
15. P. Burns, R. Berns, Modeling colorimetric error in electronic image acquisition, *Proc. OSA Annual Meeting*, 147-149 (1997).
16. G. Johnson, Computer synthesis of spectroradiometric images for color imaging systems analysis, *MS Thesis, RIT* (1998).
17. K. Iino and R. S. Berns, Building color management modules using linear optimization I. Desktop color system, *J. Imaging Sci. Tech.* **42**, 79-94 (1998).
18. K. Iino and R. S. Berns, Building color management modules using linear optimization II. Prepress system for offset printing, *J. Imaging Sci. Tech.* **42**, 99-144 (1998).
19. J. S. Arney, T. Wu and C. Blehm, Modeling the Yule-Nielsen effect on color halftones, *J. Imag. Sci. and Technol.* **42**, 335-340 (1998).
20. R. S. Berns, Challenges for color science in multimedia imaging, in L. MacDonald and M. R. Luo, Ed., Colour Imaging: Vision and Technology, John Wiley & Sons, Chichester, 1998, pp. 99-127.

21. R. S. Berns, F. H. Imai, P. D. Burns and D. Tzeng, Multispectral-based color reproduction research at the Munsell Color Science Laboratory, in *Proc. of SPIE* Vol. **3409**, Jan Bares, Editor, SPIE, Bellingham, WA, 1998, pp. 14-25.

22. F. H. Imai and R. S. Berns, High-resolution multi-spectral image capture for fine arts preservation, in *Proc. of Fourth Argentina Color Conference*, 1998, pp. 21-22.

23. D. Tzeng and R. S. Berns, Spectral-based ink selection for multiple-ink printing I. Colorant estimation of original objects, in *Proc. of the IS&T/SID Sixth Color Imaging Conference Color Science, Systems, and Applications*, IS&T, Springfield, VA, 1998, pp. 106-111.

24. G. M. Johnson and M. D. Fairchild, Computer synthesis of spectroradiometric images for color imaging system analysis, in *Proc. IS&T/SID Sixth Color Imaging Conference: Color Science, Systems and Applications*, IS&T, Springfield, VA, 1998, pp. 150-153.

25. F. H. Imai and R. S. Berns, High-resolution multi-spectral image archives: a hybrid approach, in *Proc. of the IS&T/SID Sixth Color Imaging Conference Color Science, Systems, and Applications*, IS&T, Springfield, VA, 1998, pp. 224-227.

26. F. Imai, Multi-spectral Image Acquisition and Spectral Reconstruction using a Trichromatic Digital Camera System Associated with Absorption Filters, *MCSL Technical Report*, http://www.cis.rit.edu/research/mcsl/pubs/PDFs/CameraReport.pdf (1998).

27. D. Tzeng, Spectral-based color separation algorithm development for multiple-ink color reproduction, Ph.D. Thesis, R. I.T., Rochester, NY, 1999.

28. J. S. Arney, E. Pray, and K. Ito, Kubelka-Munk theory and the Yule-Nielsen effect on halftones, *J. Imag. Sci. and Technol.* **43**, 353-358 (1999).

29. G. M. Johnson and M. D. Fairchild, Full-spectral color calculations in realistic image synthesis, *IEEE Computer Graphics & Applications* **19**, 47-33 (1999).

30. R. S. Berns, Challenges for colour science in multimedia imaging systems, in L. MacDonald and R. Luo, Editors, *Colour imaging: Vision and technology*, John Wiley & Sons, England, 99-127, 1999.

31. D. Tzeng and R. S. Berns, Spectral reflectance prediction of ink overprints by Kubelka-Munk turbid media theory, in *Proc. of TAGA/ISCC Symposium in Vancouver B.C.*, 1999, pp. 682-697.

32. A. Ito, N. Ohta, E. Kanagawa, Optimization of Probability model for color halftone by macroscopic spectral reflectance, in *Proc. of IS&T's 1999 PICS Conference*, 1999, pp. 386-389.

33. F. H. Imai and R. S. Berns, Spectral estimation using trichromatic digital cameras, in *Proc. of the International Symposium on Multispectral Imaging and Color Reproduction for Digital Archives*, Chiba University, Chiba, Japan, 1999, pp. 42-49.

34. M. R. Rosen and X. Jiang, Lippmann 2000: A spectral image database under construction, in *Proc. of the International Symposium on Multispectral Imaging and Color Reproduction for Digital Archives*, Chiba University, Chiba, Japan, 1999, pp. 117-122.

35. F. H. Imai and R. S. Berns, A comparative analysis of spectral reflectance reconstruction in various spaces using a trichromatic camera system, in *Proc. of IS&T/SID Seventh Color Imaging Conference: Color Science, Systems, and Applications*, IS&T, Springfield, 1999, pp. 21-25.

36. P. D. Burns and R. S. Berns, Quantization in multispectral color image acquisition, in *Proc. of IS&T/SID Seventh Color Imaging Conference: Color Science, Systems, and Applications*, IS&T, Springfield, 1999, pp. 32-35.

37. Tzeng and R. S. Berns, Spectral-based ink selection for multiple-ink printing II. Optimal ink selection, in *Proc. of the IS&T/SID Seventh Color Imaging Conference Color Science, Systems, and Applications*, IS&T, Springfield, VA, 1999, pp. 182-187.

38. D. R. Wyble, R. S. Berns, A Critical Review of Spectral Models Applied to Binary Color Printing, *Color Res. Appl.* **25**, 4-15 (2000).

39. R. S. Berns, Billmeyer and Saltzman's Principles of Color Technology, 3rd Edition, John Wiley & Sons (2000).

40. F. H. Imai, R. S. Berns and D. Tzeng, A comparative analysis of spectral reflectance estimation in various spaces using a trichromatic camera system, *J. Imaging Sci. Technol*. **44**, 280-287 (2000).

41. S. Gonzalez, M. Fairchild, Evaluation of Bispectral Spectrophotometry for Accurate Colorimetry of Printing Materials, in *Proc. of Eighth Color Imaging Conference: Color Science and Engineering, Systems, Technologies and Applications*, IS&T, Springfield, 2000, pp. 39-43.

42. M. R. Rosen, M. D. Fairchild, G. M. Johnson and D. R. Wyble, Color Management within a spectral image visualization tool, in *Proc. of Eighth Color Imaging Conference: Color Science and Engineering, Systems, Technologies and Applications*, IS&T, Springfield, 2000, pp.75-80.

43. F. H. Imai, M. R. Rosen and R. S. Berns, Comparison of spectrally narrow-band capture versus wide-band with *a priori* sample analysis for spectral reflectance estimation, in *Proc. of Eighth Color Imaging Conference: Color Science and Engineering, Systems, Technologies and Applications*, IS&T, Springfield, 2000, pp. 234-241.

44. D. Tzeng and R. S. Berns, Spectral-based six-color separation minimizing metamerism, in *Proc. of Eighth Color Imaging Conference: Color Science and Engineering, Systems, Technologies and Applications*, IS&T, Springfield, 2000, pp.342-247.

45. S. Quan, N. Ohta, Optimization of Camera Spectral Sensitivities, in *Proc. of Eighth Color Imaging Conference: Color Science and Engineering, Systems, Technologies and Applications*, IS&T, Springfield, 2000, pp.273-278.

46. F. H. Imai, M. R. Rosen, R. S. Berns, N. Ohta and N. Matsushiro, Preliminary study on spectral image compression, in *Proc. of Color Forum Japan 2000*, 2000, pp. 67-70.

47. R. S. Berns, The science of digitizing paintings for color-accurate image archives: A review, *J. Imaging Sci. Technol.* **45**. 373-383 (2001).

48. F. H. Imai, M. R. Rosen, D. Wyble, R. S. Berns and D. Tzeng, Spectral reproduction from scene to hardcopy I: Input and Output, in *IS&T/SPIE Conference on Color Imaging: Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications II*, M. M. Blouke, J. Canosa, N. Sampat, Editors, *Proc of SPIE* Vol. **4306**, 2001, pp. 346-357.

49. M. R. Rosen, F. H. Imai, X. Jiang and N. Ohta, Spectral reproduction from scene to hardcopy II: Image processing, in *Proc. of IS&T/SPIE Conference on Color Imaging: Device-Independent Color, Color Hardcopy, and Graphic Arts VI*, R. Eschbach, G. G. Marcu, Editors, *Proc. of SPIE* **4300**, 2001, pp. 33-41.

50. F. H. Imai, M. R. Rosen and R. S. Berns, Multi-spectral imaging of a van Gogh's self-portrait at the National Gallery of Art, Washington, D.C., in *Proc. of IS&T PICS Conference*, IS&T, Springfield, VA, 2001, pp. 185-189.

51. F. H. Imai, S. Quan, M. R. Rosen and R. S. Berns, Digital camera filter design for colorimetric and spectral accuracy, in *Proc. of Third International Conference on Multispectral Color Science*, Markku Hauta-Kasari, Jouni Hiltunen and Jarmo Vanhanen, Editors, University of Joensuu, Finland, 2001, pp. 13-16.

52. N. Matsushiro, F. H. Imai and N. Ohta, Principal component analysis of spectral images based on the independence of color matching function vectors, in *Proc. of Third International Conference on Multispectral Color Science*, Markku Hauta-Kasari, Jouni Hiltunen and Jarmo Vanhanen, Editors, University of Joensuu, Finland, 2001, pp. 77-80.

53. F. H. Imai and R. S. Berns, Spectral estimation of oil paints using multi-filter trichromatic imaging, in *Proc. of the 9th Congress of the International Colour Association*, Rochester, NY, 2001, in press.

54. R. S. Berns, J. Krueger, and M. Swicklik, 'Multiple pigment selection for inpainting using visible reflectance spectrophotometry,' *Studies in Conservation* (2001) in press.

55. M. Rosen, L. Taplin, F. Imai, R. Berns, N. Ohta, Answering Hunt's Web Shopping Challenge: Spectral Color Management for a *Virtual Swatch*, in *Proc. of Ninth Color Imaging Conference: Color Science and Engineering, Systems, Technologies and Applications*, IS&T, Springfield, 2001, in press.

