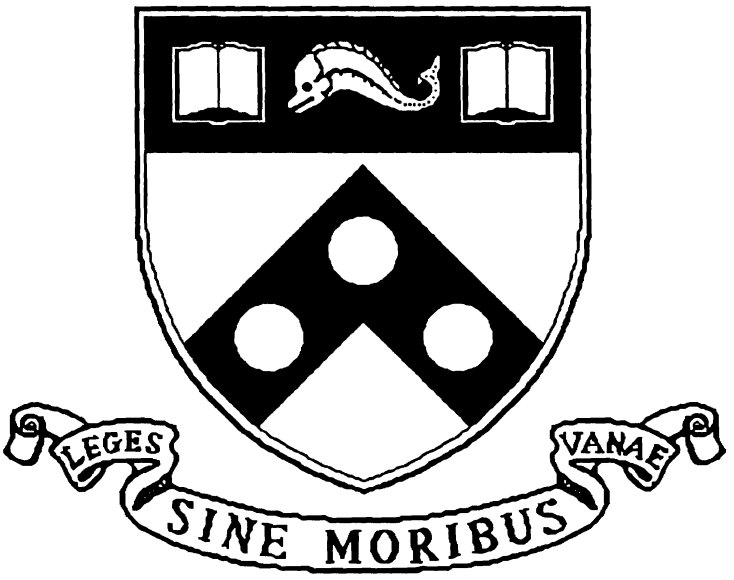
Robust Hypothesis Testing and Statistical Color Classification (Dissertation)

MS-CIS-93-90
GRASP LAB 365

Julie A. Adams



University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

November 1993

UNIVERSITY OF PENNSYLVANIA
THE MOORE SCHOOL OF ELECTRICAL ENGINEERING
SCHOOL OF ENGINEERING AND APPLIED SCIENCE

ROBUST HYPOTHESIS TESTING AND STATISTICAL COLOR CLASSIFICATION

Julie A. Adams

Philadelphia, Pennsylvania
May, 1993

A thesis presented to the Faculty of Engineering and Applied Science of the University of Pennsylvania in partial fulfillment of the requirements for the degree of Master of Science in Engineering for graduate work in Computer and Information Science.

Max Mintz
(Advisor)

Mitch Marcus
(Graduate Group Chair)

Abstract

Robust Hypothesis Testing and Statistical Color Classification

Julie A. Adams

Max Mintz, Advisor

The purpose of this research is twofold: (*i*) the development of a mathematical model for statistical color classification; and (*ii*) the testing of this model under controlled conditions.

We consider the following hypothesis testing problem: Let $Z = \theta + V$, where the scalar random variable Z denotes the sampling model, $\theta \in \Omega$ is a location parameter, $\Omega \subset R$, and V is additive noise with cumulative distribution function F. We assume F is uncertain, i.e., $F \in \mathcal{F}$, where $\mathcal{F}$ denotes a given uncertainty class of absolutely continuous distributions with a parametric or semiparametric description. The null hypothesis is $H_0 : \theta \in \Omega,\ F \in \mathcal{F}$ and the alternative hypothesis is $H_1 : \theta \notin \Omega,\ F \in \mathcal{F}$.

Through controlled testing we show that this model may be used to statistically classify colors. The color spectrum we use in these experiments is the Munsell color system which combines the three qualities of color sensation: Hue, Chroma and Value. The experiments show: (*i*) The statistical model can be used to classify colors in the Munsell color system; (*ii*) more robust results are achieved by using a Chroma-Hue match instead of a Perfect match; (*iii*) additional robustness can be achieved by classifying a color based on measurements averaged over a neighborhood of pixels verses measurements at a single pixel; and (*iv*) a larger color spectrum than the Munsell color system is needed to classify a range of man-made and natural objects.

Acknowledgements

I would like to thank my advisor Professor Max Mintz for his patience, guidance, insight, encouragement and enthusiasm.

I would also like to thank: The Multiagents group for their patience during my time of split responsibilities; Dr. Gerda Kamberova for her friendship, encouragement and LaTeX figure skills; and all the members of the GRASP Lab for their friendship and their ability to make life more colorful.[1]

[1]Portions of this research were supported by grants:
Navy Grants N00014-92-J-1647, NOOO14-92-J-1647; AFOSR Grant 88-0296; Army/DAAL 03-89-C-0031PRI; NSF Grants CISE/CDA 88-22719, IRI 89-06770, and ASC 91 0813; and the Du Pont Corporation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The purpose of this research is twofold: (*i*) the development of a mathematical model for statistical color classification; and (*ii*) the testing of this model under controlled conditions.

1.1 Mathematical Modeling Problem

We consider the following hypothesis testing problem: Let $Z = \theta + V$, where the scalar random variable Z denotes the sampling model, $\theta \in \Omega$ is the location parameter, $\Omega \subset R$, and V is additive noise with cumulative distribution function (CDF) F. We assume F is uncertain, i.e., $F \in \mathcal{F}$, where $\mathcal{F}$ denotes a given uncertainty class of absolutely continuous distributions with a parametric or semiparametric description. For example: $\mathcal{F} = \{N[0, \sigma^2] : \sigma_1 \leq \sigma \leq \sigma_2\}$, or $\mathcal{F} = \{F : F_1(x) \leq F(x) \leq F_2(x), x \in R\}$, where F_1 and F_2 are given bounding distributions. The null hypothesis is $H_0 : \theta \in \Omega, F \in \mathcal{F}$ and the alternative hypothesis is $H_1 : \theta \notin \Omega, F \in \mathcal{F}$.

To introduce these ideas, we begin with an example taken from [4] where F is known exactly: Let $H_0 : \theta_1 \leq \theta \leq \theta_2$ and $H_1 : \theta_1 > \theta$ or $\theta < \theta_2$, $Z \sim N(\theta, \sigma^2)$, and the critical (rejection) region $C = (-\infty, C_1) \cup (C_2, \infty)$. Then, if $Z < C_1$ or $Z > C_2$, we reject H_0.

We extend this previous example to the case where the noise distribution is not known exactly. Let $Z = \theta + \tilde{V} + \eta, \mid \eta \mid \leq w$ where η represents an uncertain shift or offset in the noise distribution. The distribution F of $\tilde{V}$ is assumed to be $N(0,1)$ and $\mid \theta \mid \leq \theta_0$.

The uncertain shift, η, represents a nuisance parameter. The effective noise V becomes $\tilde{V} + \eta$. Thus, $\mathcal{F}$ is $\{N[\eta, 1] : \mid \eta \mid \leq w\}$. The presence of η makes this a robust testing problem where the null hypothesis is $H_0 : \mid \theta \mid \leq \theta_0, F \in \mathcal{F}$ and the alternative hypothesis is $H_1 : \mid \theta \mid > \theta_0, F \in \mathcal{F}$. Here, robustness refers to the need to contend with uncertainty in the CDF F.

We present the development of the initial model and the expanded model in Chapter 2. This model provides an explanation for the experimental results: namely, that we are able to differentiate the chroma and hue of a color but will have difficulties differentiating between values within a chroma and hue.

1.2 Experimental Problem

Through controlled testing we show that this model may be used to statistically classify colors. The color spectrum we use in these experiments is the Munsell color system which combines the three qualities of color sensation: Hue, Chroma and Value.

The first step is to create a database of the spherical coordinates, (θ, ϕ), which represent the chroma and hue for a given color sample. This is accomplished by placing all the color chips for a certain hue onto the Munsell card, adding proper illumination, digitizing the image, and calculating the coordinates. This information is then used for comparison when classifying other images.

The first two experiments are aimed at testing the hypothesis: We can classify colors based on the spherical coordinates of a pixel color or an average of these coordinates over a set of pixels in an image.

Experiment I involves taking three images of the color chips on their original Munsell cards: (*i*) trying to recreate the base data, (*ii*) changing the aperture of the camera and (*iii*) changing the position of the extra illumination. For this experiment we must choose thresholds for (θ, ϕ). Based on the information from the statistical model building and some preliminary tests, we made the classification region for a given color representation (θ, ϕ) to be $(\theta \pm 2.0°, \phi \pm 2.0°)$. The results show we can classify colors using the spherical

coordinates and give us the actual thresholds we should use for the spherical coordinates: ($\theta \pm 2.0°$, $\phi \pm 3.0°$).

Experiment II uses the same images as Experiment I, but we are testing the hypothesis that the average of the spherical coordinates over a number of pixels will give better results than the classification of the spherical coordinates for a single pixel. This experiment provides strong evidence in support of this hypothesis correct.

Throughout both experiments we also compare the results of two types of matches: (*i*) A **Perfect match** is a match where the exact chroma, hue and value of the color are found; and (*ii*) A **Chroma-Hue match** is a match where the correct chroma and hue are found but the value is incorrect. The results from the statistical model show we should have difficulty differentiating between values for a given chroma and hue. This result is shown in the outcomes of both experiments. The number of matches attributed to the Chroma-Hue match are consistently higher than those attributed to the Perfect match.

Experiment III shows that this model can be used to classify colors of real objects. For this experiment we took images of different objects with differing colors, materials and textures. Each set of images is taken with varying light and aperture conditions. The results of this experiment is the knowledge that colors of real objects may be classified using this model as long as the color is within the database color spectrum, the Munsell color system.

The results of all experiments show that: (*i*) We can indeed classify colors of real objects based on the spherical coordinates; and (*ii*) In order for this model to be robust, the color spectrum must be much larger than the Munsell color system. We present the detail of the experimental set ups and results in Chapters 3 and 4, respectively.

Chapter 2

Statistical Model

Let $Z = \theta + V$, where the scalar random variable Z denotes the sampling model, $\theta \in \Omega$ is a location parameter, $\Omega = [-\theta_0, \theta_0]$, and V is additive noise with CDF F. We assume F may be uncertain, i.e., $F \in \mathcal{F}$, where $\mathcal{F}$ denotes a given uncertainty class of absolutely continuous distributions with a parametric or semiparametric description. Specifically, we consider three cases: (1) $\mathcal{F} = \{N[0,1]\}$, i.e., F is known completely; (2) $\mathcal{F} = \{N[0,\sigma^2] : \sigma_1 \leq \sigma \leq \sigma_2\}$, i.e., F is known up to a scale parameter which lies in a given interval; (3) $\mathcal{F} = \{N[\eta,1] : \mid \eta \mid \leq w\}$, i.e., F is known up to a location parameter which lies in a given interval. The null hypothesis is $H_0 : \theta \in \Omega, F \in \mathcal{F}$ and the alternative hypothesis is $H_1 : \theta \notin \Omega, F \in \mathcal{F}$.

Since the underlying uncertainty classes and parameter sets have inherent symmetry, we employ the following symmetric two-sided test: Reject H_0 if $|Z| > c$. In the sequel, we examine the power function for this test under cases (1-3) for $\mathcal{F}$.

2.1 Case 1:

Let $G(x)$ denote the CDF of the $N[0,1]$ distribution. Let $\Pi_1(\theta, c)$ denote the power function of the test:

$\Pi_1(\theta, c) = Pr[|Z| > c|\theta] = G(\theta - c) + G(-\theta - c)$.

We select the threshold parameter c based on: (i) the value of θ_0, and (ii) the desired

size of the test, i.e., the maximum probability of rejecting H_0 when H_0 is true [2, 8]. In this case, the threshold parameter c is selected by solving:

$\alpha(\theta_0, c) = G(\theta_0 - c) + G(-\theta_0 - c) = \alpha_0$,

where α_0 is the desired size of the test. For the purpose of this example, we let $\theta_0 = 5$, and select $\alpha_0 = 0.05$. The corresponding value of c is 6.64485.

The power function $\Pi_1(\theta, c)$ for this test and these parameters is plotted in Figure 1.

