Spectral Image Analysis and Segmentation Techniques

Sebastián Bravo López

March 26, 2007

University of Joensuu
Department of Computer Science
Master's Thesis

Abstract

In this thesis we analyze spectral images from several points of view. We develop a software capable of making some basic editing, croping spectral images, visualizing a specific range of wavelengths and applying segmentation techniques to the spectral images. In addition we develop a software capable of drawing figures to spectral images. This software will be use to create artificial spectral images in order to test the different segmentation techniques. Finally we develop a software capable of reading the spectral data of a defined area giving the mean values of the selected area. The output data are the spectral data, the L*a*b* coordinates and the sRGB values.
We make some testing and analysis of spectral images from paper samples. The paper samples are printed with three different printing densities lower, target and higher. We will obtain data from the spectral images and make a comparison of lower and higher densities respect the target density.
The segmentation results depends on the algorithm used. Our software uses three different algorithms for segmentation. The clustering algorithms work with features obtained from spectral and La data, the results are greyscale images with areas that represent the clusters. With clustering we can easily distingue parts of the original image with similar color properties, like the skin, clothes and background. Lab segmentation gives us a greyscale image with the pixels that are within a defined by the user distance from the Lab coordinates. It is useful to highlight pixels with similar Lab coordinates. The last approach is based on distance from spectral that belongs to different pixels. We can find areas with similar spectra, it does not provide necessarily information about color differences because distance is not based on the shape of the spectra. It can be used to find which areas of the image have similar spectra respect the selected area.

Contents

List of Figures

Chapter 1

Introduction

Light is all around us, it floods the world from the powerful sunlight to the small candle. It is the responsible of the particular view of the nature that we the humans have. But there are other points of view, the human eye only perceive a limited fraction of the light and only in one specific and simplified way. With the use of special devices it is possible to overcome the human limits and explore the different shades the light try to show us.

With the use of spectral images it is possible to study in a more detailed way light properties. Spectral images have more information about the scene than standard RGB images. It is possible to study spectral images from different points of view [Pee93] [WEV02].

In the software presented in this thesis we work with spectral images. Spectral images are a set of pixels with information of different wavelengths for each one. Each pixel contains a vector of length K, the dimension of the image is NxMxK.
Instead of using a value to determine the luminance of the pixels like in the greyscale images, or three component representing the three basic colors red, green, blue (RGB) like in color images, we have information about many spectral wavelengths for each pixel of the image [WEV02]. Spectral images can be considered as a collection of several monochrome images, each of them referring to different wavelength.

The amount of data of spectral images is bigger than the amount of data used to represent a RGB image. For a RGB image we simply have three values representing red, green and blue color. The spectral images use more information, one single value for each wavelength.

The aim of having such amount of data is that we can use it to study certain properties that can give us information to make segmentation or compare different spectral images in order to find similarities and differences. In spectral images it is possible to visualize only certain part of the wavelengths showing certain properties that could not be viewed in RGB images.

Segmentation of images can be done using several criterions. In this thesis we use segmentation by color differences, with the use of L*a*b* space and spectral data [Pee93]. In order to study the different algorithms we use artificial spectral images created by our own software.

Spectral images can be studied also comparing them with other spectral images. This can be useful to determine differences in spectral images studying spectra or color spaces. In this thesis we study differences between paper samples printed in with three ink densities and using different procedures. With this study we can assess the quality of the resulting printed samples.

The structure of this thesis is arranged as follows: In Chapter 2 we will present some of the hardware involved in taking the spectral images. In Chapter 3 we will study the different ways to represent spectral images in the computer, the standard observers 1931 and 1964, the effect of the selected illuminant and the L*a*b* coordinates. In Chapter 4 we will present the different clustering methods that our software make use of. Finally in Chapter 5 is the software user manual with all the functionalities available and some results of the segmentation algorithms. Finally in Chapter 6 we show the resulting images and graphs comparing the data.

Chapter 2

Spectral Image Acquisition

In greyscale images we work with sets of points with certain value for each one. This value represents how dark or light the point is. In color images the principle is the same but we work with three different values for each point.