56. R. S. Berns and F. H. Imai, Pigment identification of artist materials via multi-spectral imaging, in *Proc. of Ninth Color Imaging Conference: Color Science and Engineering, Systems, Technologies and Applications*, IS&T, Springfield, 2001, in press.

57. Q. Sun and M. D. Fairchild, Statistical characterization of spectral reflectances in spectral imaging of human portraiture, in *Proc. of Nineth Color Imaging Conference: Color Science and Engineering, Systems, Technologies and Applications*, IS&T, Springfield, 2001, pp. 73-79.

58. R. S. Berns and F. H. Imai, Pigment identification of artist materials via multi-spectral imaging, in *Proc. of Nineth Color Imaging Conference: Color Science and Engineering, Systems, Technologies and Applications*, IS&T, Springfield, 2001, pp. 85-90.

59. L. A. Taplin and R. S. Berns, Spectral color reproduction based on a six-color inkjet output system, in *Proc. of Nineth Color Imaging Conference: Color Science and Engineering, Systems, Technologies and Applications*, IS&T, Springfield, 2001, pp. 209-213.

60. M. R. Rosen, L. A. Taplin, F. H. Imai, R. S. Berns and N. Ohta, Answering Hunt's web shopping challenge: spectral color management for a virtual swatch, in *Proc. of Nineth Color Imaging Conference: Color Science and Engineering, Systems, Technologies and Applications*, IS&T, Springfield, 2001, pp. 267-273.

61. W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery . Numerical Recipes in C: The Art of Scientific, 2nd Edition, Cambridge University Press (1992).

62. J. J. Sarton, http://www.swtools.automatix.de

63. F. R. Steinberg, An adaptive algorithm for spatial grey scale, *Proc. Soc. Info Display*, **17**, p75 (1976)

64. A. Murray, Monochrome reproduction in photoengraving, *J Franklin Inst.,* **221**, pp.721-744, (1936)

65. H. E. J. Neugebauer, Die theoretischen Grundlagen des Mehrfarbenbuchdrucks. In *Neugebauer Memorial Seminar on Color Reproduction* :14-15 December 1989, Tokyo, Japan (Sayanagi, K., ed.), pp. xv, 203, SPIE

66. M. E. Demichel, *Procédé 1924*;26:17-21,26-27.

67. J. A. C. Yule, Principles of Color Reproduction, John Wiley & Sons Inc., p.262 (1967)

68. J. A. S. Viggiano, The color of halftone tints. *TAGA Proc.***37**, pp. 647-661 (1985)

69. K. J. Heuberger, Z. M. Jing, S. Persiev, Color Transformations and lookup tables. *TAGA/ISCC Proc:1992*, pp. 863-881

70. R. Rolleston, R. Balasubramanian, Accuracy of various types of Neugebauer models. *Proc IS&T SID Col. Imag. Conf.* pp.32-37 (1993)

71. P. Kubelka, F. Munk, Ein Beitrag zur Optik der Farbanstriche, *Z. Tech Phys.***12**, pp.593-601 (1931).

72. H. Fairman, Metameric Correction Using Parameric Decomposition, *Color Res. Appl,* **12**, p261 (1987).

73. J. A. C. Yule, The penetration of light into paper and its effect on halftone reproductions. *TAGA Proc.***3**, pp.65-76 (1951).

74. T. Colemann, M. A. Branch, A. Grace, Optimization Toolbox User's Guide., The Math Works Inc, pp.2.11-2.16 (1999).

75. C. L Lawson, R. J. Hanson, Solving Least Squares Problems, Prentice-Hall,, Chapter 23, p.161 (1974).

76. M. H. Vrhel, *et al.,* Measurement and Analysis of Object Reflectance Spectra. *Color Res Appl. **19**, 4 (1994)*

APPENDIX B. MATLAB CODE

Much of the research source code was first written in MATLAB. The flowing appendix includes many of the functions used to implement and test the forward and inverse model.

CIE helper functions

A series of functions were written to calculate the values associated with the CIE colorspaces and color differences. *xyz*, computes tristimulus values from reflectance spectra and *lab* in turn computes CIELAB values. The function *illD*, returns the CIE daylight illuminant for any correlated color tempature. CIE color difference equations for ΔE^*_{ab}, ΔE^*_{94} and ΔE_{00} are provided. ΔE00 based index of metamerism (*meta_idx00*) is computed using the parameric correction in *pcorrect*.

---

xyz.m

```
function [XYZ, xy]=xyz(ref, cmf, ill)
% XYZ: returns the XYZ Tristimulus values for a refelectance sample

if nargin ~= 3
  fprintf('xyz called with wrong number of arguments.\n');
  return;
end

XYZ = (ill*ones(1,3).*cmf)'*ref.*100./sum(cmf(:,2).*ill);

if nargout == 2
  sumXYZ = XYZ * ones(1,3);
  xy = [XYZ(:,1)./sumXYZ, XYZ(:,2)./sumXYZ];
end
```

---

lab.m

```
function output = lab(XYZ1, XYZn)
% LAB: returns the CIE L* a* b* values of at a given whitepoint

t1=XYZ1./(XYZn*ones(1,size(XYZ1,2)));

t2=zeros(size(XYZ1));
t2(find(t1>0.008856)) = t1(find(t1>0.008856)).^(1/3);
t2(find(t1<=0.008856)) = t1(find(t1<=0.008856))*7.787+(16/116);

output = [116*t2(2,:)-16; 500*(t2(1,:)-t2(2,:)); 200*(t2(2,:)-t2(3,:))];
```

---

illD.m

```
function [ill]=illD(Tc)
% illD: returns the daylight illuminant for at the given color temperature

eigD = load('CIE_eigD.txt');

if Tc <= 7000
  xD = -4.6070*10^9/Tc^3+2.9678*10^6/Tc^2+0.09911*10^3/Tc+0.244063;
else
  xD = -2.0064*10^9/Tc^3+1.9018*10^6/Tc^2+0.24748*10^3/Tc+0.23704;
end
yD = -3.000*xD^2 + 2.870*xD - 0.275;
M1 = (-1.3515 - 1.7703*xD + 5.9114*yD)/(0.0241 + 0.2562*xD  - .7341*yD);
M2 = (0.0300 - 31.4424*xD + 30.0717*yD)/(0.0241 + 0.2562*xD  - .7341*yD);
ill = eigD(:,1) + M1.*eigD(:,2) + M2.*eigD(:,3);
```

deltaEab.m

```
function   DEab=deltaEab(Lab1, Lab2)
% deltaEab: Caculates the CIE Delta Eab Color Difference

DEab = sqrt(sum((Lab1-Lab2).^2,1));
```

deltaE94.m

```
function   De94=deltaE94(Lab1, Lab2)
% deltaE94: Caculates the CIE Delta E94 Color Difference

DLab = Lab1 - Lab2;
DEab = sqrt(sum(DLab.^2,1));
C1 = sqrt(Lab1(2,:).^2+Lab1(3,:).^2);
C2 = sqrt(Lab2(2,:).^2+Lab2(3,:).^2);
DC = C1-C2;
DH = sqrt(DEab.^2 - DLab(1,:).^2 - DC.^2);
SL = ones(1,size(Lab2, 2));
SC = 1 + 0.045.*C2;
SH = 1 + 0.015.*C2;
De94 = sqrt((DLab(1,:)./SL).^2 + (DC./SC).^2 + (DH./SH).^2);
```

deltaE00.m

```
function   De00=deltaE00(Lab1, Lab2)
% deltaE94: Calculates the CIE Delta E00 Color Difference

%CIELAB Chroma
C1 = sqrt(Lab1(2,:).^2+Lab1(3,:).^2);
C2 = sqrt(Lab2(2,:).^2+Lab2(3,:).^2);

%Lab Prime
mC = (C1+C2)./2;
G=0.5*(1-sqrt((mC.^7)./((mC.^7)+(25.^7))));
LabP1 = [Lab1(1,:) ; Lab1(2,:).*(1+G) ; Lab1(3,:)];
LabP2 = [Lab2(1,:) ; Lab2(2,:).*(1+G) ; Lab2(3,:)];

%Chroma
CP1 = sqrt(LabP1(2,:).^2+LabP1(3,:).^2);
CP2 = sqrt(LabP2(2,:).^2+LabP2(3,:).^2);
```

```
%Hue Angle
hP1t = atan2Deg(LabP1(3,:),LabP1(2,:));
hP2t = atan2Deg(LabP2(3,:),LabP2(2,:));

%Add in 360 to the smaller hue angle if absolute value of difference is > 180
hP1 = hP1t + ((hP1t<hP2t)&(abs(hP1t-hP2t)>180)).*360;
hP2 = hP2t + ((hP1t>hP2t)&(abs(hP1t-hP2t)>180)).*360;

%Delta Values
DLP = LabP1(1,:) - LabP2(1,:);
DCP = CP1 - CP2;
DhP = hP1 - hP2;
DHP = 2*(CP1.*CP2).^(1/2).*sinDeg(DhP./2);

%Arithmetic mean of LCh' values
mLP = (LabP1(1,:)+LabP2(1,:))./2;
mCP = (CP1+CP2)./2;
mhP = (hP1+hP2)./2;

%Weighting Functions
SL = 1+(0.015.*(mLP-50).^2)./sqrt(20+(mLP-50).^2);
SC = 1+0.045.*mCP;
T = 1-0.17.*cosDeg(mhP-30)+0.24.*cosDeg(2.*mhP)+ ...
    0.32.*cosDeg(3.*mhP+6)-0.2.*cosDeg(4.*mhP-63);
SH = 1+0.015.*mCP.*T;

%Rotation function
RC = 2.*sqrt((mCP.^7)./((mCP.^7)+25.^7));
DTheta = 30.*exp(-((mhP-275)./25).^2);
RT = -sinDeg(2.*DTheta).*RC;

%Parametric factors
kL = 1;
kC = 1;
kH = 1;

De00 = ((DLP./kL./SL).^2+(DCP./kC./SC).^2+(DHP./kH./SH).^2+ ...
    (RT.*(DCP./kC./SC).*(DHP./kH./SH))).^(1/2);

function out = cosDeg(in);
out = cos(in.*pi./180);

function out = sinDeg(in);
out = sin(in.*pi./180);

function out = atan2Deg(inY,inX);
out = atan2(inY,inX).*180./pi;
out = out+(out<0).*360;
```

---

meta_idx00.m

```
function meta_index=meta_idx00(standard, trial, R_paper, cmf, ill1, ill2)
%Compute the CIE DE2000 index of metamerism between a standard and trial set

corrected_spectra=pcorrect(standard, trial, ill1,cmf);

XYZn = xyz(R_paper, cmf,ill2);

LabStandard=lab(xyz(standard,cmf,ill2),XYZn);
LabCorrected=lab(xyz(corrected_spectra,cmf,ill2),XYZn);

meta_index=deltaE00(LabStandard, LabCorrected);
```

---

pcorrect.m

```
function corrected_spectra=pcorrect(standard, trial, light_source, cmf)
%performs parameric correction on trial spectra so it matches standard
%when viewed under specified lightsource by observer with given
%color matching functions

A = diag(light_source)*cmf;
R=A*inv(A'*A)*A';

[m,n]=size(R);

identity=diag(ones(m,1));

corrected_spectra=R*standard + (identity-R)*trial;
```

Printed Image Utilities

Two functions were written to aid in creating printed images from the system. The first *patch_image* takes an array of digital counts and converts it into a image made up of patches with those digital counts. The second, *epson_image*, takes an image array and creates a printer compatible binary file. This is done in several steps and involves invoking the both the external halftoning and printer driver routines.