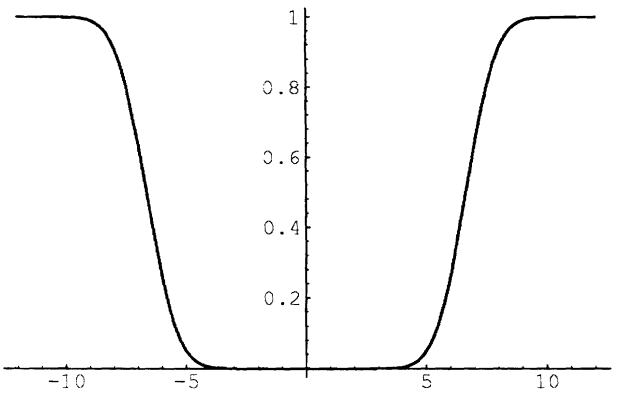


Figure 1: $\Pi_1[\theta.6.64485]$, $|\theta| \leq 12$

This example provides us with the base model for the sequel. We examine the behavior of the power function of the test parametrically in the context of the uncertainty classes $\mathcal{F}$ in cases (2-3).

2.2 Case 2:

Here, $\mathcal{F} = \{N[0, \sigma^2] : \sigma_1 \leq \sigma \leq \sigma_2\}$, i.e., F is known up to a scale parameter which lies in a given interval. Let $\Pi_2(\theta, \sigma, c)$ denote the power function of the test:

$\Pi_2(\theta, \sigma, c) = Pr[|Z| > c|\theta, \sigma] = G((\theta - c)/\sigma) + G((-\theta - c)/\sigma)$.

Figure 2 depicts $\Pi_2(\theta, \sigma, c)$ as a function of θ for the following fixed values of σ: $\{0.5, 1.0, 2.0\}$, based on a threshold: $c = 6.64485$.

The power functions in Figure 2 are keyed by dash-size: small ($\sigma = 1.0$), medium ($\sigma = 0.5$), large ($\sigma = 2.0$).

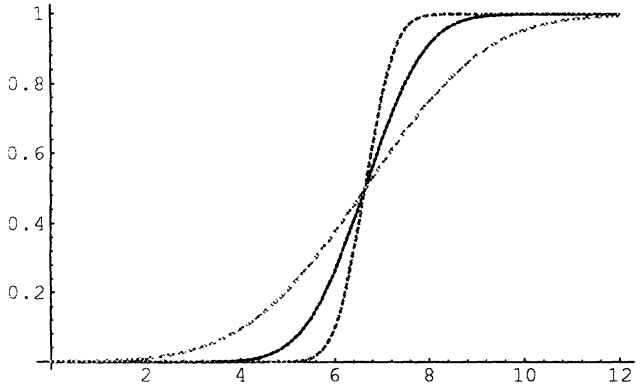


Figure 2: $\Pi_2[\theta, \sigma, 6.64485]$, $0 \leq \theta \leq 12, \sigma \in \{0.5, 1.0, 2.0\}$

Figure 3 depicts $\Pi_2(\theta, \sigma, c)$ as a function of θ for the following fixed values of σ: $\{0.5, 1.0, 2.0\}$, based on a threshold: $c = 5.82243$. This threshold corresponds to a size 0.05 test when $\sigma = 0.5$.

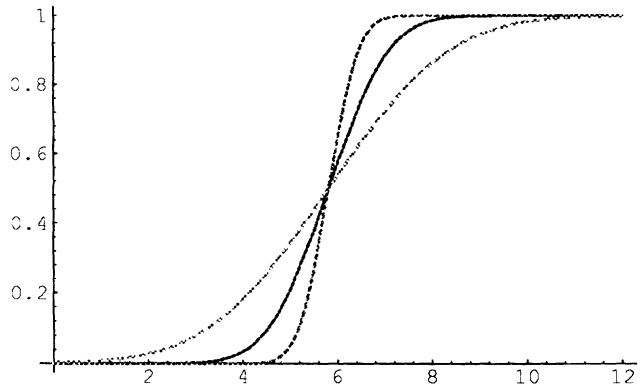


Figure 3: $\Pi_2[\theta, \sigma, 5.82243]$, $0 \leq \theta \leq 12, \sigma \in \{0.5, 1.0, 2.0\}$

The power functions in Figure 3 are keyed by dash-size: small ($\sigma = 1.0$), medium ($\sigma = 0.5$), large ($\sigma = 2.0$).

Figure 4 depicts $\Pi_2(\theta, \sigma, c)$ as a function of θ for the following fixed values of σ: $\{0.5, 1.0, 2.0\}$, based on a threshold: $c = 8.28971$. This threshold corresponds to a size 0.05 test when $\sigma = 2.0$.

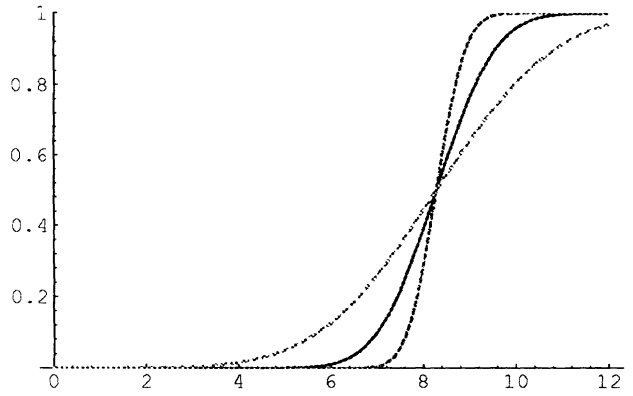


Figure 4: $\Pi_2[\theta, \sigma, 8.28971]$, $0 \leq \theta \leq 12, \sigma \in \{0.5, 1.0, 2.0\}$

The power functions in Figure 4 are keyed by dash-size: small ($\sigma = 1.0$), medium ($\sigma = 0.5$), large ($\sigma = 2.0$).

Figure 5 depicts $\Pi_2(\theta, 1.0, c)$ as a function of θ for $c \in \{6.64485, 5.82243, 8.28971\}$.

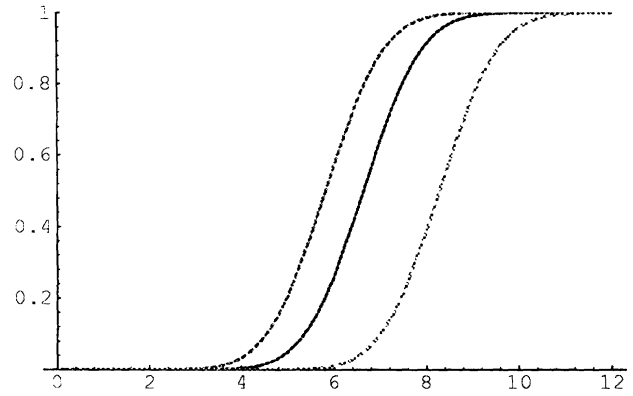


Figure 5: $\Pi_2[\theta, 1.0, c]$, $0 \leq \theta \leq 12, c \in \{6.64485, 5.82243, 8.28971\}$

The power functions in Figure 5 are keyed by dash-size: small ($c = 6.64485$), medium ($c = 5.82243$). large ($c = 8.28971$).

We conclude from this analysis, that when σ is uncertain, i.e., $\sigma_1 \leq \sigma \leq \sigma_2$, then we can design the test based on the upper-value σ_2. This approach is feasible when $(\sigma_2 - \sigma_1)/\sigma_2$ is small.

2.3 Case 3:

Here, $\mathcal{F} = \{N[\eta, 1] :| \eta | \leq w\}$, i.e., F is known up to a location parameter which lies in a given interval. We adopt a partial Bayesian approach to analyzing this case. We assume the parameter η can be modeled by a random variable with a uniform distribution on $[-w, w]$. Thus, the effective noise density becomes:

$h(x, w) = (G(x + w) - G(x - w))/2w.$

The CDF $H(x, w)$ is obtained by numerical integration. Figure 6 depicts the CDF $H(x, w), x \geq 0$ for $w \in \{0.005, 0.05, 0.1\}$. Figure 7 depicts the CDF $H(x, w), x \geq 0$ for $w \in \{0.05, 0.5, 0.99\}$. We observe that as the value of w increases, the CDF H tends to flatten. As w increases, the deviation of H from $N[0, 1]$ becomes more pronounced. We conclude from this analysis, that when η is uncertain, i.e., $|\eta| \leq w$, then we can design the w-dependent test based on the CDF $H(x, w)$. This approach is feasible when w is small in comparison with θ_0.

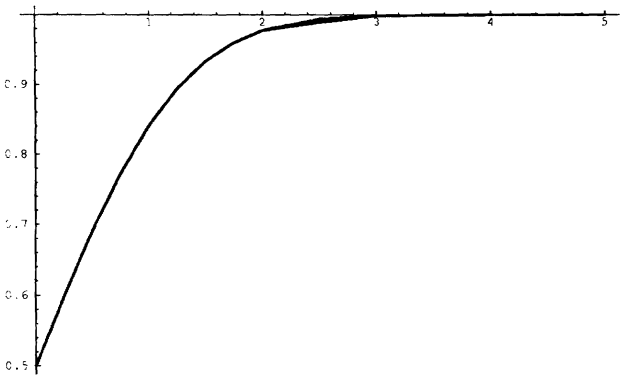


Figure 6: The CDF $H(x, w), w \in \{0.005, 0.05, 0.01\}$

Based on the foregoing, we conclude, we can classify the colors of the Munsell color system based on the mean and variance of the normalized unit color vectors. We next discuss the experimental test this hypothesis in Chapters 3 and 4.

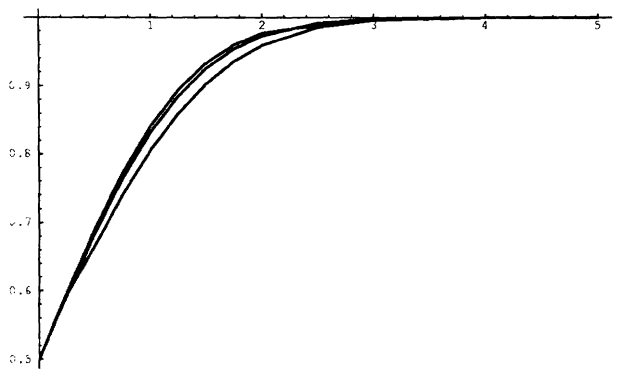


Figure 7: The CDF $H(x, w), w \in \{0.05, 0.5, 0.99\}$

Chapter 3

Experiment

3.1 Set Up

The experiment set up includes a Sony XC-77/77CE CCD B/W video camera module with a gamma factor of 1. A 50 mm Nikon lens and a set of Kodak color filters, including no. 25 red, no. 58 green and no. 47 blue. In addition, a complete set of Munsell student color charts. Extra incandescent and fluorescent illumination devices are employed. See Figure 8.

3.2 Base Information

The chips are properly arranged into the Munsell order of value, chroma and hue on the cards provided with the color system. Then extra illumination is added because the ordinary overhead illumination is insufficient. Pictures are digitized using an 8-bit digitizer[1] for the Sun 4 machine.

Each hue is taken one at a time. The extra illumination is directed at the center of the card so there is approximately equal illumination over the entire set of chips for a single hue. The camera aperture is adjusted for calibration of the system for each hue, such that the brightest chip has an intensity value close to, if not exactly, 225 with the Red filter.

[1]The intensity values for this digitizer range from 0 to 255.