In spectral images we have pixels with information from more channels each one related to different wavelength. We can define spectral images as groups of greyscale images each of them referring to a single wavelength. Spectral images are taken with spectral cameras. These devices can obtain information about the light in different wavelengths [TT99].

A spectrometer is an optical device capable of measure a specific portion of the light. In comparison with standard RGB cameras, a spectral camera provides much more color information per pixel. Color resolution is much better and we can easily obtain and modify information like the illuminant applied.

A spectral camera has a set of sensors sensitive to certain light wavelengths or a single sensor capable of dividing the light into wavelengths. The spectral camera has a prism that separates the light into different wavelengths. The light is spread along the sensor. The sensor is a square with several rows, each row is sensible to a defined wavelength.
It is possible to obtain spectral data with only few sensors, this approach is often called multichannel camera. In fact a spectral image can be obtained from sensors sensible to two wavelengths or more. The whole spectral image can be obtained then with the use of interpolation technique [TT99].

For more detailed analysis it is better to have a spectral camera sensible to as much wavelengths as possible.

The spectral camera used to take the spectral images for this thesis have sensors capable of detect a wide amount of wavelengths. This kind of spectral cameras can measure one line of an area at the same time and record spectral information about that line.

To take the spectral image some previous steps are needed. The sensors should be calibrated establishing a white reference and black reference. Sensors do not have exactly the same properties, the materials and the circuits inside the camera can give different values depending on the ambient heat and other factors. The calibration of the camera white

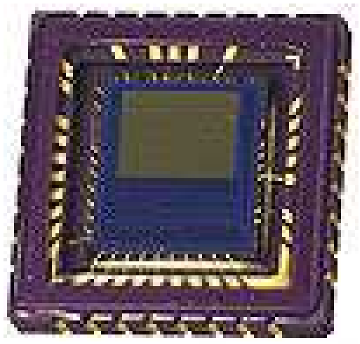


Figure 2.1: CMOS Sensor, courtesy of AXIS Comunications

and black values is useful to correct the obtained data to more accurate values [TT99].

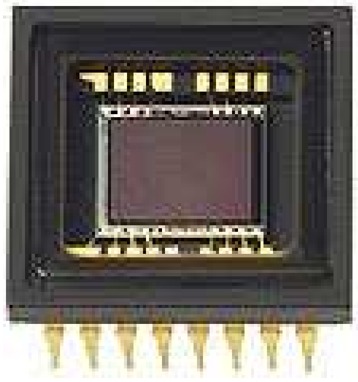


Figure 2.2: CCD Sensor, Courtesy of AXIS Comunications

There are two different types of sensors depending on the technology used. CCD (Charge Coupled Device) sensors (Fig. 2.2) and CMOS (Complementary Metal Oxide Semiconductor) (Fig. 2.1) [GCKP01]are two different approaches. Both sensors convert light into electric signals.

The working operation of CCD sensors imply that one sensor or pixel transfer the information to a limited number of output nodes and after that the electric signal is converted to digital with the use of a AD (Analog to Digital) converter [Reg76].

Because the information is passed through the nodes, the final information of a column is affected by all the nodes in the column. This have the not desired effect that if it exist a very bright source light in some area or there exist a scene with lot of contrast, it will affect the whole column and vertical stripes can appear. This effect is called smear.

In a CMOS sensor each pixel has his own AD converter and we can read the information of one pixel directly from the sensor without taking into account the information of other nodes. CMOS sensor also has some nice features built in like noise reduction and correction.