---

patch_image.m

```
function output_image = patch_image(patches,labels);

%patch size parameters are hard coded
dpi = 720;
patch_height = 3/8 *dpi;
patch_width = 3/8 *dpi;
patch_spacing = 1/16 * dpi;

%setup space for row and column labels if requested
label_rows = 0;
label_columns = 0;
if nargin == 2
  label_rows = 1/4 * dpi;
  label_columns = 1/4 * dpi;
end

%get patch data dimensions
[rows, columns, inks] = size(patches);

%create output image
Image_Width  = label_columns + columns*patch_width+patch_spacing*(columns-1);
Image_Height = label_rows + rows*patch_height+patch_spacing*(rows-1);
output_image = uint8(zeros(Image_Height,Image_Width,6));

%populate patch image with patch data
for r = 1:rows % row loop
  for c = 1:columns % column loop
```

```
    end_row = label_rows + r*patch_height + (r-1)*patch_spacing;
    start_row = end_row-patch_height+1;
    end_column = label_columns + c*patch_width + (c-1)*patch_spacing;
    start_column = end_column-patch_width+1;
    for i = 1:inks % ink loop
      output_image(start_row:end_row, start_column:end_column,i) = ...
            ones(1+end_row-start_row,1+end_column-start_column)*patches(r,c,i);
    end % ink loop
  end % column loop
end % row loop

%add in the row and column labels if requested.
if nargin == 2;

end;
```

---

epson_image.m

```
function [] = epson_image(input_image, filename)

[H,W,inks] = size(input_image);

%pick tempfile name
tf = tempname;

Compressed_Image = zeros(W,H);

%dither each channel and add to compressed binary
for i=1:6
  fprintf('Dithering Channel %d ... ',i);
  imwrite(input_image(:,:,i)',tf,'tiff','Compression','none');
  eval(['!/usr/lib/print/ditherstiff -I' tf ' -O' tf 'dithered']);
  Compressed_Image = Compressed_Image+double(imread([tf 'dithered'],'tiff')).*2^(i-1);
  fprintf('Done \n');
end

%write binary file
fprintf('Writing Binary File ... ');
fwriteid = fopen(['/usr/people/lat3977/cis/epson1200/' filename '.bin'],'w');
count = fwrite(fwriteid,Compressed_Image,'uint8');
status = fclose(fwriteid);
fprintf('Done.\n')

%process with jean-jaques sparc printer driver. Note that “ride” is the
%  name of the Solaris box.
fprintf('Processing Binary with Printer Driver... ');
eval(['!rsh ride epson1200/epsP1200 -i epson1200/', filename,...
     '.bin -o epson1200/' filename '.out -h ' int2str(H) ' -w ', int2str(W)]);
fprintf('Done.\n');

%cleanup
fprintf('Deleting Temp Files... ');
eval(['!rm -f ' tf]);
eval(['!rm -f ' tf '_dithered']);
eval(['!rm -f /usr/people/lat3977/cis/epson1200/' filename '.bin']);
fprintf('Done.\n');
```

Characterization Targets

Two scripts are used to create the single ink ramp target and the Neugebauer patch target needed to characterize the printer for the YNSN model.

neugpatches.m

```
%model parameters
inks=6;
max_dc=255;

neugpri = zeros(2^inks,inks);
for i=1:inks;
  neugpri(:,i)=max_dc*bitget(1:(2^inks),i)';
end

patches = reshape(neugpri, 2^(inks/2), 2^(inks/2),inks);

%generate image of patch data
rimage = patch_image(patches, 1);

%Generate the epson binary file
epson_image(rimage, 'NeugPatches');

%print message
fprintf('Please Copy the NeugPatches.out file to the printer\n');
```

---

one_ink_ramps.m

```
%model parameters
  inks=6;
  ramp_steps=8;
  max_dc=255;

%create patch array
  dc = round(max_dc/ramp_steps * (ramp_steps:-1:1));
  patches = zeros(ramp_steps,inks,inks);
  for i=1:inks
    patches(:,i,i) = dc;
  end

%save the ramp dc levels
  save 'dc.txt' dc -ASCII;

%generate image of patch data
  rimage = patch_image(patches, 1);

%Generate the epson binary file
  epson_image(rimage, '1InkRamps');

%print message
  fprintf('Please Copy the 1InkRamps.out file to the printer\n');
```

Ink set Structure Initialization

The characterization and model parameters for the printing system are stored together in a structure called an *inkset* several functions and scripts are dedicated to building up the structure. *Init_model* loads the illuminant and color matching functions and then evokes a

function, *init_inkset6* that loads information specific to the six-color inkjet printer and stores it to the inkset structure.

---

init_model.m

```
addpath \dupont\pythsixink\munsell

global ill cmf ill_A ill_D65

%load the illuminants and color matching functions
cmf = load('CIE1931_2deg.txt');
ill_D65 = illD(6500);
ill_A = load('CIE_illA.txt');
ill = ill_D65;

%get the inkset
inkset = init_inkset6;
```

---

init_inkset6.m

```
% initalize the ink-set of the six-color inkjet printing system.

function inkset=init_inkset6;

datapath = '\dupont\pythsixink\data\';

%set the number of inks in the model
  inks = 6;

%Load the printer digital counts used to printer the calibration ramps
  dc = load([datapath, 'dc.txt']);
  numSteps = length(dc);

%Build the Neugebauer primary variable
  neugmeas = load([datapath, 'neugmeas.txt']);
  meas_per_sample = 5;
  neprmy = zeros(31,2^inks);
  rowtemp = (0:2^inks:2^inks*(meas_per_sample-1));
  for i=1:2^inks
    neprmy(:,i) = mean(neugmeas(rowtemp+i,:))';
  end

%define the paper reflectance from the last Neugebauer primary
  R_paper = neprmy(:,2^inks);

%Build the ramps variable
  rampmeas = load([datapath,'rampmeas.txt']);
  meas_per_sample = 5;
  ramps = zeros(31,numSteps+1,inks);
  rowtemp = (0:inks*numSteps:inks*numSteps*(meas_per_sample-1));
  for i=1:inks
    for j=1:numSteps
      ramps(:,j,i) = mean(rampmeas(rowtemp+j+(i-1)*numSteps,:))';
    end
    ramps(:,numSteps+1,i) = R_paper;
  end

  numSteps = numSteps+1; %white has been added in
  dc = [dc 0];

% read in n-value
  n = load([datapath, 'n.txt']);
```

```
% raise reflectance measurements to 1/n now so it only has to be done once
  neprmy_n = neprmy .^ (1/n);
  ramps_n = ramps .^ (1/n);
  R_paper_n = R_paper .^ (1/n);

%Calculate the LUT to go from effective area coverage to printer digital count
  eff2DClut = zeros(inks,numSteps);
  for i=1:inks
    eff2DClut(i,:) = inv_murr(ramps_n(:,:,i), R_paper_n, n);
  end

%Create the Demichel equation mask
  global dmask
  dmask = zeros(2^inks,inks);
  for i=1:inks;
    dmask(:,i)=bitget(1:(2^inks),i)';
  end

  %load the CQF
  cqf = load([datapath, 'cqf.txt']);

%return the inkset structure
inkset = struct('inks',inks,...
                'n',n,...
                'dc',dc,...
                'eff2DClut',eff2DClut,...
                'neprmy',neprmy,...
                'neprmy_n',neprmy_n,...
                'R_paper',R_paper,...
                'R_paper_n',R_paper_n,...
                'dmask',dmask,...
                'ramps',ramps,...
                'cqf',cqf);
```

Colorant space transformations

Several functions were implemented to transform between digital counts and effective area coverage as well as to determine the effective area coverage based on the Inverse Murray-Davies model (*inv_murr*).

---

dc2eff.m

```
% Convert digital counts to effective area coverage

function theo=dc2eff(eff_ac, inkset);

theo=zeros(size(eff_ac));
for i=1:inkset.inks
  theo(:,i) = interp1(inkset.dc, inkset.eff2DClut(i,:), eff_ac(:,i), 'spline');
end
```

---

eff2dc.m

```
% Convert effective area coverage to digital counts
```

```
function theo=eff2dc(est_ac, inkset);

theo=zeros(size(est_ac));
for i=1:inkset.inks
  theo(:,i) = round(interp1(inkset.eff2DClut(i,:), inkset.dc, est_ac(:,i), 'spline'));
end
```

---

inv_murr.m

```
% compute the effective area coverage of a reflectance vector

function dot_area=inv_murr(Rref_n, Rp_n, n)
[p,q] = size(Rref_n);
dot_area = pinv(Rref_n(:,1)-Rp_n)*(Rref_n-(Rp_n*ones(1,q)));
```

Yule-Nielsen-spectral-Neugebauer (Forward Model)

The forward model is implemented in a single function. The model relies on the reflectance data collected from the characterization target for the Neugebauer primaries. This reflectance data is already converted to 1/n space and stored in the *inkset* structure that is passed in as a function argument. The Demichel weightings are computed with the aid of the *dmask* variable which contains a set of binary values indicating which inks were in which Neugebauer primaries.