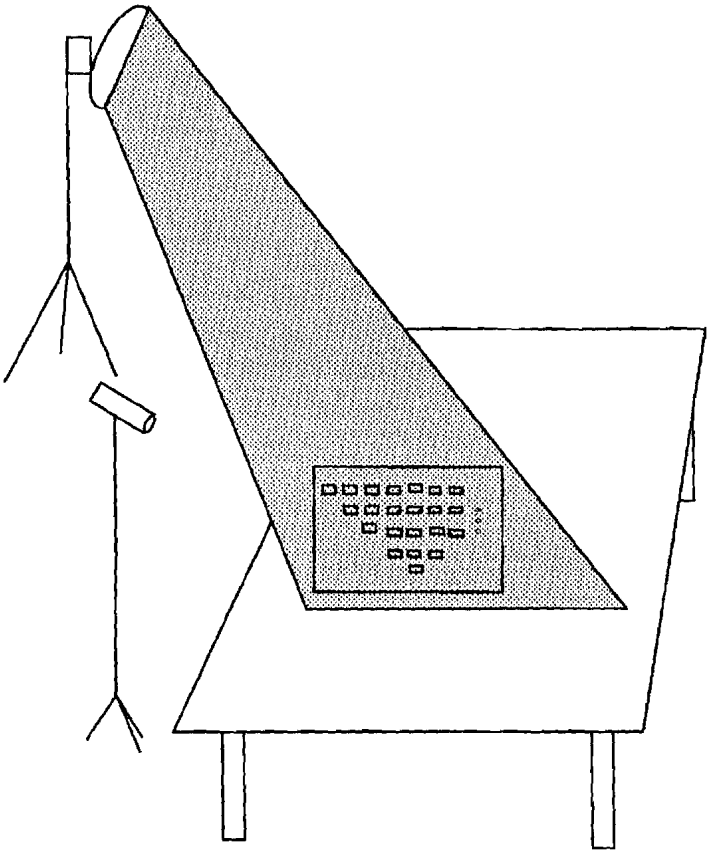


Figure 8: Initial Set Up

This intensity value is chosen to keep the blue pixel intensity values from being lower than 30. The intensity values should not be higher than 225 to prevent oversaturation.

After we digitize each image, we partition the image into individual chips. The software to manipulate the digitized images is PM[2]. This is used to cut the images such that each individual chip is put into its own set of files. Each chip is cut into a 32x32 pixel image which is centered at the center of the particular chip in the original image. This creates 708 individual files, three files, representing the red, green and blue components for each of the 236 color chips of the Munsell system.

After each color chip is set up into its own set of files, containing the red, green and

[2] "PM is a group of programs to be executed in the UNIX shell, and is also a collection of C functions to be used by programmers doing work in image processing."[12]

blue components, the three files are scaled to unit length using the normalization program. This program reads in each component file for a chip, namely the red, green and blue. The scale for the filters plus the camera are: red = 2.1875137; green = 3.0927197 and blue = 4.5181403. Each pixel from the original image is read into the program. As they are read, they are multiplied by the scale values. The values of each pixel are then normalized:

$$normalize = \sqrt{r^2 + g^2 + b^2}$$

Then each component is retrieved by dividing the scaled component by the normalize value. It is then stored into a file with the .rgb extension which is used by the mean program.

The normalized information, in the .rgb file, is then used to find the spherical coordinates, referred to as (θ, ϕ) in the mean program. This program takes each normalized component of each pixel and accumulates a sum for (θ, ϕ):

$$sum\theta = sum\theta + \arccos(b)$$

$$sum\phi = sum\phi + \arctan(g \div r)$$

After all pixels have been processed the sums of (θ, ϕ) are divided by the number of pixels in the image, this results in the final values of (θ, ϕ). This information is combined with the actual chip name, i.e., 5-8r, and is used as the base data of the classification program. See Appendix B for the actual C program code.

3.3 Classification Program

This program is used to calculate the spherical coordinates, (θ, ϕ), for a specific pixel or a neighborhood of pixels. This information is used to search the base data file for a match. All chip names and spherical coordinates which are within a certain threshold range are printed out. If the color which is being classified is one of the names printed out by the program we have a match.

Based on the discussion in Chapter 2 Section 3, we show that it is difficult to distinguish between values within a chroma and hue because the value information is lost due to normalization. This means that there are two types of matches we will discuss in the results.

A **Perfect match** is a match where the exact color being classified is found. For example, if a pixel of a 5-8r chip is being classified and the results include 5-8r, the value, chroma and hue all match.

A **Chroma-Hue match** is a match where the results include the correct chroma and hue but not the correct value. For example, if a pixel of a 5-8r chip is being classified and the results include 5-2r, 5-4r, 5-6r, 5-10r, 5-12r or 5-14r but not 5-8r, then the chroma and hue match, but not the value.

Refer to Appendix B for the code of the classification program and all other programs used in this system.

Chapter 4

Results

4.1 Experiment I

The first experiment takes random pictures of the Munsell color chips. This is accomplished by setting up the chips in their proper order on the Munsell color cards. Lighting and aperture are then varied.

The first set of images are taken using an extra incandescent light directed at the center of the card. The aperture of the camera is open such that the brightest chip has about the same intensity value it had in the base data, an intensity value of approximately 225 for the image taken with the red filter. This set of images are meant to duplicate the base data and the set up is similar to Figure 8.

The second set of images are taken with the same conditions as the first set except that the aperture on the camera is closed slightly. This causes the intensity values of the brightest chip to be lower by an intensity value of approximately 25 for images taken with the red filter.

The third set of images are taken with indirect lighting and a wide enough aperture to make the brightest chip have intensity values as close to the 225 value as possible for the image taken with the red filter. The extra lighting in this case is to the right side of the Munsell card and is not directed to the center of the card but off to the top of the card.

In all three experiments the actual Munsell card remains in the same position, as does the camera. The only equipment which moves throughout the experiments is the incandescent lamp.

These images are then normalized using the normalize program which was used on the base data. Pixels are then chosen from each chip and are input to the classification program. This program requires threshold values for (θ, ϕ) be chosen ahead of time and are initially chosen to be $(\theta \pm 2.0°, \phi \pm 2.0°)$. We base this choice on the knowledge the values should be similar to the original.

The initial run is used to create the proper threshold values and to test the correctness of the hypothesis. This experiment shows the threshold of $(\theta \pm 2.0°)$ for gives an accuracy of 92.08%, and the same threshold for ϕ gives an accuracy of 81.72%.

Figure 9 shows the histogram of the θ values after taking the absolute values of the difference between the θ value calculated by the classification program and the θ value in the base data. Figure 10 shows the histogram of the same data, less 4 outlying points, before the absolute values are taken. Figure 10 plots 99.81% of the data points. Figure 11 plots the histogram of the ϕ values after taking the absolute values of the difference between the ϕ value calculated by the classification program and the ϕ value in the base data. Figure 12 shows the histogram of the same data, less 26 outlying points, before the absolute values are taken. This figure plots 98.78% of the data points.

This information is used to find the threshold values needed to obtain at least a 90% accuracy for both (θ, ϕ). Figures 9 and 10 show the θ threshold should be $(\theta \pm 2.0°)$ which gives a 92.08% accuracy. Figure 11 and 12 show the ϕ threshold should be $(\phi \pm 3.0°)$ which gives a 93.01% accuracy.

The experiment also shows illumination effects the results. The first two sets of images with direct illumination give significantly better results than the third set of results when indirect illumination is used, Table 1. This is an expected result, since the chip vector values will decrease when illumination is changed significantly.

The experiment shows aperture effects the classification but the effect is insignificant compared to that of illumination. It is found when the aperture is changed by a significant

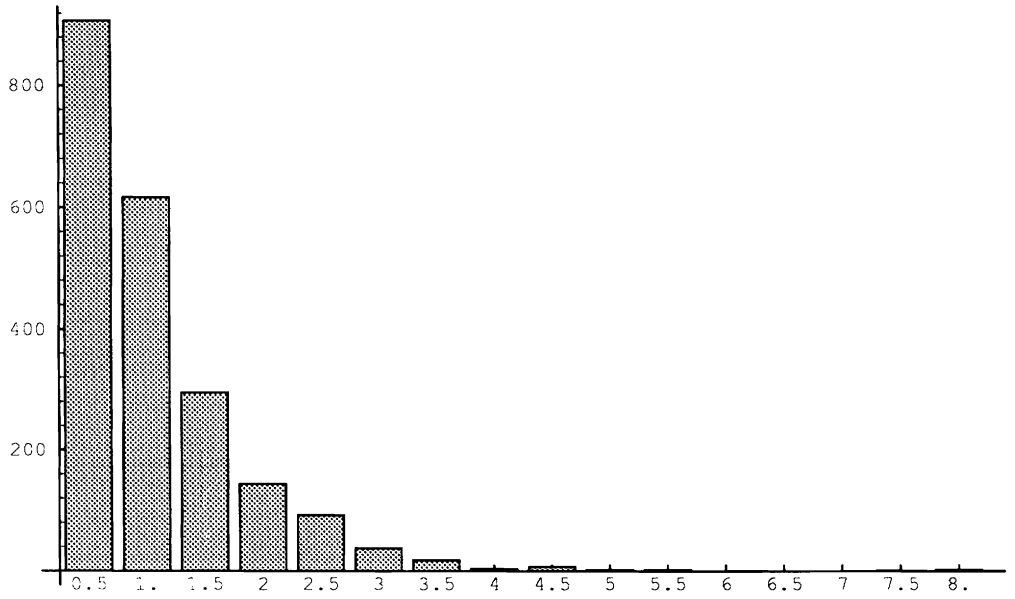


Figure 9: θ

Set	Hit	Total	Percentage
1	662	712	92.98
2	635	712	89.19
3	373	712	52.39
Total	1493	2136	69.90

Table 1: All Three Sets

degree, the results can be much lower. This is expected since by lowering the aperture we are not allowing as much illumination to enter the system. This is seen in the results of the second set of images in Table 1.

The above information is relevant to a **Perfect match**, the results of the **Chroma-Hue match** must be considered. Table 2 shows a comparison of the total number of matches for a Perfect match verses a Chroma-Hue match. This information further supports the hypothesis that it is very difficult to distinguish between values of a chroma and hue based on normalized data. The Chroma-Hue match results are 21.02% better.

The conclusions drawn from this particular experiment:

1 - It is feasible to use this type of a model for color classification;