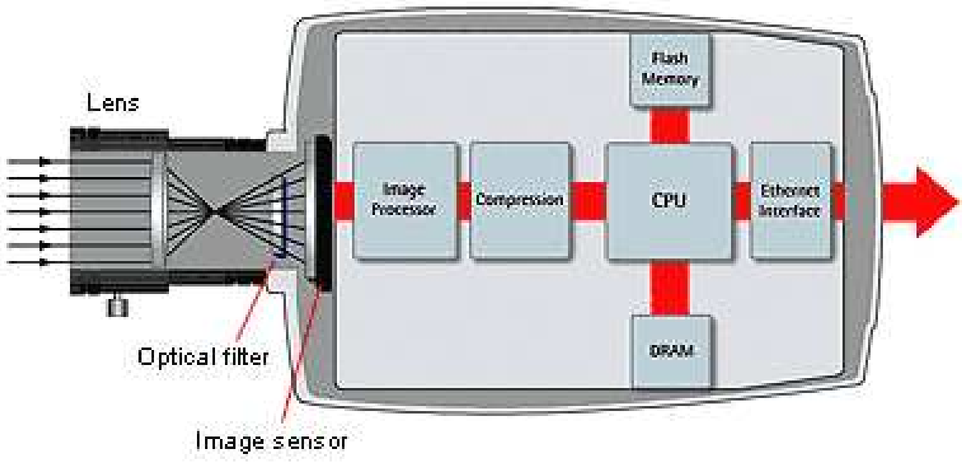


Figure 2.3: Inside Camera Operation, courtesy of AXIS Comunications

CMOS sensors needs more physical space than the CCD ones due to the need to implement more elements like the noise reduction and AD converters into the sensor itself.

CCD sensors where invented in the 1960s (Fig. 2.4) and CMOS in the 1970s. The first ones had better quality because they produce less noise and they produced at the beginning images with better resolution than CMOS technology. But CCDs are very expensive to produce due to the materials used while CMOS technology use cheap materials and the production techniques are exactly the same used for any chip.

Right now the need to have very high quality images with high resolution and very low noise makes the companies to produce spectral cameras with CCD sensors. For the RGB cameras the CCD surface is made of sensors with certain filters for the red, green and blue, creating a pattern. Two common patters are the Bayer arrangement and the RGBE arrangement that includes also filters for cyan color.

In the spectral camera (Fig. 2.3 and Fig. 2.5) the principle is the same, we have a CCD or CMOS sensor and each line is sensible to a certain wavelength. The method is even simpler than the used to take RGB images because we do not need to create a specific complex pattern with red, green and blue filters, just put a filter for certain wavelength to every row.

After we obtain the line information (Fig. 2.6) we move either the camera or the object to measure to the next line. The information is recorded to create the whole spectral image as a set of lines with spectral information.

Figure 2.4: Willard Boyle (left) and George Smith (right). Courtesy of Lucent Technologies. 1969

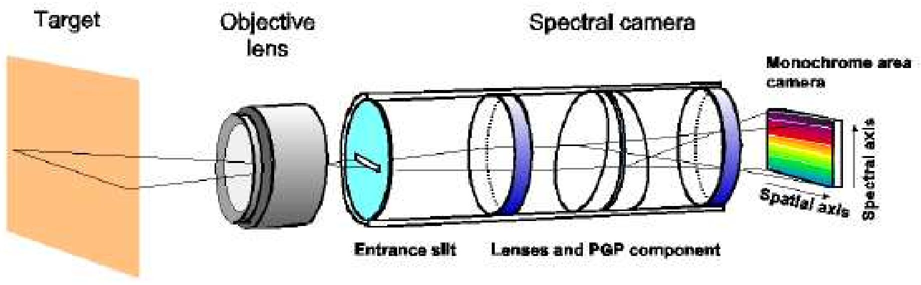


Figure 2.5: Spectral Camera Operation, courtesy of Spectral Imaging Ltd. ImSpector