---

neug.m

```
% YNSN forward model

function R_predicted=neug(a, inkset)

area = ones(2^inkset.inks, 1) * a;
R_predicted = (inkset.neprmy_n *...
   prod(area.*inkset.dmask + (1-area).*~inkset.dmask, 2)) .^inkset.n;
```

Inverse Model Optimization

Inversion of the YNSN forward model is accomplished using non-linear optimization. The MATLAB optimization toolbox provided the optimization algorithm. The script *inverse6* go through the steps required to load a set of sample spectra and then match

them using the optimization routine. While matching each spectra the function *ink_sep* is called. The optimization objective function is called *sep1_obj_RMS* and calls the forward model using the current estimate of effective area coverage then computes the RMS spectral error between the model prediction and the original spectra.

---

inverse6.m

```
% Match reflectance spectra using six-color inkjet printing system

% initialize the required variables
init_model;
datapath = '\dupont\pythsixink\data\';

%uncommenting the following line will enable CQF/Matrix-R weighting
%inkset.cqf = ones(31,1);

%load the sample data reflectance spectra
sample = load('ColorChecker.txt');

fprintf('Begin inverse model processing.\n');

% start all area coverages as 0.1
est_ac=ones(size(sample,2),inkset.inks)*0.1;

%build placeholder for predicted values
predicted = zeros(31,size(sample,2));

% Loop through the sample reflectances one at a time
rms_error = zeros(size(sample,2),1);

output_stats = zeros(size(sample,2),3);

%Start the clock
clock_in = clock;

%start the flop count
flops_in = flops;

for i=1:size(sample,2);
   [rms_error(i), est_ac(i,:), exitflag, output]= ink_sep(sample(:,i), est_ac(i,:),
inkset);
   fprintf('Sample %d: est_ac=(%f %f %f %f %f %f)\n',i, ...
      est_ac(i,1),est_ac(i,2),est_ac(i,3),est_ac(i,4),est_ac(i,5),est_ac(i,6));
   predicted(:,i) = neug(est_ac(i,:), inkset);
   rms_error(i) = rms(predicted(:,i),sample(:,i));
   output_stats(i,1) = output.iterations;
   output_stats(i,2) = output.funcCount;
   output_stats(i,3) = exitflag;
end

%stop the flop count
flops_out = flops;

%Stop the Clock
clock_out = clock;

% Generate colorimetric results for reporting and plotting
ill = illD(6500);

%calculate the estimated digital counts to send to the printer
```

```
est_dc = eff2dc(est_ac, inkset);

start_secs = clock_in(4)*3600 + clock_in(5)*60 + clock_in(6);
end_secs = clock_out(4)*3600 + clock_out(5)*60 + clock_out(6);
fprintf('inverse model processing time: %f\n', end_secs - start_secs);
fprintf('seconds per pixel: %f\n', (end_secs - start_secs)/size(sample,2));
```

---

sep1_obj_RMS.m

```
function f=sep1_obj_RMS(dot_est, sample, inkset);
predicted = neug(dot_est', inkset);
f = RMS( sample.*inkset.cqf, predicted.*inkset.cqf );
```

RMS.m

```
function RMS = RMS ( reflectance1, reflectance2 )
% RMS: Returns the Root Mean Square Error between pairs of vectors
   sqr_error = (reflectance1-reflectance2).^2;
   RMS = ( mean ( sqr_error(:) ) )^(1/2);
```

---

ink_sep.m

```
function [spec_error, dot_est, oflag, out]=ink_sep(sample, start, inkset);
% Input parameters:
%   sample: measured spectral reflectance values 31 x 1
%   start: starting values of
%   inkset: structure containing printing system information and model
%            parameters

ink_est = start';

% These are the parameters for the optimization routine
options = optimset(...
  'Display','off', ...            % 'off','iter','final'
  'TolX',1e-6, ...
  'MaxIter',50,...
  'TolFun',1e-6, ...               % was 1e-4
  'Diagnostics','off',...
  'LargeScale','off');

% constraint parameters in the optimization
A = ones(1,inkset.inks);
b = 5.0; % maximum ink coverage
lb = zeros(1,inkset.inks);
ub = ones(1,inkset.inks);
nlcon = [];

[ink_out,spec_rms, exitflag,
output]=fmincon('sep1_obj_RMS',ink_est,A,b,[],[],lb,ub,nlcon, ...
     options,sample,inkset);

dot_est=ink_out';
spec_error=spec_rms;
oflag = exitflag;
out = output;
```

Spectral, Colorimetric and Metameric Reporting

To aid in assessing the results from the various spectral matching experiments a routine was written to pull together the various error metrics, colorimetric error vector plots and error histogram into one report.

---

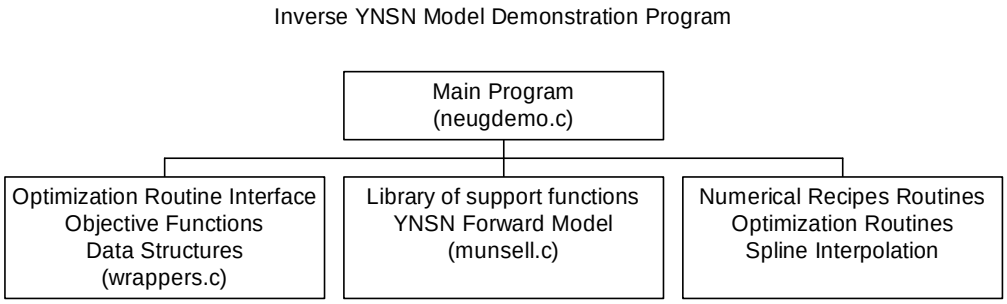
reportstats00.m

```
% Generate colorimetric results for reporting and plotting
XYZn = xyz(ones(31,1), cmf,ill);
LabSample=lab(xyz(sample,cmf,ill),XYZn);
LabPredicted=lab(xyz(predicted,cmf,ill),XYZn);
LabNeprmy=lab(xyz(neugpri,cmf,ill),XYZn);
delta_e94=deltaE94(LabSample, LabPredicted);
delta_eab=deltaEab(LabSample, LabPredicted);
delta_e00=deltaE00(LabSample, LabPredicted);
s = zeros(4,3);

%Colorimetric stats
s(1,1) = mean(delta_e00);
s(2,1) = std(delta_e00);
s(3,1) = max(delta_e00);
s(4,1) = min(delta_e00);

%Metameric stats
midx=meta_idx00(sample, predicted, inkset.R_paper, cmf, ill_D65, ill_A);
s(1,2) = mean(midx);
s(2,2) = std(midx);
s(3,2) = max(midx);
s(4,2) = min(midx);

%Spectral stats
rms_data = zeros(1,size(sample,2));
for i=1:size(sample,2)
   rms_data(i) = rms(sample(:,i),predicted(:,i));
end
s(1,3) = mean(rms_data);
s(2,3) = std(rms_data);
s(3,3) = max(rms_data);
s(4,3) = min(rms_data);

if printoutput

%create top level figure for the report
report = figure;
set(report, 'resize','off');
set(report, 'units','inches');
set(report, 'position',[.5,.5,7,9]);
set(report, 'PaperPositionMode','auto');

textblock = subplot('position',[0.01636 0.857638 0.482142 0.1168981]);
set(textblock,'visible','off');
text_title = text(0,1.1,0,report_title);


%output color difference statistics
t1 = sprintf('\\DeltaE^*_0_0 Between Sample Set Measurements and Predictions: \n');
t2 = sprintf( 'Mean                %3.2f\n', mean(delta_e00) );
t3 = sprintf( 'Standard Deviation  %3.2f\n', std(delta_e00) );
t4 = sprintf( 'Maximum             %3.2f\n', max(delta_e00) );
t5 = sprintf( 'Minimum             %3.2f\n', min(delta_e00) );
```

```
spectral_RMS = RMS(sample, predicted );
t6 = sprintf( 'RMS Spectral error  %3.2f\n', spectral_RMS );

text1 = text(0,.5,0,[t1,t2,t3,t4,t5,t6]);
set(text1, 'FontSize',8);
set(text1, 'FontName','Courier');

t7 = sprintf('Metameric Index (MI_0_0) under Ill. A:\n');
t8 = sprintf( 'Mean               %3.2f\n', mean(midx) );
t9 = sprintf( 'Standard Deviation %3.2f\n', std(midx) );
t10= sprintf( 'Maximum            %3.2f\n', max(midx) );
t11= sprintf( 'Minimum            %3.2f\n\n', min(midx) );

text2 = text(1.2,.5,0,[t7,t8,t9,t10,t11]);
set(text2, 'FontSize',8);
set(text2, 'FontName','Courier');

p1 = subplot('position',[0.0625, 0.0625, 0.4000, 0.3333]);
axis([-100,100,0,100]);
axis manual;
hold on;
plot(LabNeprmy(2,:), LabNeprmy(1,:), 'ko')
arrow([LabSample(2,:);LabSample(1,:)]',[LabPredicted(2,:);LabPredicted(1,:)]',...
   'length',4);
title('Measured->Predicted');
xlabel('a*');
ylabel('L*');

p2 = subplot('position',[0.5696,0.4942,0.4000,0.3333]);
axis([-100,100,0,100]);
axis manual;
hold on;
plot(LabNeprmy(3,:), LabNeprmy(1,:), 'ko')
arrow([LabSample(3,:);LabSample(1,:)]',[LabPredicted(3,:);LabPredicted(1,:)]',...
   'length',4);
title('Measured->Predicted');
xlabel('b*');
ylabel('L*');

p3 = subplot('position',[0.5696,0.0625,0.4000,0.3333]);
axis([-100,100,-100,100]);
axis manual;
hold on;
plot(LabNeprmy(2,:), LabNeprmy(3,:), 'ko')
arrow([LabSample(2,:);LabSample(3,:)]',[LabPredicted(2,:);LabPredicted(3,:)]',...
   'length',4);
title('Measured->Predicted');
xlabel('a*');
ylabel('b*');

p4 = subplot('position',[0.0655,0.4919,0.4000,0.3333]);
hist(delta_e00,20)
title('Histogram of \DeltaE^*_0_0');
xlabel('\DeltaE^*_0_0');
ylabel('Count of Samples');

end
```

APPENDIX C. RESEARCH SOURCE CODE IN C

The following diagram shows the structure of the C program that demonstrates the inverse Yule-Nielsen-spectral-Neugebauer model (YNSN). The program allows for the selection of the optimization routine, objective function and model starting point. A set of input spectra are matched using the six inks of the target printer. The predicted spectra and effective area coverage of the formulated matches are outputted in addition to the colorimetric and spectral error associated with each match. Source code for the Numerical Recipes routines have not been included because of their license agreement.