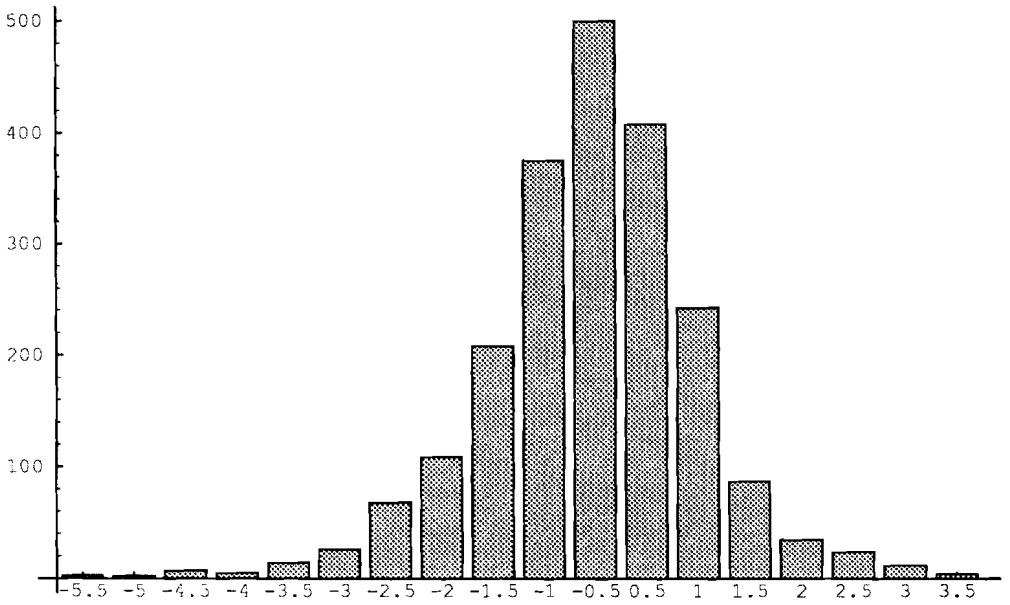


Figure 10: θ with Positive and Negative Values

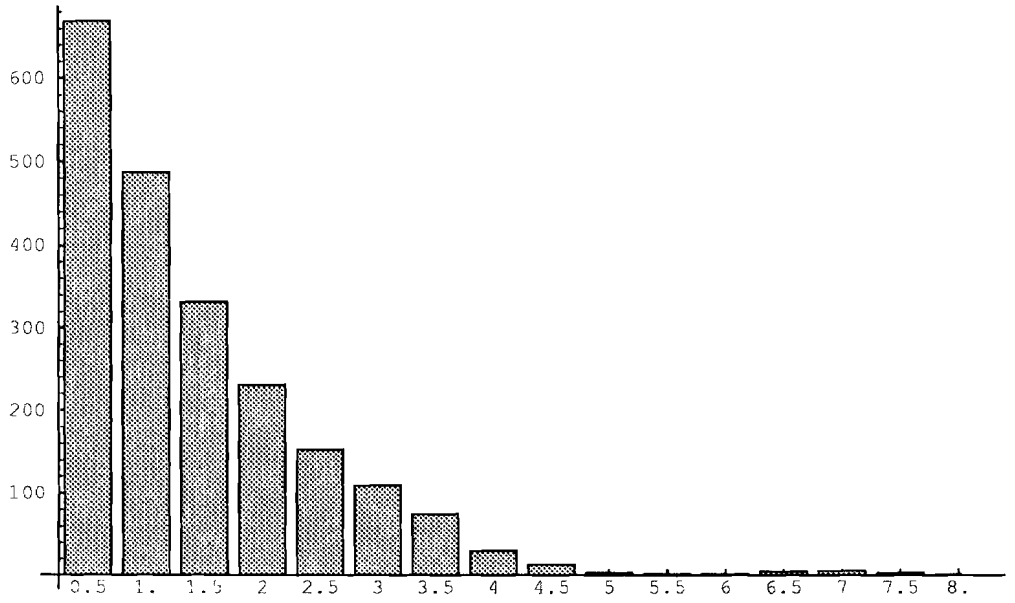


Figure 11: ϕ

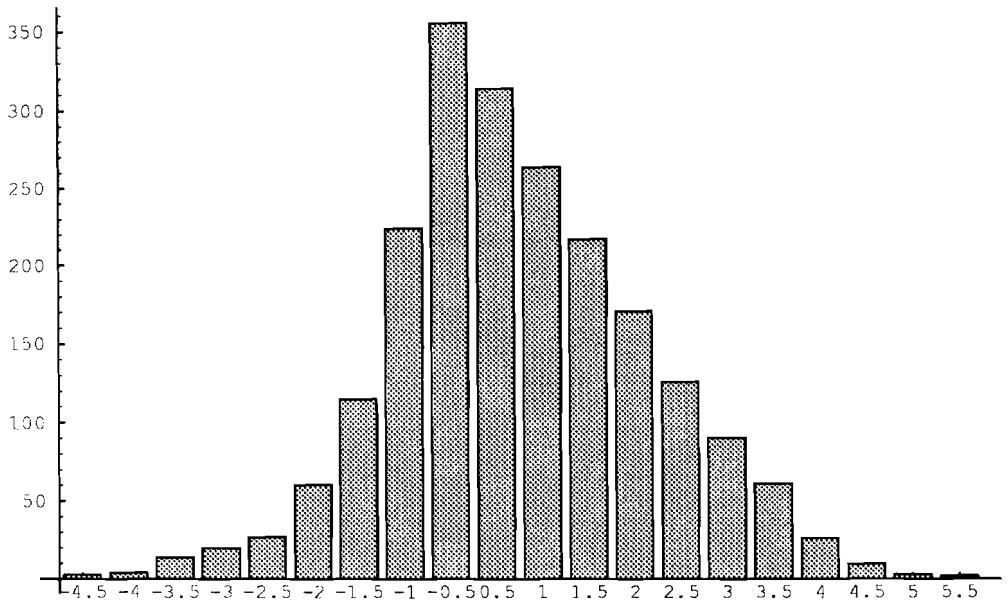


Figure 12: ϕ with Positive and Negative Values

Match Type	Hit	Total	Percentage
Perfect	1493	2136	69.90
Chroma-Hue	1942	2136	90.92
Difference	449		21.02

Table 2: Perfect Match vs. Chroma-Hue Match of All Three Image Sets

2 - The threshold values of (θ, ϕ) should be $(\theta \pm 2.0°, \phi \pm 3.0°)$;

3 - Illumination is a significant factor in the classification;

4 - Aperture can effect the classification when it is significantly closed.

4.2 Experiment II

This experiment utilizes the same information as Experiment I but we are now using the average of (θ, ϕ) over a neighborhood of 9 pixels, around the pixel used in Experiment I, to create the classification results. The purpose of this experiment is to verify the hypothesis that the averaging of the spherical coordinates over a neighborhood of pixels should give better results than the spherical coordinates of a single pixel. The thresholds for the

averaged (θ, ϕ) are $(\theta \pm 2.0°, \phi \pm 3.0°)$.

Figure 13 shows the histogram of the θ values after taking the absolute values of the difference between the θ value calculated by the classification program and the θ value in the base data. Figure 14 shows the histogram of the same data, less 6 outlying points, before the absolute values are taken. Figure 14 plots 99.72% of the data points. Figure 15 the histogram of the ϕ values after taking the absolute values of the difference between the ϕ value calculated by the classification program and the ϕ value in the base data. Figure 16 shows the histogram of the same data, less 38 outlying points, before the absolute values are taken. This figure plots 98.22% of the data points.

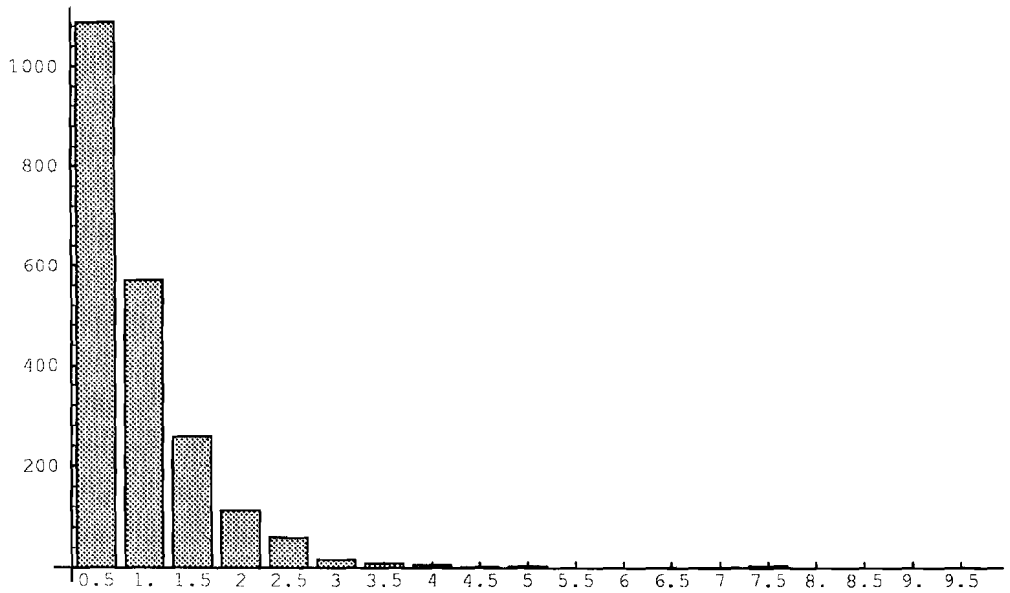


Figure 13: Average θ

The four figures show the results are indeed better. The θ threshold of $(\theta \pm 2.0°)$ gives a 95.27% accuracy, an improvement of 3.19%. The ϕ threshold of $(\phi \pm 3.0°)$ gives a 93.16% accuracy, an improvement of 0.15%. A future experiment could include testing smaller and larger neighborhoods to find the neighborhood size which gives optimal results.

This experiment also shows improved results can be achieved for images when indirect illumination is used. Table 3 shows a comparison of the results for classification of the