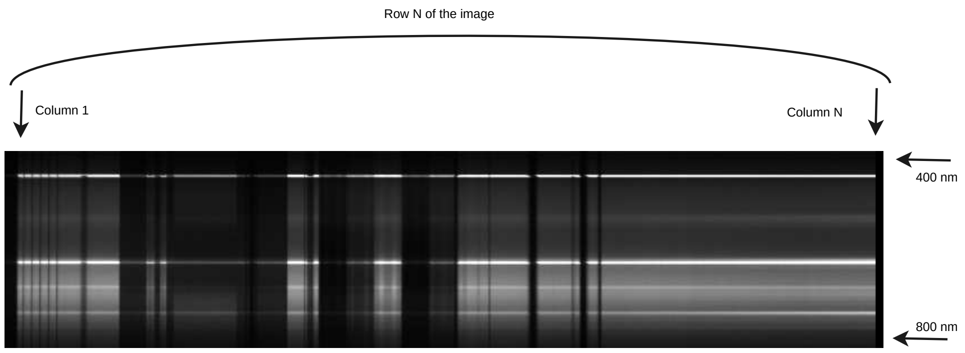


Figure 2.6: Line Scanned by the spectral camera

Chapter 3

From Spectral Image To Three Component Image

In this chapter we will study the methods used to transform a spectral image to RGB image format in order to represent it as a color image in the computer monitor or printed paper. We will see the properties of the human vision and some approaches to represent RGB from spectral data. From spectral data we can also obtain L*a*b* coordinates that is one of the steps between spectral image and RGB image. L*a*b* coordinates are closer to the human vision than RGB values.

3.1 Eye Properties

The visible range of wavelengths that the human eye can detect goes from 380 nanometers to about 780 nanometers. This range gives the human eye a lot of information to be processed [WEV02].

The acquisition of the light information by the eye makes use of certain light sensitive cells, these elements are known as cones and rods (Fig. 3.1). The cones can be separated into three classes, each class being sensitive to a different spectral distribution of radiation (Fig. 3.2).

This trichromacy of colour sensation means that many different spectral distributions can produce the same perceived colour. Such equivalent stimuli, even though they have physically different spectral distributions, are called metamers and the phenomena metamerism [Pee93] [WEV02] [WS82] [otCIdL86]. Metamerism is important to understand how the colors are perceived by humans and therefore it allow the creation of models for representing the different wavelengths and colors.

When we try to study how the colors are perceived by the human eye we need to define which features of the light the eye detects and how they are detected.

Because the light is not composed of a single wavelength, the perceived color is a function distribution of the whole spectra. The different shapes and curves generated by the different wavelengths are perceived by the human eye as colors or chromacity. The intensity is given by the height of the distribution of the spectra [Pee93] [WEV02].

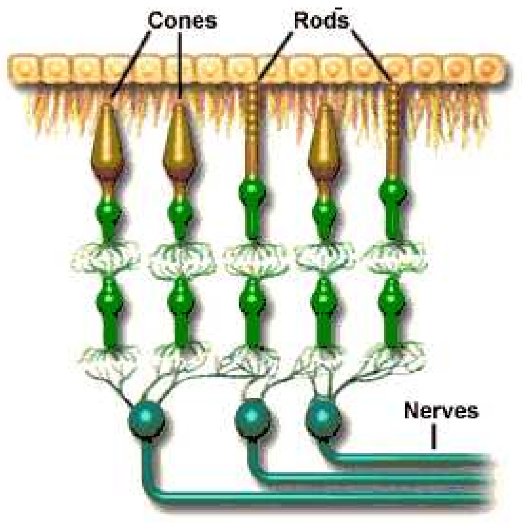


Figure 3.1: Eye Cones And Rods

The different cones in the eye can respond to colors and intensity, different wavelengths are weighted by rods and cones to produce the effect of color and intensity. All of them provide information to the brain to reconstruct the image with all his properties.

Different set of wavelengths can produce the perception of the same color. The characteristics of the human eye with the use of the cones makes possible to represent different wavelengths as the same color and vice versa [WEV02] [WS82] [otCIdL86]. For the particular case of the human eye, there are about 6 million of receptors (cones and rods).

Rods are responsible of the intensity of the image more sensitive in the peak 498nm. They are not sensitive to color differences but to different intensities of the light. Most part of the receptors are rods. The cones are the responsible of the sensation of color or hue and the rods take care about the luminance or light intensity.

The cones respond to the different colors in the following way:

- yellow-green wavelengths (peak in 564nm), very often called long or L
- blue-green (peak in 534nm) often called medium or M
- blue-violet (peak in 420nm) often called short or S

The visible range of wavelengths can be represented as three component with given intensity [Pee93] [WEV02].