---

neugdemo.c

```
//Demonstration program for inversion of YNSN
#define BANDS 31

#include "nr.h"
#include "nrutil.h"
#include "wrappers.h"
#include <stdio.h>
#include <math.h>
#include "nnls.h"
#include "munsell.h"
#include <time.h>
#include <SIOUX.h>

int main(void);
struct inkset *init_inkset(void);


struct inkset *G_inkset;
struct optimization_settings *G_optimset;
```

```
int nfunc,ndfunc,noconverge;

float *G_ill;
float *G_cmf;

struct inkset *init_inkset(void) {
  struct inkset *iset;
  int i,j,k;
  float tempR1[BANDS], tempR2[BANDS];
  long m=BANDS,n=1,mode;
  float x[1], temp;

  //print section banner
  printf("\nPrinter characterization:\n");

  //allocate memory for inkset structure
  iset = (struct inkset *) malloc(sizeof(struct inkset));

  //set the number of inks in the model
  iset->inks = 6;

  //get the number of ramp steps
  iset->rampsteps = get_int("Enter the number of ramp steps",9);

  //load the printer digital counts used to pint the calibration ramps
  //note that the ramps must be in ascending order, with the first
value 0 and the last 255
  iset->dc = get_data("Select dc data file","data\\dc.txt",1,iset-
>rampsteps);

  //load the ramp reflectance data
  iset->ramps = get_data("Select the ramp reflectance
file","data\\ramps100601.txt",BANDS,iset->rampsteps*iset->inks);

  //Load neugebauer primary reflectance data
  iset->npri = get_data("Select Neugebauer Primary Reflectance
data","data\\neug100601.txt",BANDS,64);

  //Get the Yule-Nielsen n-value
  iset->N = get_float("Enter Yule-Nielsen n-value",6);

  //Compute the 1/n version of the neugebauer data
  iset->npriN = (float *) malloc(sizeof(float)*BANDS*pow(2,iset-
>inks));
  for(i=0;i<(BANDS*pow(2,iset->inks));i++){
    *(iset->npriN+i)=pow(*(iset->npri+i),1.0/iset->N);
  }

  //Set the paper reflectance
  iset->rpaper=(float *) malloc(sizeof(float)*BANDS);
  iset->rpaperN=(float *) malloc(sizeof(float)*BANDS);
  for(i=0;i<BANDS;i++){
    *(iset->rpaper+i) = *(iset->npri+i+BANDS*((long)pow(2,iset->inks)-
1));
    *(iset->rpaperN+i) = *(iset->npriN+i+BANDS*((long)pow(2,iset-
>inks)-1));
```

```
  }

  //calculate the LUT to go from effective area coverage to printer
digital counts
  iset->eff2dcLUT=(float *) malloc(sizeof(float)*iset->rampsteps*iset-
>inks);
  for(i=0;i<iset->inks;i++){
    for(j=1;j<iset->rampsteps;j++){
      for(k=0;k<BANDS;k++){

        //compute ink - paper
        tempR1[k]=pow(*(iset->ramps+(i+1)*BANDS*iset->rampsteps-
BANDS+k),1.0/iset->N)-*(iset->rpaperN+k);

        // compute rampstep - paper
        tempR2[k]=pow(*(iset->ramps+BANDS*(i*iset-
>rampsteps+j)+k),1.0/iset->N)-*(iset->rpaperN+k);
      }

      //store inverse yule-nielsen fit of area coverage to lut (clip to
max area coverage of 1)
      nnls(tempR1, &m, &n, tempR2, x, &mode);
      x[0] = (x[0]>1 ? 1:x[0]); //value might be slightly over 1
      *(iset->eff2dcLUT+i*iset->rampsteps+j) = x[0];
    }
    //set the first ramp step to 0% area coverage
    *(iset->eff2dcLUT+i*iset->rampsteps) = 0;
  }

  //Compute the second derivatives of for spline interpolation
  iset->eff2dcLUT2 = (float *) malloc(sizeof(float)*iset-
>rampsteps*iset->inks);
  for (i=0;i<iset->inks;i++) {
    temp = (*(iset->dc+1)-*(iset->dc))/(*(iset->eff2dcLUT+iset-
>rampsteps*i+1) -
      *(iset->eff2dcLUT+iset->rampsteps*i));
    spline(iset->eff2dcLUT+iset->rampsteps*i-1,
      iset->dc-1,iset->rampsteps,temp,0,iset->eff2dcLUT2+iset-
>rampsteps*i-1);
  }

  //allocate space for the unit ks values of the inkset
  iset->ks = (float *) malloc(sizeof(float)*(iset->inks+1)*BANDS);

  //compute KS of the paper
  for(i=0;i<BANDS;i++){
    temp = *(iset->rpaper+i);
    *(iset->ks+i) = (1-temp)*(1-temp)/2.0/temp;
  }

  //compute the ks values of the inks (using reflectance data in the
ramp dataset)
  for(i=0;i<iset->inks;i++){
    for(j=0;j<BANDS;j++){
      temp = *(iset->ramps+(i+1)*BANDS*iset->rampsteps-BANDS+j);
      *(iset->ks+(i+1)*BANDS+j)= (1-temp)*(1-temp)/2/temp - *(iset-
>ks+j);
```

```
    }
  }

  //allocate space for the lookup table that converts between effective
area coverage and concentration
  iset->c2effLUT=(float *) malloc(sizeof(float)*iset->rampsteps*iset-
>inks);

  //populate the area coverage to concentration lut
  for(i=0;i<iset->inks;i++){
    for(j=1;j<iset->rampsteps;j++){
      for(k=0;k<BANDS;k++){

        // copy ink ks value to tempR1
        tempR1[k]=*(iset->ks+(i+1)*BANDS+k);

        // compute rampstep KS - paper KS
        temp = *(iset->ramps+BANDS*(i*iset->rampsteps+j)+k);
        tempR2[k]=(1-temp)*(1-temp)/2/temp-*(iset->ks+k);
      }

      //store inverse km fit of area coverage to lut (clip to max
concentration of 1)
      nnls(tempR1, &m, &n, tempR2, x, &mode);
      x[0] = (x[0]>1 ? 1:x[0]); //value might be slightly over 1
      *(iset->c2effLUT+i*iset->rampsteps+j) = x[0];
    }
    //set the first ramp step to 0 concentration
    *(iset->c2effLUT+i*iset->rampsteps) = 0;
  }

  //Compute the second derivatives for spline interpolation
  iset->c2effLUT2 = (float *) malloc(sizeof(float)*iset-
>rampsteps*iset->inks);
  for (i=0;i<iset->inks;i++) {
    temp = (*(iset->eff2dcLUT+1)-*(iset->eff2dcLUT))/(*(iset-
>c2effLUT+iset->rampsteps*i+1)
      - *(iset->c2effLUT+iset->rampsteps*i));
    spline(iset->c2effLUT+iset->rampsteps*i-1,iset->eff2dcLUT+iset-
>rampsteps*i-1,iset->rampsteps,
      temp,0,iset->c2effLUT2+iset->rampsteps*i-1);
  }

  return iset;
}

int main( void ) {
  struct optimization_settings optimset;
  struct target original, predicted;
  float *ones, temp, *cmf, *ill, dc[6];
  float *DEab, *DE94, *DE00, *rms_data;
  int i,j, *fcount, *dfcount, *verge, count;
  char outputname[150];
  clock_t start_time, end_time;
  FILE *output_file, *output_file2;
  float inkmask[6]={1,1,0,0,1,1};
```

```
  SIOUXSettings.autocloseonquit = 0;
  SIOUXSettings.asktosaveonclose = 0;

  //Display welcome message
  printf("YNSN inversion demonstration program\n");
  printf("(C) 2001 Munsell Color Science Laboratory - Rochester
Institute of Technology\n");
  printf("LAT (10/13/01)\n\n");

  G_inkset = init_inkset();
  G_optimset = &optimset;

  //print section banner
  printf("\nSample Selection:\n");

  //get the count of samples
  count = get_int("Enter Sample Count",24);

  //Read in the sample data
  original.ref = get_data("Select Sample Refelectance
File","data\\ColorChecker.txt",BANDS,count);

  //print section banner
  printf("\nColorimetric Data:\n");

  //Get CMF data
  cmf = get_data("Select CIE CMF
File","data\\CIE1931_2deg.txt",BANDS,3);
  G_cmf = cmf;

  //Get Illuminant data
  ill = get_data("Select Illuminant
data","data\\CIE_illD65.txt",BANDS,1);
  G_ill = ill;

  //print section banner
  printf("\nOptimization Settings:\n");

  //select optimization algorithm
  printf("Select the optimization algorithm:\n");
  printf("       1)DFP\n");
  printf("       2)FRPR\n");
  printf("       3)Powell\n");
  printf("       4)Amoeba\n");
  printf("       5)KS\n");
  optimset.algorithm = (enum algorithms) get_int("Specify Number",1);

  //select optimization objective
  printf("\nSelect the optimization objective:\n");
  printf("       1)RMS\n");
  printf("       2)DEab\n");
  printf("       3)RMS(Matrix-R)\n");
  printf("       4)Multistage (RMS->DEab)\n");
  optimset.objective = (enum objectives) get_int("Specify Number",4);

  if(optimset.objective == matr_obj){
    //Get Matrix R Data
```

```
    optimset.swf = get_data("Select spectral weighting function Data
File","data\\matrixr.txt",BANDS,1);
  }

  if(optimset.objective == multi_obj){
    //Select second pass tolerence
    optimset.tweak_range=get_float("Enter the second pass adjustment
range",0.05);
  }

  //get the algorithm starting value
  temp = get_float("Enter optimization starting value (Number between 0
and 1, or -1 for KS)",-1);
  optimset.start_val = (float *) malloc(sizeof(float)*6);
  for (i=0;i<6;i++) *(optimset.start_val+i) = temp;

  //allocate memory
  original.ac = (float*) malloc(sizeof(float)*6*count); // area
coverage
  predicted.ac = (float*) malloc(sizeof(float)*6*count); // area
coverage
  predicted.dc = (int*) malloc(sizeof(int)*6*count); // area coverage
  predicted.ref = (float *) malloc(sizeof(float)*BANDS*count); //
predicted reflectance
  fcount = (int *) malloc(sizeof(int)*count); // calls to objective
function
  dfcount = (int *) malloc(sizeof(int)*count); // calls to function
derivative
  verge = (int *)  malloc(sizeof(int) * count); // samples that don't
converge
  optimset.lb = (float *) malloc(sizeof(float)*6); // lower bound for
optimization
  optimset.ub = (float *) malloc(sizeof(float)*6); // upper bound for
optimization

  //print section banner
  printf("\nOutput Files:\n");

  //Select Area Coverage Output File
  printf("Select Output filename for digital counts and colorimetric
data <results\\output2.txt>:");
  get_string(outputname,"results\\output2.txt");
  if ((output_file=fopen(outputname,"w")) == (FILE *) NULL) {
    printf("*** error opening output file. ***\n");
    return 1;
  }

  //Select Area Coverage Output File
  printf("Select Output filename for predicted reflectance data
<results\\output_ref2.txt>:");
  get_string(outputname,"results\\output_ref2.txt");
  if ((output_file2=fopen(outputname,"w")) == (FILE *) NULL) {
    printf("*** error opening output file. ***\n");
    return 1;
  }

  //calculate XYZ values for samples
```

```
original.XYZ = (float *) malloc (sizeof (float) * 3 * count);
ciexyz(cmf,ill,original.ref,original.XYZ,count);

//calculate XYZn for Whitepoint
ones = (float *) malloc (sizeof (float) * BANDS);
for(i=0;i<BANDS;i++) *(ones+i)=1;
original.XYZn = (float *) malloc(sizeof(float)*3);
optimset.XYZn = (float *) malloc(sizeof(float)*3);
ciexyz(cmf,ill,ones,original.XYZn,1);
ciexyz(cmf,ill,ones,optimset.XYZn,1);

//calculate CIELAB values of samples
original.Lab = (float *) malloc (sizeof (float) * 3 * count);
cielab(original.XYZ,original.XYZn,original.Lab,count);

printf("\nProcessing Samples\n");
start_time=clock();

//Loop through samples and find match through nonlinear optimization
for(i=0;i<count;i++){
  optimset.orig_ref = original.ref+BANDS*i;
  optimset.pred_ref = predicted.ref+BANDS*i;
  optimset.pred_ac = predicted.ac+6*i;

  //intialize lower and upper bounds to 0 and 1
  for (j=0;j<6;j++) {
    *(optimset.lb+j)=0*inkmask[j];
    *(optimset.ub+j)=1*inkmask[j];
  }

  //reset the function counter
  nfunc=ndfunc=noconverge=0;

  //set the objective function pointers
  switch (optimset.objective) {
    case rms_obj:
      optimset.func = obj_rms;
      optimset.dfunc = dobj_rms;
      break;
    case deab_obj:
      optimset.func = obj_deab;
      optimset.dfunc = dobj_deab;
      break;
    case matr_obj:
      optimset.func = obj_matr;
      optimset.dfunc = dobj_matr;
      break;
    case multi_obj:
      optimset.func = obj_rms;
      optimset.dfunc = dobj_rms;
      break;
  }

  //set the starting values
  if (optimset.start_val[0] == -1)
    ksmatch();
  else
```

```
      for(j=0;j<6;j++)
*(optimset.pred_ac+i)=optimset.start_val[j]*inkmask[j];

    // run the first stage of optimization
    switch (optimset.algorithm) {
      case dfp_algo:
        dfpmatch();
        break;
      case frpr_algo:
        frprmatch();
        break;
      case powell_algo:
        powellmatch();
        break;
      case amoeba_algo:
        amoebamatch();
        break;
      case ks_algo:
        ksmatch();
        break;
      }

    //run the second stage of optimization if selected
    if (optimset.objective==multi_obj){
      //loop through ink area coverages and set bounds
      for (j=0;j<6;j++){
        *(optimset.lb+j)=(*(predicted.ac+6*i+j)-
optimset.tweak_range)*inkmask[j];

*(optimset.ub+j)=(*(predicted.ac+6*i+j)+optimset.tweak_range)*inkmask[j
];
        if (*(optimset.lb+j)<0) *(optimset.lb+j)=0;
        if (*(optimset.ub+j)>1) *(optimset.ub+j)=1;
      }

      //change to the second objective function (DEab)
      optimset.func = obj_deab;
      optimset.dfunc = dobj_deab;

      // run the second stage of optimization
      switch (optimset.algorithm) {
        case dfp_algo:
          dfpmatch();
          break;
        case frpr_algo:
          frprmatch();
          break;
        case powell_algo:
          powellmatch();
          break;
        case amoeba_algo:
          amoebamatch();
          break;
        case ks_algo:
          ksmatch();
        break;
      }
```

```
    }

      //convert predicted area coverage to digital counts
      for(j=0;j<6;j++){
      splint(G_inkset->eff2dcLUT+9*j-1,G_inkset->dc-1,G_inkset-
>eff2dcLUT2+9*j-1,9,*(predicted.ac+j+i*6),dc+j);
      *(predicted.dc+j+i*6) = round(*(dc+j));
    }

    printf("Sample %3d [",i+1);
    //for(j=0;j<6;j++) printf("%1.2f (%3.0f) ",*(predicted.ac+j+i*6),
round(dc[j]));
    for(j=0;j<6;j++) printf("%3d ", *(predicted.dc+j+i*6));
    printf("]");
      printf(" Func: %5d",nfunc);
      printf(" Deriv: %5d\n",ndfunc);

      *(fcount+i)=nfunc;
      *(dfcount+i)=ndfunc;
      *(verge+i)=noconverge;
  }
  end_time=clock();

  //  calculate XYZ values for predictions
  predicted.XYZ = (float *) malloc (sizeof (float) * 3 * count);
  ciexyz(cmf,ill,predicted.ref,predicted.XYZ,count);

  //  calculate XYZn for Whitepoint
  predicted.XYZn = (float *) malloc(sizeof(float)*3);
  ciexyz(cmf,ill,ones,predicted.XYZn,1);

  //  calculate CIELAB values of predictions
  predicted.Lab = (float *) malloc (sizeof (float) * 3 * count);
  cielab(predicted.XYZ,predicted.XYZn,predicted.Lab,count);

  //calculate CIELAB Delta Eab values of each sample
  DEab = (float *) malloc (sizeof (float) * count);
  deltaEab(original.Lab,predicted.Lab,DEab,count);

  //calculate CIELAB Delta E94 values of each sample
  DE94 = (float *) malloc (sizeof (float) * count);
  deltaE94(original.Lab,predicted.Lab,DE94,count);

  //calculate CIELAB Delta E00 values of each sample
  DE00 = (float *) malloc (sizeof (float) * count);
  deltaE00(original.Lab,predicted.Lab,DE00,count);

  //calculate rms between the samples and predictions
  rms_data = (float *) malloc (sizeof (float) * count);
  for(i=0;i<count;i++)

*(rms_data+i)=rms(predicted.ref+i*BANDS,original.ref+i*BANDS,BANDS);

  //print colorimetric results
  for(i=0;i<count;i++){
    printf("Sample %3d DeltaE94=%5.2f RMS=%6.4f Func=%d\n", i+1,
*(DEab+i), *(rms_data+i),*(fcount+i));
```

```
  }
  printf("Pixel Processing Speed: %5.3f pixels/sec \n",count/((double)
(end_time-start_time)/CLOCKS_PER_SEC));

  //Write out Header for first output file (Area Coverage and
Colorimetric Stats

fprintf(output_file,"Sample\tDCk\tDCc\tDCm\tDCy\tDCg\tDCo\tDEab\tDE94\t
DE00\tRMS\tFcount\tDFcount\tNoConverge\n");

  for(i=0;i<count;i++){
    //sample number
    fprintf(output_file,"%d\t",i+1);

    //predicted digital counts
    for(j=0;j<6;j++)
      fprintf(output_file,"%3d\t",*(predicted.dc+j+i*6));

    //DeltaEab DeltaE94 DeltaE00

fprintf(output_file,"%7.3f\t%7.3f\t%7.3f\t%6.4f\t",*(DEab+i),*(DE94+i),
*(DE00+i),*(rms_data+i));

    //calls to OBJ_RMS and DOBJ_RMS, noconverge flag
    fprintf(output_file,"%d\t%d\t%d\n",*(fcount+i), *(dfcount+i),
*(verge+i));
  }

  //Close the output file
  fclose(output_file);

  //write out the reflectance data
  for(i=0;i<count;i++){
    //sample number
    fprintf(output_file2,"%d\t",i+1);

    //predicted reflectance
    for(j=0;j<BANDS;j++)
      fprintf(output_file2,"%6.4f\t",*(predicted.ref+j+i*BANDS));

    fprintf(output_file2,"\n");
  }

  //Close the second output file
  fclose(output_file2);
  printf("Done.\n");

  return 0;
}
```