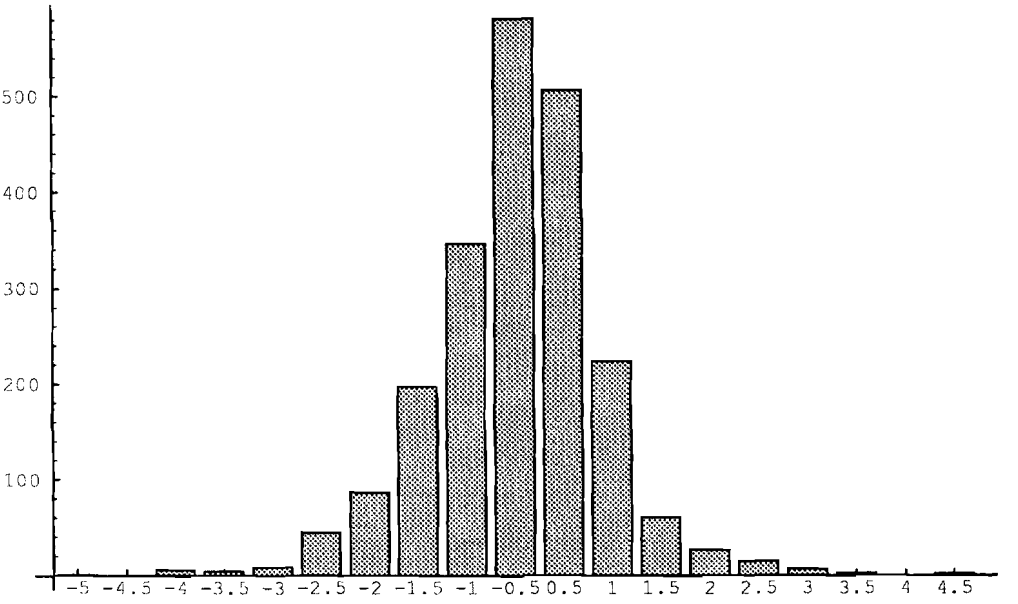


Figure 14: Average θ with Positive and Negative Values

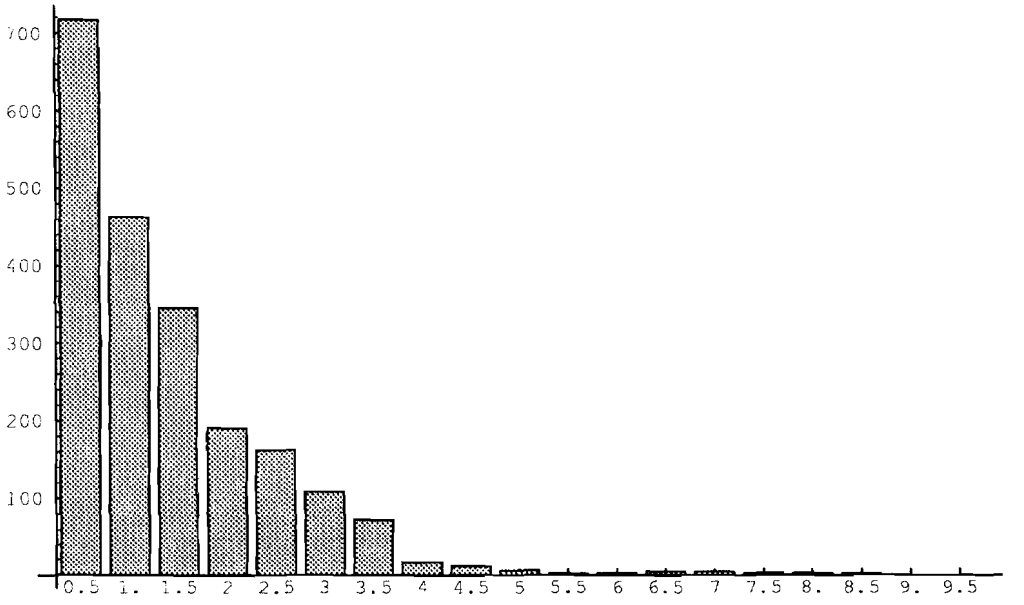


Figure 15: Average ϕ

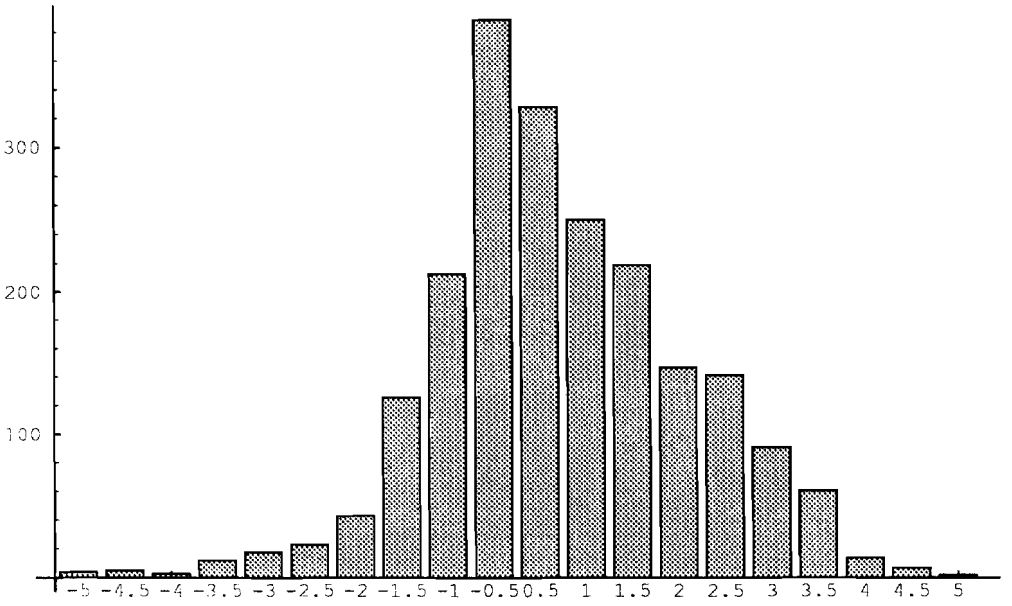


Figure 16: Average ϕ with Positive and Negative Values

third set of images using just a single pixel and the averaging method. Recall the third set of images is taken with indirect lighting. As can be seen from Table 3, the results are significantly improved, 30.76%, when the averaging method is used.

We show the difference in accuracy when using the Chroma-Hue matching between the single pixel method and the averaging method of classification, in Table 4. Once again the averaging method out performs the single pixel method, by 11.65%.

Table 5 shows the Chroma-Hue match is 30.48% more accurate for the single pixel method than the Perfect match. Table 6 demonstrates the Chroma-Hue match is 11.37% better than the perfect match when using the averaging method.

4.3 Experiment III

The third experiment involves classifying images of real objects as opposed to the Munsell color chips. The purpose of this experiment is to verify the model will work on real objects.

The first set of images, Figure 17, contains ten shiny plastic blocks and five wooden

Method	Hit	Total	Percentage
Single pixel	373	712	52.39
Averaging	592	712	83.15
Difference	219		30.76

Table 3: Single Pixel vs. Averaging for the Third Set of Images and a Perfect Match

Method	Hit	Total	Percentage
Single pixel	590	712	82.87
Averaging	673	712	94.52
Difference	83		11.65

Table 4: Single Pixel vs. Averaging for the Third Set of Images and a Chroma-Hue Match

Method	Hit	Total	Percentage
Perfect	373	712	52.39
Chroma-Hue	590	712	82.87
Difference	217		30.48

Table 5: Perfect match vs. Chroma-Hue match for the Single Pixel Method

Method	Hit	Total	Percentage
Perfect	592	712	83.15
Chroma-Hue	673	712	94.52
Difference	81		11.37

Table 6: Perfect match vs. Chroma-Hue match for the Averaging Method

blocks. Two different images are taken with differing illumination. The colors of the blocks include white, pink, yellow, blue, red, orange and green. The white, blue, pink and orange blocks are colors outside of the Munsell color system. The classifying program correctly classifies the yellow, red and green blocks. Once again illumination is found to cause a problem when not directly upon the objects.

The second set of images, Figure 18, includes a *Wheat Thins*^{tm} box and five shiny plastic balls. The balls are green, yellow, red, blue and pink, left to right in Figure 18. Three different images are taken with differing illumination. The green and blue balls are outside of the Munsell color system as are the blue letters on the box. The pink ball is consistently classified as red and red-purple. The tomato, green leaves above the tomato and the yellow of the box are classified correctly.

Other images include the *Mathematica*^{tm} conference posters from 1991 and 1992. Classification on these images are almost continuously correct. See the following copies of the images to see what portions of the posters we use. Figure 19 is the image from the 1991 poster. Figures 20 and 21 are the images taken from the 1992 poster. These portions are chosen for the number of different shades and hues in them.

This experiment shows this model can be used to classify real objects with colors within the Munsell color system, but will give incorrect or no results if the color is not within the color system. It also shows the material of the object; plastic, wood, cardboard or paper; does not necessarily effect the results.