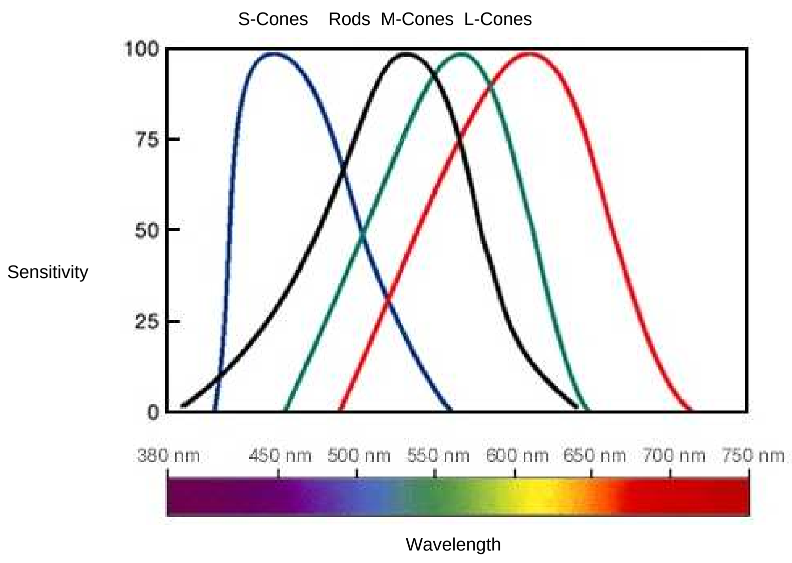


Figure 3.2: Perceived color by cone type. From Foundations of Vision by B. Wandell

3.2 Standard Observer 1931 & 1964

The International Commission on Illumination (CIE) set the 1931 and 1964 color matching functions for colorimetric observer viewer for imaging systems [otCIdL86].

The primary colors are extracted from the spectra with the use of color matching functions. Because the final results obtained depends on the matching functions selected, there exist several standards that define the way we obtain color information to the tristimulus values.

In this thesis we use two different approaches called CIE Standard Observers. Both of them are defined by the "International Commission of Illumination" to obtain the three stimulus values by the use of a given color matching function [otCIdL86].

Experiments done in the late 1920s by W. David Wright (Wright 1928) and John Guild (Guild 1931) derived in a color reconstruction function (Fig. 3.3). Those experiments were made to create certain matching functions in order to obtain the three components to represent a image from the spectra.

The experiments were made in a 2 degree visual angle and with the use of three source lights representing the colors red, blue, and green. In addition Wright and Guild adapted the matching functions to ensure a good luminosity representation.

$$R = \int_0^\infty \bar{r}(\lambda)\, d\lambda; G = \int_0^\infty \bar{g}(\lambda)\, \bar{g}\lambda; B = \int_0^\infty \bar{b}(\lambda)\, d\lambda$$

This system in often called 2 degree observer or 1931 standard observer.

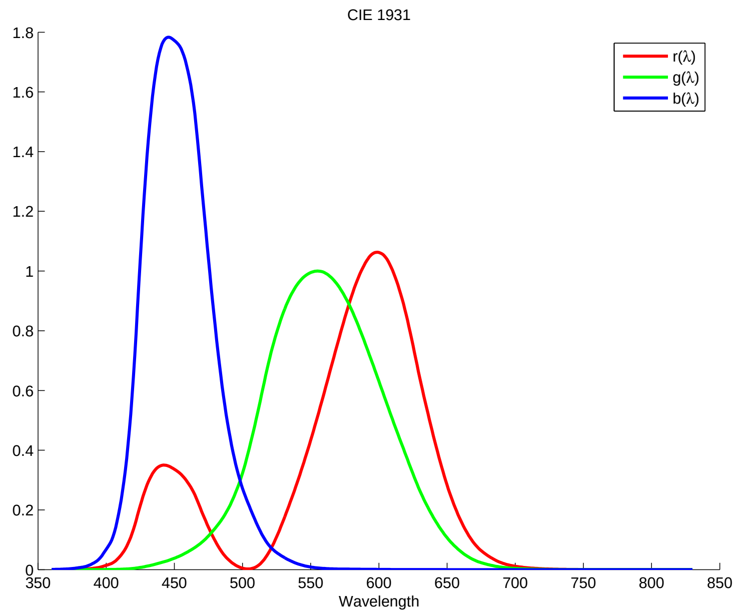


Figure 3.3: The CIE 1931 RGB Color matching functions. $\bar{r}(\lambda)$, $\bar{g}(\lambda)$, $\bar{b}(\lambda)$

There are two problems derived from the CIE 1931 standard observer. One is that the color matching functions have both positive and negative values and they have an angle of field view of only 2 degrees.