---

wrappers.h

```
//Optimization Algorithm Wrappers for YNSN
#define BANDS 31
#define NDIM 6
```

```
#define FTOL 1.0e-5
#define PIO2 1.5707963
#define MP 7
#define NP 6
#define GTOL 1.0e-4

void rescale(float *X, float *lb, float *ub, float *out);
void unscale(float *x, float *lb, float *ub, float *out);
float obj_rms(float x[]);
void dobj_rms(float x[],float df[]);
float obj_deab(float x[]);
void dobj_deab(float x[],float df[]);
float obj_matr(float x[]);
void dobj_matr(float x[],float df[]);
void dfpmatch();
void frprmatch();
void powellmatch();
void amoebamatch();
void ksmatch();

//enumerate the algorithms
enum algorithms { dfp_algo=1, frpr_algo=2, powell_algo=3, amoeba_algo=4,
ks_algo=5 };

//enumerate the objective functions
enum objectives { rms_obj=1, deab_obj=2 , matr_obj=3, multi_obj=4 };

struct inkset {
  int inks;
  float N;
  float *dc;
  float *eff2dcLUT;
  float *eff2dcLUT2;
  float *c2effLUT;
  float *c2effLUT2;
  float *npri;
  float *npriN;
  float *rpaper;
  float *rpaperN;
  float *ks;
  int rampsteps;
  float *ramps;
  };

struct optimization_settings {
  enum algorithms algorithm;
  enum objectives objective;
  float (*func)(float []);
  void (*dfunc) ( float [], float []);
  float *start_val;
  float *lb;
  float *ub;
  float tweak_range;
  float *swf; //spectral weighting function
  float *orig_ref;
  float *pred_ref;
  float *pred_ac;
```

```
  float *ill;
  float *cmf;
  float *XYZn;
  };

struct target {
  long count;
  char *desc;
  float *ref;
  float *ac;
  int *dc;
  float *XYZ;
  float *XYZn;
  float *Lab;
  };
```