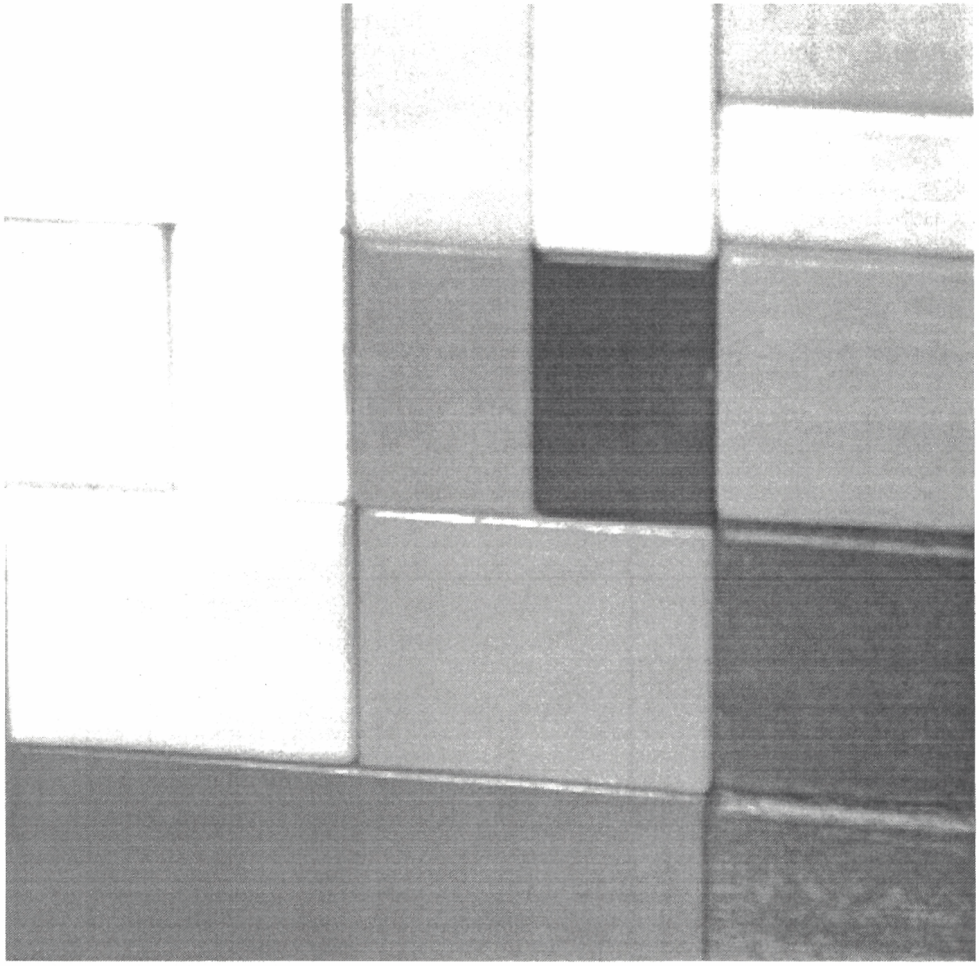


Figure 17: Image of assorted blocks with direct lighting

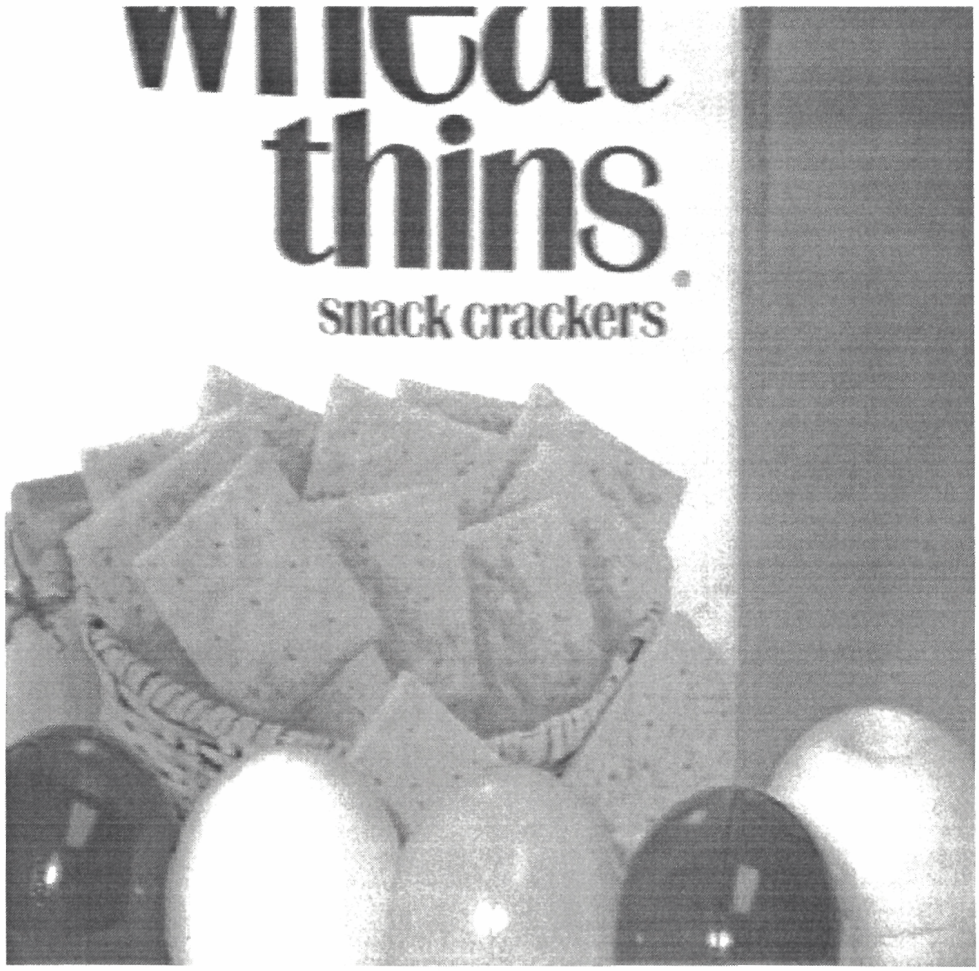


Figure 18: Image of *Wheat Thins*^{tm} box with direct lighting

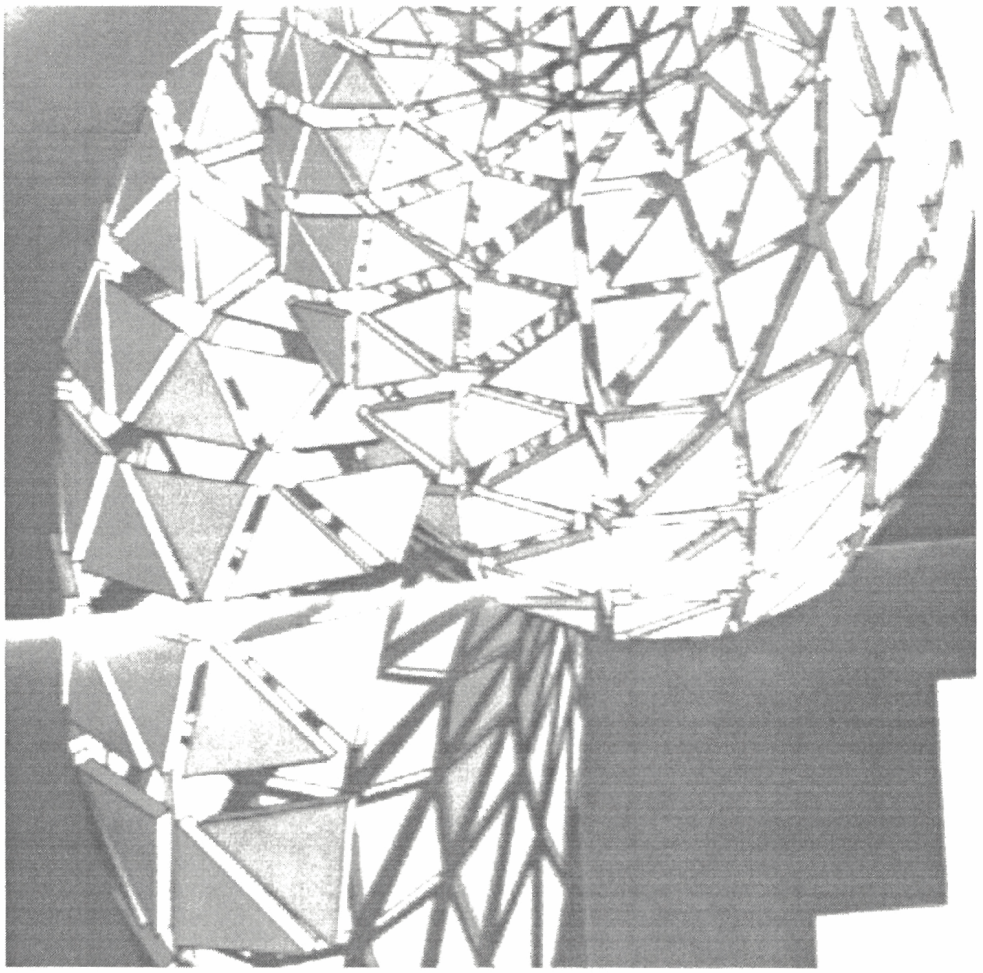


Figure 19: Image of 1991 *Mathematica*^{tm} Conference Poster

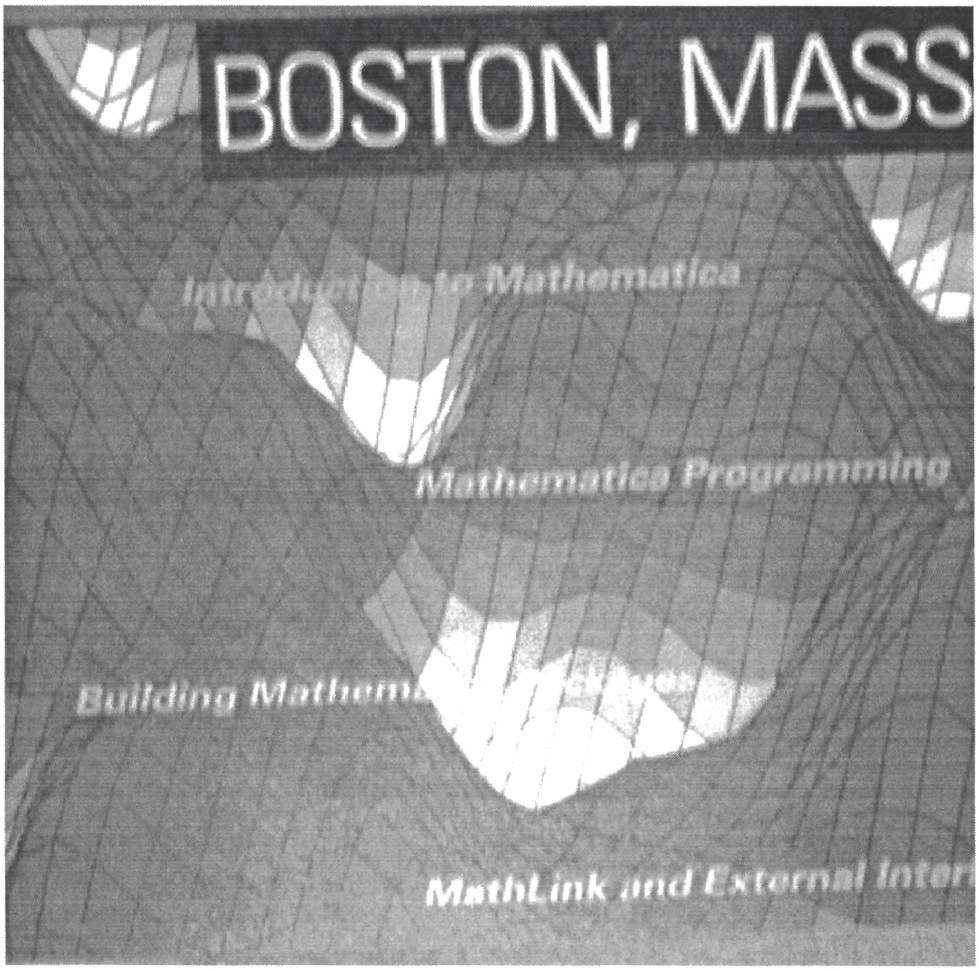


Figure 20: Image of a portion of the 1992 *Mathematica*^{tm} Conference Poster

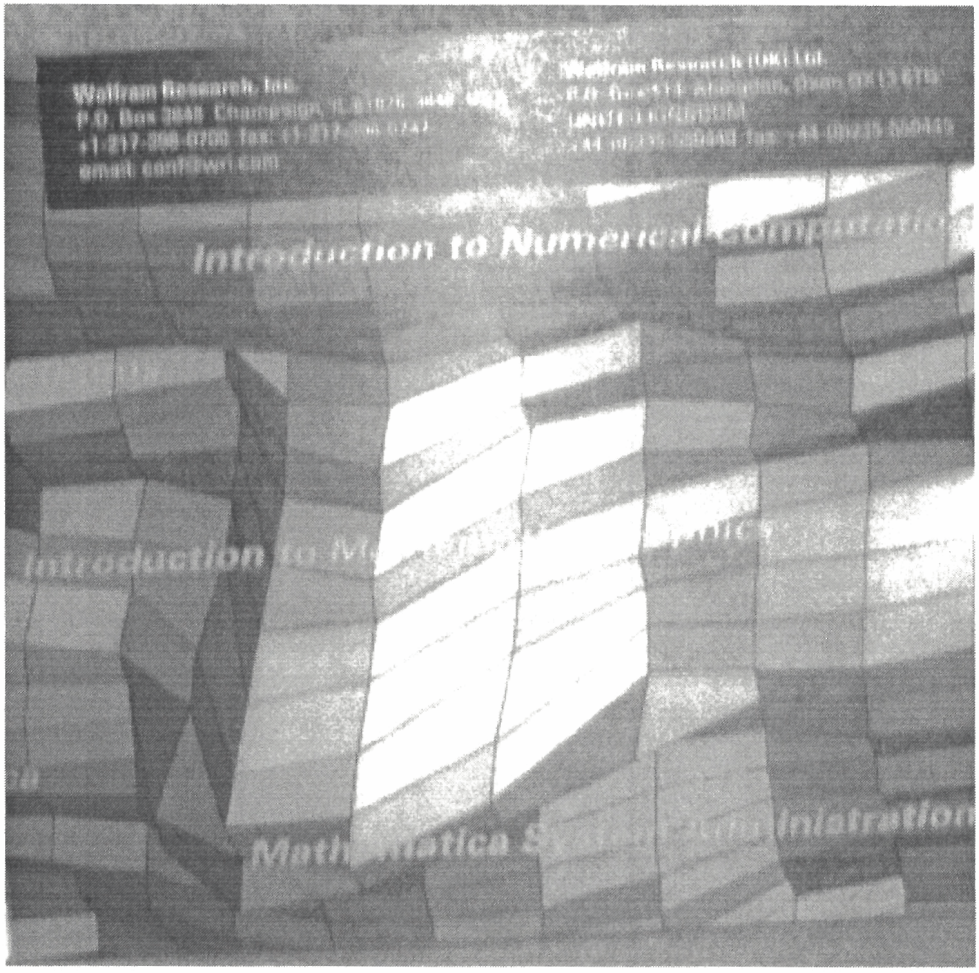


Figure 21: Image of another portion of the 1992 *Mathematica*^{tm} Conference Poster

Chapter 5

Conclusions and Future Research

5.1 Conclusions

We developed a statistical hypothesis testing model which incorporates uncertainty in the distribution of the observation noise. We begin with $Z = \theta + V$, where the scalar random variable Z denotes the sampling model, $\theta \in \Omega$ is a location parameter, $\Omega = [-\theta_0, \theta_0]$, and V is additive noise with CDF F. We assume F may be uncertain, i.e., $F \in \mathcal{F}$, where $\mathcal{F}$ denotes a given uncertainty class of absolutely continuous distributions with a parametric or semiparametric description. Specifically, we consider three cases: (1) $\mathcal{F} = \{N[0,1]\}$, i.e., F is known completely; (2) $\mathcal{F} = \{N[0,\sigma^2] : \sigma_1 \leq \sigma \leq \sigma_2\}$, i.e., F is known up to a scale parameter which lies in a given interval; (3) $\mathcal{F} = \{N[\eta,1] :\mid \eta \mid \leq w\}$, i.e., F is known up to a location parameter which lies in a given interval. The null hypothesis is $H_0 : \theta \in \Omega, F \in \mathcal{F}$ and the alternative hypothesis is $H_1 : \theta \notin \Omega, F \in \mathcal{F}$.

We are able to build a suitable threshold test for (θ, ϕ), but since the value parameter is normalized out, in the transformation to the spherical coordinates, the value information is lost. Therefore, we can distinguish between chroma and hue but will have difficulties distinguishing value.

The experiments show improved results can be obtained by taking the average of (θ, ϕ) over a neighborhood of pixels as opposed to using just the single pixel method. The spatial

averaging has the effect of reducing the observation noise and the volume of the underlying uncertainty class. The experiments also show the results can be improved based on the type of match required, Perfect verses Chroma-Hue.

Illumination effects the results, direct illumination is best, based on the base data where the original illumination is directed at the center of the hue card. Aperture can effect results if it is closed too much such that the intensity values of the image created with the blue filter are too low.

Overall this model is capable of being used for color segmentation as long as the base data includes a large enough color spectrum to classify all natural colors. This can not be accomplished with the Munsell color system.

5.2 Future Research

This paper is a study to see if the hypotheses is correct. Now further studies are need to improve the robustness of the model.

The effects of illumination need to be studied further to extract more accurate error bounds when: (*i*) using different sources of illumination, such as incandescent or fluorescent lighting; (*ii*) varying the position of the illumination; and (*iii*) varying the intensity of the illumination. This issue is related to the value discrimination problem. In order to test or estimate the value of a color, the following procedure could be tested: (*i*) Test for or estimate (θ, o), based on the normalized data. (*ii*) Take the unnormalized data and estimate the value parameter based on the estimated hue.

The effects of changing aperture should also be studied in detail. It is important to know the exact threshold where the classification becomes incorrect due to aperture.

Other color systems should be tested to find one which has a broad enough color spectrum to classify the most real objects. The actual model should need no changes for these tests. The purpose of this study should be to find a color system which is encompassing enough to work on as many real objects as possible.

Further study should also include different object materials. Studies of the effects of

shadows and reflections are needed. A study should be conducted to determine the proper neighborhood size which gives the best consistent classification results when using the averaging method.

Appendix A

The Munsell Color System

The primary experiments in this study are based on the Munsell color chart chips. This appendix describes what the Munsell color chart is and its values. This color chart was designed by A. H. Munsell. It combines the three qualities of color sensation: Hue, Chroma, and Value.

Hue refers to the name of the color [11], such as red, purple, green or blue. Value is the lightness of the color, or the whiteness in a color. The darker a color the lower its value and the more it tends towards black. The Chroma coincides with the deepness of the color. As the chroma increases the intensity of the color increases. The combination of the three characteristics give us the different shades of colors in the chart.

The purpose of this system is to classify color relations. This is how the colors like yellow-red and purple-blue are developed. Yellow and red are neighbors and the intermediary space between them represents the color yellow-red. The same is true with purple-blue, red-purple, green-yellow, etc. Figure 22 is a picture of the Munsell student color chart.

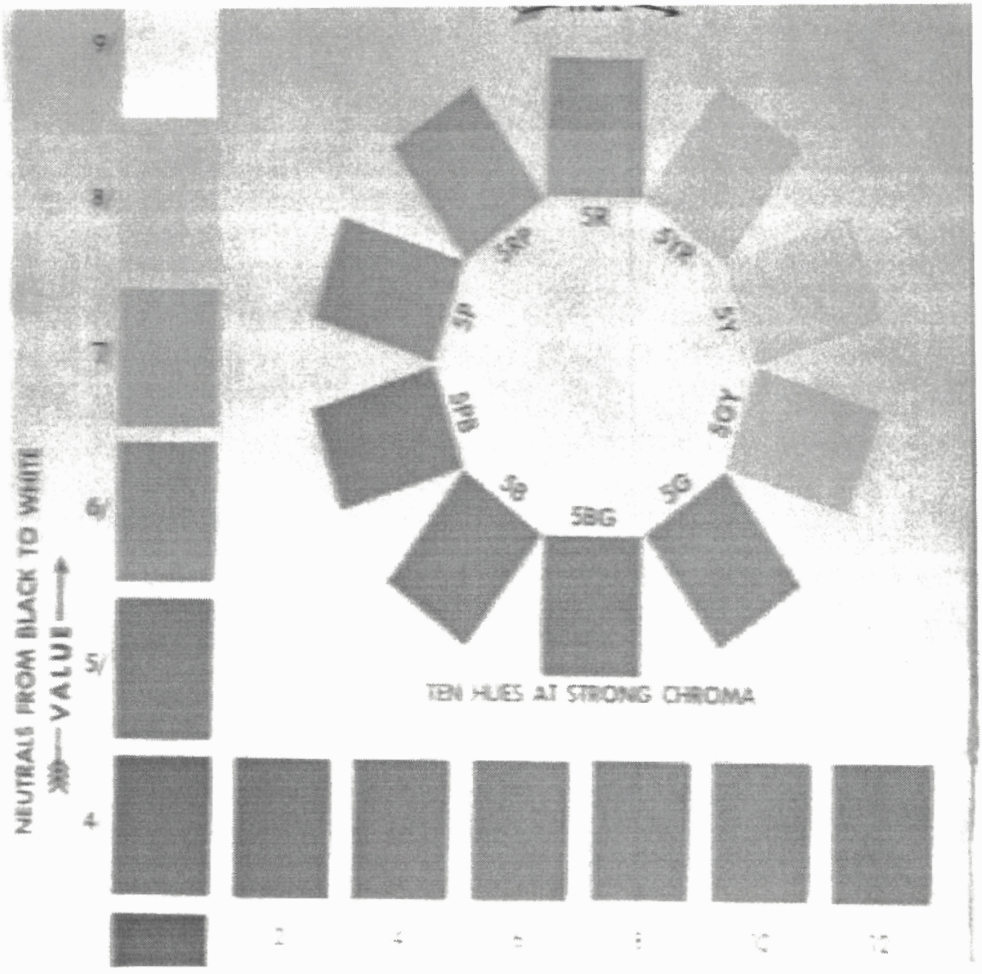


Figure 22: The Munsell Color Chart

Appendix B

Programs for Data Analysis

This Appendix delineates the complete programs which are used for the data analysis in this study.

B.1 Vector Normalization Program

This program was originally written by Jakov Kucan. It takes three files containing red, green and blue components and scales them to unit length. The files are named [name].r, [name].g and [name].b. If -s is used, the rgb components are scaled with RSCALE, GSCALE, and BSCALE read from a scale-file. If -d is used, the default scale values are assumed. If -n is used the PM_F file containing norms of rgb vectors is produced. The usage of this program is: **RGBnorm -s [scale-file] -d -n [name]**.

```
#include<stdio.h>
#include<math.h>
#include<local/pm.h>
#include<string.h>

#define TRUE 1
```

```
#define FALSE 0

#define NARGS 1

#define max(a,b) (((a) > (b)) ? (a) : (b))
#define min(a,b) (((a) < (b)) ? (a) : (b))

#define RSCALE 2.1875137   /* RED   filter #25 + camera */
#define GSCALE 3.0927197   /* GREEN filter #58 + camera */
#define BSCALE 4.5181403   /* BLUE  filter #47 + camera */

#define OPTIONS "s:dn"
extern int optind;
extern char *optarg;

main(argc,argv)
     int argc;
     char **argv;

{ int i;
  pmpic *Rpic, *Gpic, *Bpic;
  pmpic *result, *norm_pic;
  float *res, *norm_img;
  double r, g, b, norm, norm1;
  double r_scale, g_scale, b_scale;
  char fname[256];
  FILE *fd;
  int err, scale, do_norm;
  char *cmd, c;
```

```
err = 0;
scale = FALSE;
do_norm = FALSE;
cmd = argv[0];

while((c = getopt(argc,argv,OPTIONS)) != EOF)
  { switch(c)
      { case 's':
          if((fd = fopen(optarg,"r")) == NULL)
            { fprintf(stderr,"%s: Cannot open %s.\n",cmd,
                      optarg);
              scale = FALSE;
            }
          else
            { fscanf(fd,"%s",fname);
              r_scale = atof(fname);
              fscanf(fd,"%s",fname);
              g_scale = atof(fname);
              fscanf(fd,"%s",fname);
              b_scale = atof(fname);
              if(r_scale * g_scale * b_scale)
                scale = TRUE;
              else
                {fprintf(stderr,"%s: Scale is 0, ignored.\n",
                         cmd);
                  scale = FALSE;
                }
            }
```

```
          break;
        case 'd':
          r_scale = RSCALE;
          g_scale = GSCALE;
          b_scale = BSCALE;
          scale = TRUE;
          break;
        case 'n':
          do_norm = TRUE;
          break;
        case '?':
          err++;
        }
  }
if(optind >= argc)
  err++;
if(err)
  { fprintf(stderr,"Usage: %s {-s [scale-file]} {-d}
            [fname]\n",cmd);
    exit(err);
  }

strcpy(fname,argv[optind]);
strcat(fname,".r");
if((fd = fopen(fname,"r")) == NULL)
  { fprintf(stderr,"%s: Cannot find %s.\n",cmd,fname);
    exit(-1);
  }
Rpic = pm_read(fd,NULL);
```

```
fclose(fd);
strcpy(fname,argv[optind]);
strcat(fname,".g");
if((fd = fopen(fname,"r")) == NULL)
  { fprintf(stderr,"%s: Cannot find %s.\n",cmd,fname);
    exit(-1);
  }
Gpic = pm_read(fd,NULL);
fclose(fd);
strcpy(fname,argv[optind]);
strcat(fname,".b");
if((fd = fopen(fname,"r")) == NULL)
  { fprintf(stderr,"%s: Cannot find %s.\n",cmd,fname);
    exit(-1);
  }
Bpic = pm_read(fd,NULL);
fclose(fd);

result = pm_alloc();
result->pm_nband = 3;
result->pm_ncol = Rpic->pm_ncol;
result->pm_nrow = Rpic->pm_nrow;
result->pm_form = PM_F;
result = pm_prep(result,result);
res = (float *) result->pm_image;
if(do_norm)
  { norm_pic = pm_alloc();
    norm_pic->pm_ncol = Rpic->pm_ncol;
    norm_pic->pm_nrow = Rpic->pm_nrow;
```

```
    norm_pic->pm_form = PM_F;
    norm_pic = pm_prep(norm_pic,norm_pic);
    norm_img = (float *) norm_pic->pm_image;
  }

for(i=0;i<pm_nelm(Rpic);i++)
  { r =  (Rpic->pm_image[i]);
    g =  (Gpic->pm_image[i]);
    b =  (Bpic->pm_image[i]);
    if(do_norm)
      norm1 = sqrt(r * r + g * g + b * b);
    if(scale)
      { r *= r_scale;
        g *= g_scale;
        b *= b_scale;
      }
    norm = sqrt(r * r + g * g + b * b);
    r /= norm;
    g /= norm;
    b /= norm;
    res[3*i] = r;
    res[3*i+1] = g;
    res[3*i+2] = b;
    if(do_norm)
      norm_img[i] = norm1;
  }

strcpy(fname,argv[optind]);
strcat(fname,".rgb");
```

```
  fd = fopen(fname,"w");
  pm_write(fd,result);
  fclose(fd);

  if(do_norm)
    { strcpy(fname,argv[optind]);
      strcat(fname,".norm");
      fd = fopen(fname,"w");
      pm_write(fd,norm_pic);
      fclose(fd);
    }
}
```

B.2 Mean Program

This program finds the mean RGB vector with respect to the spherical coordinates. It finds the (θ, ϕ) values as well as the means of the red, green and blue components. This program also calculates the dot product with the mean. This program was originally written by Jakov Kucan and modified by myself. The usage is: **RGBmean [name]** where the name of the file is the output of the normalization program.