---

wrappers.h

```
//Optimization Algorithm Wrappers for YNSN
#include <math.h>
#include "wrappers.h"
#include "nr.h"
#include "nrutil.h"
#include "munsell.h"
#include "nnls.h"

extern int nfunc,ndfunc,noconverge;
extern struct optimization_settings *G_optimset;
extern struct inkset *G_inkset;
extern float *G_cmf;
extern float *G_ill;

long idum=(-64);

//rescale the values of vector x from sine space back to lb <= x <= ub
void rescale(float *x, float *lb, float *ub, float *out){
  int i;

  for (i=0;i<6;i++)
    *(out+i) = *(lb+i) + (*(ub+i)-*(lb+i))*(sin(*(x+i))+1)/2;
}

//unscale the vector x into sine space
void unscale(float *x, float *lb, float *ub, float *out){
  int i;

  for (i=0;i<6;i++){
    if (*(lb+i)==*(ub+i))
      *(out+i) = 0;
    else
      *(out+i) = -asin((2**(x+i)-*(lb+i)-*(ub+i))/(*(lb+i)-*(ub+i)));
  }
}

float obj_rms(float x[]){
```

```
  float ac[6];
  float pred[BANDS];

  // increment function call counter
  nfunc++;

  // convert from repeating SINE space
  rescale(x+1,G_optimset->lb,G_optimset->ub,ac);

  // evaluate the current ac levels with YNSN
  ynsn(G_inkset->npriN,ac,G_inkset->N,pred);

  return rms(G_optimset->orig_ref,pred,BANDS);
}

void dobj_rms(float x[],float df[])
{
  int i;
  float f;
  float dx=0.001;

  ndfunc++;
  f=obj_rms(x); //objective function at current point

  for (i=1;i<7;i++){
    x[i]+=dx;
    df[i]= (obj_rms(x)-f)/dx; //partial derivative
    x[i]-=dx;
  }
}

float obj_deab(float x[]){
  float ac[6];
  float pred[BANDS];
  float xyz_pred[3],xyz_orig[3],lab_pred[3],lab_orig[3];
  float deab;

  // increment function call counter
  nfunc++;

  // convert from repeating SINE space
  rescale(x+1,G_optimset->lb,G_optimset->ub,ac);

  // evaluate the current ac levels with YNSN
  ynsn(G_inkset->npriN,ac,G_inkset->N,pred);

  // calculate xyz values
  ciexyz(G_cmf,G_ill,G_optimset->orig_ref,xyz_orig,1);
  ciexyz(G_cmf,G_ill,pred,xyz_pred,1);

  // calculate CIELAB values
  cielab(xyz_orig,G_optimset->XYZn,lab_orig,1);
  cielab(xyz_pred,G_optimset->XYZn,lab_pred,1);

  // calculate color difference
  deltaEab(lab_orig,lab_pred,&deab,1);
```

```
  return deab;
}

void dobj_deab(float x[],float df[])
{
  int i;
  float f;
  float dx=0.001;

  ndfunc++;
  f=obj_deab(x); //objective function at current point

  for (i=1;i<7;i++){
    x[i]+=dx;
    df[i]= (obj_deab(x)-f)/dx; //partial derivative
    x[i]-=dx;
  }
}

float obj_matr(float x[]){
  float ac[6];
  float pred[BANDS];
  float orig[BANDS];
  int i;

  nfunc++; //increment function call counter
  rescale(x+1,G_optimset->lb,G_optimset->ub,ac); //convert from
repeating SINE space
  ynsn(G_inkset->npriN,ac,G_inkset->N,pred); //evaluate the current ac
levels with YNSN
  for (i=0;i<BANDS;i++){
    pred[i] *= *(G_optimset->swf+i);
    orig[i] = *(G_optimset->orig_ref+i)**(G_optimset->swf+i);
  }
  return rms(orig,pred,BANDS);
}

void dobj_matr(float x[],float df[])
{
  int i;
  float f;
  float dx=0.001;

  ndfunc++;
  f=obj_matr(x); //objective function at current point

  for (i=1;i<7;i++){
    x[i]+=dx;
    df[i]= (obj_matr(x)-f)/dx; //partial derivative
    x[i]-=dx;
  }
}


void dfpmatch(){
  int iter;
  float p[7],fret;
```

```
  // get the starting value (in colorant space) and move it to sine
space
  unscale(G_optimset->pred_ac, G_optimset->lb, G_optimset->ub, p+1);

  // call optimizer
  dfpmin(p,NP,GTOL,&iter,&fret,G_optimset->func,G_optimset->dfunc);

  // convert back from Sine space
  rescale(p+1,G_optimset->lb,G_optimset->ub,G_optimset->pred_ac);

  // compute predicted reflectance with YNSN model
  ynsn(G_inkset->npriN,G_optimset->pred_ac,G_inkset->N,G_optimset-
>pred_ref);
}

void frprmatch(){

  int iter;
  float fret,p[7];

  // get the starting value (in colorant space) and move it to sine
space
  unscale(G_optimset->pred_ac, G_optimset->lb, G_optimset->ub, p+1);

  // call optimizer
  frprmn(p,NDIM,FTOL,&iter,&fret,G_optimset->func,G_optimset->dfunc);

  // convert back from Sine space
  rescale(p+1,G_optimset->lb,G_optimset->ub,G_optimset->pred_ac);

  // compute predicted reflectance with YNSN model
  ynsn(G_inkset->npriN,G_optimset->pred_ac,G_inkset->N,G_optimset-
>pred_ref);
}

void amoebamatch(){

  int i,tnfunc,j,ndim=NP;
  float *x,*y,**p;
  float high[6],low[6];

  // initalize values
  x=vector(1,NP);
  y=vector(1,MP);
  p=matrix(1,MP,1,NP);

  // set the starting values
  for (i=0;i<6;i++){
    low[i] = 0;
    high[i] = 1;
  }
  unscale(high,G_optimset->lb,G_optimset->ub,high);
  unscale(low,G_optimset->lb,G_optimset->ub,low);

  //initalize matrix p and vectors x and y
  for (i=1;i<=MP;i++) {
```

```
    for (j=1;j<=NP;j++)
      x[j]=p[i][j]=(i == (j+1) ? high[i-1] : low[i-1]);
    y[i]=obj_rms(x);
  }

  amoeba(p,y,ndim,FTOL,G_optimset->func,&tnfunc); //call the optimizer
  rescale(p[1]+1,G_optimset->lb,G_optimset->ub,G_optimset->pred_ac);
//convert back from sine space
  ynsn(G_inkset->npriN,G_optimset->pred_ac,G_inkset->N,G_optimset-
>pred_ref); //compute the predicted reflectance

  free_matrix(p,1,MP,1,NP);
  free_vector(y,1,MP);
  free_vector(x,1,NP);
}

void powellmatch(){

  float p[7];
  int i,iter,j;
  float fret,**xi;

  // get the starting value (in colorant space) and move it to sine
space
  unscale(G_optimset->start_val, G_optimset->lb, G_optimset->ub, p+1);

  xi=matrix(1,NDIM,1,NDIM);
    for (i=1;i<=NDIM;i++)
      for (j=1;j<=NDIM;j++)
        xi[i][j]=(i == j ? 1.0 : 0.0);
  powell(p,xi,NDIM,FTOL,&iter,&fret,G_optimset->func);

  free_matrix(xi,1,NDIM,1,NDIM);

  // convert back from Sine space
  rescale(p+1,G_optimset->lb,G_optimset->ub,G_optimset->pred_ac);

  // compute predicted reflectance with YNSN model
  ynsn(G_inkset->npriN,G_optimset->pred_ac,G_inkset->N,G_optimset-
>pred_ref);
}

void ksmatch(){
  int i,j;
  float otemp[BANDS];
  float * gtest = G_inkset->ks;
  float x[6];
  float ks[BANDS*6];
  long m,n;
  long mode;

  //convert the original reflectance to K/S Space and subtract paper
k/s
  for (i=0;i<BANDS;i++)
    otemp[i]=(1-*(G_optimset->orig_ref+i)**(G_optimset->orig_ref+i)) /
2.0 / *(G_optimset->orig_ref+i)-*(G_inkset->ks+i);
```

```
  //copy the k/s values of the inks to a scratch space
  m=BANDS;
  n=6;
  for (i=0;i<m;i++)
    for (j=0;j<n;j++)
      ks[i+j*m]=*(G_inkset->ks+i+(j+1)*m);

  //make a least squares match with ink k/s values of inks
  nnls(ks, &m, &n, otemp, x, &mode);

  for(i=0;i<6;i++){
    if (x[i]>1) x[i]=1;
    //*(G_optimset->pred_ac+i)=x[i];
    splint(G_inkset->c2effLUT+9*i-1,G_inkset->eff2dcLUT+9*i-1,G_inkset-
>c2effLUT2+9*i-1,9,x[i],G_optimset->pred_ac+i);
  }


  //compute the final predicted reflectance
  ynsn(G_inkset->npriN,G_optimset->pred_ac,G_inkset->N,G_optimset-
>pred_ref);
}
```

---

munsell.h

```
//MCSL Library of Color Science Support Functions

//get string from user
void get_string(char *output, char *defaultval);

//get a number from the terminal
float get_float(char *prompt, float defaultval);

//get a number from the terminal
int get_int(char *prompt, int defaultval);

//Get a line from the terminal
void read_line(char *buffer);

//Read in a chunk of data
float *read_data(char * filename, int rows, int cols);

//Prompt for filename and get data
float *get_data(char *prompt, char *defaultname, int rows, int cols);

//Calculate Tristimulus values (XYZ)
void ciexyz(float *cmf, float *ill, float *ref,float *out,int count);

//Calculate CIELAB values
void cielab(float *XYZ, float *XYZn, float *out, int count);

//Calculate CIELAB color difference
void deltaEab(float *Lab1, float *Lab2, float* out, int count);

//Calculate DE*94 color difference
```

```
void deltaE94(float *Lab1, float *Lab2, float *out, int count);

//Calculate DeltaE00 color difference
void deltaE00(float *Lab1, float *Lab2, float *out, int count);

//Calculate RMS Error between two vectors
float rms(float *ref1, float *ref2, int count);

//YNSN Model for Six Ink System
void ynsn(float *npriN, float *ac, float N, float *pred);
```

---

munsell.c

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include "munsell.h"
#include "nnls.h"

#define BANDS 31
#define D2R 0.01745329251994

//get string
void get_string(char *output, char *defaultval){
  read_line(output);
  if (sscanf(output,"%s")!=1)
    strcpy(output, defaultval);
}

//Get a number from the user
int get_int(char *prompt, int defaultval){
  float value;
  char buffer[150];
  printf("%s <%d>: ",prompt,defaultval);
  read_line(buffer);
  if (sscanf(buffer,"%f",&value)!=1)
    value = defaultval;
  return value;
}

//Get a number from the user
float get_float(char *prompt, float defaultval){
  float value;
  char buffer[150];
  printf("%s <%5.3f>: ",prompt,defaultval);
  read_line(buffer);
  if (sscanf(buffer,"%f",&value)!=1)
    value = defaultval;
  return value;
}

//Read a line of text from the console
void read_line(char *buffer){
  char character;
```

```
  int i=0;

  do{
    character = getchar();
    *(buffer+i)=character;
    i++;
  }
  while (character !='\n');
  *(buffer+i-1)='\0';
}

//Read in a chunk of data from a file
float *read_data(char * filename, int rows, int cols){
  FILE *in;
  float *data;
  int i,j;

  if ((in=fopen(filename,"r")) == (FILE *) NULL) {
    printf("*** data file %s could not be opened. ***\n", filename);
    exit(1);
  }
  data = (float *) malloc (sizeof (float) * rows * cols);
  for (i=0;i<rows;i++)
    for (j=0;j<cols;j++){
      fscanf(in, "%f",(data+i+rows*j));
  }
  fclose(in);
  return data;
}

//Prompt for filename and get data
float *get_data(char *prompt, char *defaultname, int rows, int cols){
  char filename[150];

  printf("%s <%s>: ",prompt, defaultname);
  read_line(filename);
  if (filename[0]!='\0')
    return read_data(filename, rows, cols);
  else
    return read_data(defaultname, rows, cols);
}


//Calculate Tristimulus values (XYZ)
void ciexyz(float *cmf, float *ill, float *ref,float *out,int count){
  int i,j,k;
  float temp;
  float K=0;
  float *p1;

  //calculate constant K as sum(ybar*ill)/100
  for(i=0;i<BANDS;i++){
   K+=*(cmf+BANDS+i) * *(ill+i);
  }
  K=1/K*100;

  //calculate the XYZ values
```

```
  p1 = out;
  for(i=0;i<count;i++){
    for(j=0;j<3;j++){
      temp=0;
      for(k=0;k<BANDS;k++) temp+=*(cmf+k+BANDS*j) * *(ill+k) *
*(ref+k+BANDS*i);
       *p1 = temp*K;
       p1++;
    }
  }
}

//Calculate cielab Values
void cielab(float *XYZ, float *XYZn, float *out, int count){

  int i,j;
  float temp[3];


  //loop through the samples
  for (i=0;i<count;i++){
    //calculate ratios and corrected negative portion
    for (j=0;j<3;j++){
      temp[j]=*(XYZ+i*3+j)/XYZn[j];
      if (temp[j] > 0.008856)
        temp[j]=pow(temp[j],1.0/3.0);
      else
        temp[j]=temp[j]*7.787+(16.0/116);
     }

     //calculate L*
     *(out+i*3)=116*temp[1]-16;

     //calculate a*
     *(out+i*3+1)=500*(temp[0]-temp[1]);

     //calculate b*
     *(out+i*3+2)=200*(temp[1]-temp[2]);
   }
}

//Calculate CIELAB color difference
void deltaEab(float *Lab1, float *Lab2, float *out, int count){
  int i,j;

  for(i=0;i<count;i++){
    *out=0;
    for(j=0;j<3;j++){
      *out+=pow(*(Lab1)-*(Lab2),2);
      Lab1++;
      Lab2++;
    }
    *out=sqrt(*out);
    out++;
  }
}
```

```
//Calculate DeltaE*94 color difference
void deltaE94(float *Lab1, float *Lab2, float *out, int count){
  int i;
  float C1, C2, DL, DC, DH, SL, SC, SH, DEab;

  for(i=0;i<count;i++){
    deltaEab(Lab1, Lab2, &DEab, 1);
    C1 = sqrt(*(Lab1+1)**(Lab1+1) + *(Lab1+2)**(Lab1+2));
    C2 = sqrt(*(Lab2+1)**(Lab2+1) + *(Lab2+2)**(Lab2+2));
    DL = *Lab1-*Lab2;
    DC = C1-C2;
    DH = sqrt(DEab*DEab - (*Lab1-*Lab2)*(*Lab1-*Lab2) - DC*DC);
    SL = 1;
    SC = 1 + 0.045*C2;
    SH = 1 + 0.015*C2;
    *out = sqrt(DL*DL/SL/SL + DC*DC/SC/SC + DH*DH/SH/SH);
    Lab1 +=3;
    Lab2 +=3;
    out++;
  }
}

//Calculate DeltaE00 color difference
void deltaE00(float *Lab1, float *Lab2, float *out, int count){
  int i;
  float C1, C2, mC, G, Pa1, Pa2, CP1, CP2, hP1t, hP2t, hP1, hP2;
  float DLP,DCP,DhP,DHP,mLP,mCP,mhP,SL,SC,T,SH,RC,DTheta,RT;

  for (i=0;i<count;i++){

    //CIELAB Chroma
    C1 = sqrt(*(Lab1+1)**(Lab1+1) + *(Lab1+2)**(Lab1+2));
    C2 = sqrt(*(Lab2+1)**(Lab2+1) + *(Lab2+2)**(Lab2+2));

    //Lab Prime
    mC = (C1+C2)/2;
    G=0.5*(1-sqrt(pow(mC,7)/(pow(mC,7)+pow(25,7))));
    Pa1 = *(Lab1+1)*(1+G);
    Pa2 = *(Lab2+1)*(1+G);

    //Chroma
    CP1 = sqrt(Pa1*Pa1 + *(Lab1+2)**(Lab1+2));
    CP2 = sqrt(Pa2*Pa2 + *(Lab2+2)**(Lab2+2));

    //Hue Angle
    hP1t = atan2(*(Lab1+2),Pa1)/D2R;
    hP2t = atan2(*(Lab2+2),Pa2)/D2R;

    //Add in 360 to the smaller hue angle if absolute value of
difference is > 180
    hP1 = hP1t + ((hP1t<hP2t)&(fabs(hP1t-hP2t)>180))*360;
    hP2 = hP2t + ((hP1t>hP2t)&(fabs(hP1t-hP2t)>180))*360;

    //Delta Values
    DLP = *Lab1 - *Lab2;
    DCP = CP1 - CP2;
    DhP = hP1 - hP2;
```

```
    DHP = 2*sqrt(CP1*CP2)*sin(DhP/2*D2R);

    //Arithmetic mean of LCh' values
    mLP = (*Lab1+*Lab2)/2;
    mCP = (CP1+CP2)/2;
    mhP = (hP1+hP2)/2;

    //Weighting Functions
    SL = 1+(0.015*(mLP-50)*(mLP-50))/sqrt(20+(mLP-50)*(mLP-50));
    SC = 1+0.045*mCP;
    T = 1-0.17*cos((mhP-
30)*D2R)+0.24*cos(2*mhP*D2R)+0.32*cos((3*mhP+6)*D2R)-0.2*cos((4*mhP-
63)*D2R);
    SH = 1+0.015*mCP*T;

    //Rotation function
    RC = 2*sqrt(pow(mCP,7)/(pow(mCP,7)+pow(25,7)));
    DTheta = 30*exp(-((mhP-275)/25)*((mhP-275)/25));
    RT = -sin(2*DTheta*D2R)*RC;

    *out =
sqrt((DLP*DLP/SL/SL)+(DCP*DCP/SC/SC)+(DHP*DHP/SH/SH)+(RT*(DCP/SC)*(DHP/
SH)));
    Lab1+=3;
    Lab2+=3;
    out++;
}

}

//Calculate RMS Error between two vectors
float rms(float *ref1, float *ref2, int count){
  int i;
  float temp;

  temp=0;
  for(i=0;i<count;i++){
    temp+=pow((*(ref1)-*(ref2)),2)/count;
    ref1++;
    ref2++;
  }
  return(sqrt(temp));
}

//YNSN Model for Six Ink System
void ynsn(float *npriN, float *ac, float N, float *pred){
  int i,j;
  float w;

  //initalize the predicted value back to zeros
  for (i=0;i<BANDS;i++) *(pred+i)=0;

  //loop through neugebauer primaries and weight with demichel
  for (i=0;i<64;i++) {
    w=1;
    for (j=0;j<6;j++) w*=(((i+1)&(1<<j))?*(ac+j):1-*(ac+j));
    for (j=0;j<BANDS;j++) *(pred+j) += w* *(npriN + BANDS*i + j);
```

```
  }

  // convert back from 1/n space
  for (j=0;j<BANDS;j++) *(pred+j) = pow(*(pred+j),N);
}
```