The program is used to create a file called all.means. This file contains the red, green and blue component averages and the (θ, ϕ) values. It also is used to create a file called means which contains the (θ, ϕ) values for every color chip. The means file is the base data which the classification program uses to compare to.

```
#include<stdio.h>
#include<math.h>
#include<local/pm.h>
#include<string.h>
```

```
#define TRUE 1
#define FALSE 0
#define RAD2DEG 57.29578

#define NARGS 1

#define max(a,b) (((a) > (b)) ? (a) : (b))
#define min(a,b) (((a) < (b)) ? (a) : (b))

main(argc,argv)
     int argc;
     char **argv;

{ int i;
  pmpic *pic, *dot_pic;
  float *pic_image, *dot_pic_img;
  double r, g, b, fi, theta;
  double mean_r, mean_g, mean_b, sum_fi, sum_theta;
  double dot, sum_dot, sum_2_dot, max_dot, mean_dot,
         var_dot;
  char fname[256];
  FILE *fd;
  char *cmd;

  cmd = argv[0];
  if(--argc < NARGS)
    { fprintf(stderr,"Usage: %s [fname]\n",cmd);
      exit(0);
```

```
    }

  strcpy(fname,argv[1]);
  strcat(fname,".rgb");
  if((fd = fopen(fname,"r")) == NULL)
    { fprintf(stderr,"%s: Cannot find %s.\n",cmd,
              fname);
      exit(-1);
    }
  pic = pm_read(fd,NULL);
  fclose(fd);


/* find the mean wrt spherical coordinates */

  pic_image = (float *) pic->pm_image;

  sum_fi = sum_theta = 0.0;
  for(i=0;i<pm_nelm(pic);i++)
    { r = pic_image[3*i];
      g = pic_image[3*i+1];
      b = pic_image[3*i+2];
      sum_fi += atan(g/r);
      sum_theta += acos(b);
    }
  fi = sum_fi / pm_nelm(pic);
  theta = sum_theta / pm_nelm(pic);
  mean_r = sin(theta) * cos(fi);
  mean_g = sin(theta) * sin(fi);
```

```
  mean_b = cos(theta);

  printf("%s:\n",argv[1]);
  printf("%lf %lf %lf\n",mean_r,mean_g,mean_b);
  printf("%s\t%lf\t%lf\n",argv[1],theta*RAD2DEG,
         fi*RAD2DEG);

/* do the dot product with mean (i.e. d(x,y) =
   arccos(<x,y>)) */

  dot_pic = pm_alloc();
  dot_pic->pm_nrow = pic->pm_nrow;
  dot_pic->pm_ncol = pic->pm_ncol;
  dot_pic->pm_form = PM_F;
  dot_pic = pm_prep(dot_pic,dot_pic);
  dot_pic_img = (float *) dot_pic->pm_image;

  sum_dot = sum_2_dot = 0.0;
  max_dot = 0.0;
  for(i=0;i<pm_nelm(pic);i++)
    { r = pic_image[3*i];
      g = pic_image[3*i+1];
      b = pic_image[3*i+2];
      dot = acos(mean_r * r + mean_g * g + mean_b * b);
      printf("%lf,",dot*RAD2DEG);
      if(dot>max_dot)
        max_dot = dot;
      sum_dot += dot;
      sum_2_dot += dot * dot;
```

```
      dot_pic_img[i] = dot;
    }
  mean_dot = sum_dot / pm_nelm(pic);
  var_dot = (sum_2_dot / pm_nelm(pic));
  printf("%lf\n\n",max_dot*RAD2DEG);

  strcpy(fname,argv[1]);
  strcat(fname,".dot");
  fd = fopen(fname,"w");
  pm_write(fd,dot_pic);
  fclose(fd);

}
```

B.3 Single Pixel Classification Program

This file reads in the (θ,φ) values from a file "means". It puts this information into an array of structures Theta_Fi. This array will be searched during the classification portion of the program.

Then the normalized vectors are read in. The (θ,φ) values are calculated from this image for a specific pixel. These are the values we will search for in the indata array. The binary search is used to find these values.

The usage of the program is **class [name] row column**, where name is the file created by the normalizing program and row column is the position of the pixel in the image we wish to classify. The normalizing program must be run before this classification can be completed.

```
#include<stdio.h>
#include<math.h>
```

```
#include<local/pm.h>
#include<string.h>

#define TRUE 1
#define FALSE 0
#define RAD2DEG 57.29578
#define MAX 237
#define NARGS 1

#define max(a,b) (((a) > (b)) ? (a) : (b))
#define min(a,b) (((a) < (b)) ? (a) : (b))

typedef struct t_f_struct
{ char  *name;
  double theta;
  double fi;
} theta_fi;


main(argc,argv)
     int argc;
     char **argv;

{


  theta_fi *indata[237];
  theta_fi *item;
  int i,pixel,high,low = 0,middle,row_pixel,col_pixel,
```

```
    x,j;
  float *pic_image;
  pmpic *pic;
  double fi, theta, r, g, b,buff2,buff3,sum_fi,fiaccum,
         thetaaccum;
  char fname[256];
  char *name;
  FILE *fd;
  char *cmd;
  char *buff = (char *) malloc(1024);

  cmd = argv[0];
  if(--argc < NARGS)
    { fprintf(stderr,"Usage: %s [fname]\n",cmd);
      exit(0);
    }
  row_pixel = atoi(argv[3]);
  col_pixel = atoi(argv[2]);

/* Read in the file containing the  Theta and Fi values
   to be used for classification. The array indata will be
   searched for the classification */

  fd = fopen("means","r");
  i = 0;
  while ((!feof(fd)) && (i < MAX))
    {
      fscanf(fd,"%s\t%lf\t%lf\n",buff,&buff2,&buff3);
      item =  (struct t_f_struct *)
```

```
        malloc(sizeof(struct t_f_struct));
      name = strdup(buff);
      item->name = name;
      item->theta = buff2;
      item->fi = buff3;
      indata[i++] = item;
    }

  fclose(fd);
  high = i;
/* Read in the normalized vectors */

  strcpy(fname,argv[1]);
  strcat(fname,".rgb");
  if((fd = fopen(fname,"r")) == NULL)
    { fprintf(stderr,"%s: Cannot find %s.\n",cmd,fname);
      exit(-1);
    }
  pic = pm_read(fd,NULL);
  fclose(fd);


/* Calculate theta and fi for a specific pixel in
   the image */

  pic_image = (float *) pic->pm_image;
  theta = fi = 0;

  i = ((row_pixel - 1)*512) + j;
```

```
  r = pic_image[3*i];
  g = pic_image[3*i+1];
  b = pic_image[3*i+2];

  fi= atan(g/r)*RAD2DEG;
  theta= acos(b)*RAD2DEG;

  row_pixel = i / 512;
  col_pixel = i - (row_pixel * 512);

  low = 0;

/* here we want the search stuff */

  printf("\n\nThe pixel is row %d and column %d of %s\n",
          row_pixel+1,
  col_pixel,fname);
  printf("Theta is %lf and Fi is %lf\n",theta,fi);
  printf("\nThe results of the search are: \n");

  while(low <= 236) {
        if (((theta + 2.0) >= indata[low]->theta) &&
                ((theta - 2.0) <= indata[low]->theta)) {
                if (((fi + 3.0) >= indata[low]->fi) &&
                    ((fi - 3.0) <= indata[low]->fi))
                        {
                          printf("%s\t%lf\t%lf\n",
                                indata[low]->name,
```

```
                                indata[low]->theta,
                                indata[low]->fi);
                        }
                }
      low = low + 1;
    }

}
```

B.4 Averaging Pixel Classification Program

This file is exactly like the single pixel classification program except that it takes an average of a range of pixels and uses this to find the classification results. This particular copy of the program is set to take the average over a neighborhood of 9 pixels.

```
#include<stdio.h>
#include<math.h>
#include<local/pm.h>
#include<string.h>

#define TRUE 1
#define FALSE 0
#define RAD2DEG 57.29578
#define MAX 237
#define NARGS 1

#define max(a,b) (((a) > (b)) ? (a) : (b))
#define min(a,b) (((a) < (b)) ? (a) : (b))
```

```
typedef struct t_f_struct
{ char  *name;
  double theta;
  double fi;
} theta_fi;


main(argc,argv)
     int argc;
     char **argv;

{


  theta_fi *indata[237];
  theta_fi *item;
  int i,pixel,high,low = 0,middle,row_pixel,col_pixel,
      x,j;
  float *pic_image;
  pmpic *pic;
  double fi, theta, r, g, b,buff2,buff3,sum_fi,fiaccum,
         thetaaccum;
  char fname[256];
  char *name;
  FILE *fd;
  char *cmd;
  char *buff = (char *) malloc(1024);

  cmd = argv[0];
```

```
  if(--argc < NARGS)
    { fprintf(stderr,"Usage: %s [fname]\n",cmd);
      exit(0);
    }
  row_pixel = atoi(argv[3]);
  col_pixel = atoi(argv[2]);

/* Read in the file containing the  Theta and Fi values
   to be used for classification. The array indata will
   be searched for the classification */

  fd = fopen("means","r");
  i = 0;
  while ((!feof(fd)) && (i < MAX))
    {
      fscanf(fd,"%s\t%lf\t%lf\n",buff,&buff2,&buff3);
      item =  (struct t_f_struct *)
        malloc(sizeof(struct t_f_struct));
      name = strdup(buff);
      item->name = name;
      item->theta = buff2;
      item->fi = buff3;
      indata[i++] = item;
    }

  fclose(fd);
  high = i;

/* Read in the normalized vectors */
```

```
  strcpy(fname,argv[1]);
  strcat(fname,".rgb");
  if((fd = fopen(fname,"r")) == NULL)
    { fprintf(stderr,"%s: Cannot find %s.\n",cmd,fname);
      exit(-1);
    }
  pic = pm_read(fd,NULL);
  fclose(fd);


/* Calculate theta and fi for a set of pixels in
   the image */

  sum_fi = 0;
  pic_image = (float *) pic->pm_image;
  for(x = (row_pixel - 1);x < (row_pixel + 2);x++)
        for(j = (col_pixel - 1);j < (col_pixel + 2);j++)
        {
        theta = fi = 0;

        i = ((row_pixel - 1)*512) + j;

        r = pic_image[3*i];
        g = pic_image[3*i+1];
        b = pic_image[3*i+2];

        fiaccum += atan(g/r)*RAD2DEG;
        thetaaccum += acos(b)*RAD2DEG;
```

```
        }
    fi = fiaccum/9;
    theta = thetaaccum/9;


/* here we want the search stuff */
    printf("\n\nThe pixel is row %d and column %d of %s\n",
           row_pixel+1,col_pixel,fname);
    printf("Theta is %lf and Fi is %lf\n",theta,fi);
    printf("\nThe results of the search are: \n");

    while(low <= 236) {
       if (((theta + 2.0) >= indata[low]->theta) &&
           ((theta - 2.0) <= indata[low]->theta)){
         if (((fi + 3.0) >= indata[low]->fi) &&
             ((fi - 3.0) <= indata[low]->fi))
           {
             printf("%s\t%lf\t%lf\n",indata[low]->name,
                    indata[low]->theta,
                    indata[low]->fi);
           }
       }
       low = low + 1;
     }
}
```

Bibliography

[1] Berger, J. (1985) *Statistical Decision Theory and Bayesian Analysis*, Second Edition, Springer-Verlag, New York.

[2] Casella, G. and Berger, R. L. (1990) *Statistical Inference*, Wadsworth and Brooks, Pacific Grove, Cal.

[3] Dowdy, S. and Wearden, S. (1983) *Statistics for Research*, Wiley, New York.

[4] Ferguson, T. (1967) *Mathematical Statistics A Decision Theoretic Approach*, Academic Press, New York.

[5] Funt, B. and Finalyson, G. (1991) *Color Constant Color Indexing*, Technical Report, Simon Fraser University, CSS/LCCR TR91-09.

[6] Hall, Roy (1989) *Illumination and Color in Computer Generated Imagery*, Springer-Verlag, New York.

[7] Kamberova, G. and Mintz, M. (Aug. 1990) *Robust Multi-Sensor Fusion: A Decision-Theoretic Approach*, Technical Report, University of Pennsylvania, MS-CIS-90-57, (229).

[8] Lehmann, E L. (1986) *Testing Statistical Hypothesis*, John Wiley & Sons, New York.

[9] McKendall, R. (1990) *Minimax Estimation of a Discrete Location Parameter for a Continuous Distribution*, PhD Thesis, University of Pennsylvania, MS-CIS-90-28, (214).

[10] McKendall, R. and Mintz, M (Sept. 1990) *Sensor-Fusion with Statistical Decision Theory: A Prospectus of Research in the GRASP Lab*, Technical Report, University of Pennsylvania, MS-CIS-90-68, (234).

[11] Munsell, A. H. (1946) *A Color Notation*, Munsell Color Company, Inc., Baltimore, Maryland.

[12] General Robotics and Active Sensory Perception Laboratory (1990) *PM Manual*, University of Pennsylvania.

[13] Swain, M. (1990) *Color Indexing*, PhD Thesis, University of Rochester, TR-360.

[14] Walpole, R. E. and Myers, R. H. (1985) *Probability and Statistics for Engineers and Scientists*, MacMillan Publishing Co., New York, Pages 259 - 315.