The standard observer 1964 (Fig. 3.4) solves these problems, adding a wider angle of view in the field of 10 degree. Also it was increased the sensitivity of wavelengths below 460nm, which was underestimated in CIE 1931 standard observer.

We will not discuss in deep the changes and properties of the new model because is not the aim of this thesis. The $x(\lambda), y(\lambda), z(\lambda)$ can be used with both observer to negative values. The software developed here makes use of both approaches and there were needed only minor changes between the CIE 1931 model and the CIE 1964, in fact it is only needed to change the distribution functions for the three primary colors [otCIdL86] [WS82].

3.3 Illuminants

The perception of colors relies on the source of the light [Pee93] [WHD03]. In fact the light is reflected by objects and they add their color properties to the reflected light. So the color depends partly on the source light that illuminates the object. Human vision is affected by cones and rods response to illumination, human eye can cancel partly illumination effects and detects spectra of surface.

The spectral radiant power of the light source S_λ is multiplied by the spectral reflectance characteristic of the object surface R_λ [WHD03]. The result is the object color received by the observer P_λ :

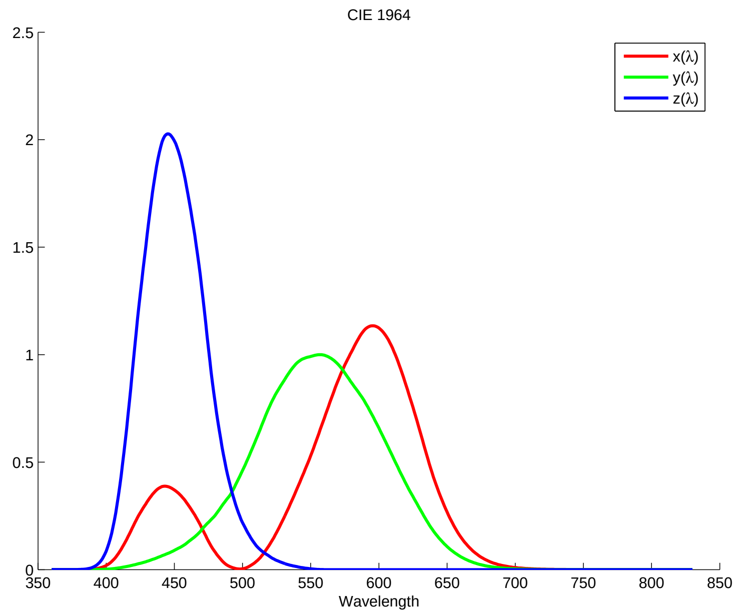


Figure 3.4: The CIE 1964 Color matching functions. $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, $\bar{z}(\lambda)$

$$P_\lambda d\lambda = R_\lambda S_\lambda d\lambda$$

In the CIE 1931 and CIE 1964 standard, the Y parameter represents the brightness or luminance of a color. The chromaticity of a color is obtained from the spectral data distribution, the light source and the reflectance of the object, being $x_\lambda, y_\lambda, z_\lambda$ spectral functions [WS82].

$$X = k\int_\lambda R_\lambda S_\lambda \bar{x}_\lambda d\lambda$$

$$Y = k\int_\lambda R_\lambda S_\lambda \bar{y}_\lambda d\lambda$$

$$Z = k\int_\lambda R_\lambda S_\lambda \bar{z}_\lambda d\lambda$$

The CIE commission recommends to restrict the colorimetric measurements to a set of predefined distributions of radiant power called CIE standard illuminants. The distribution on CIE illuminants A, C, D50 and D65 are shown in figure (Fig. 3.5).
However in the software developed for this thesis we only use illuminants A, C, D50 and D65 with both CIE 1931 and CIE 1964 standard observers.

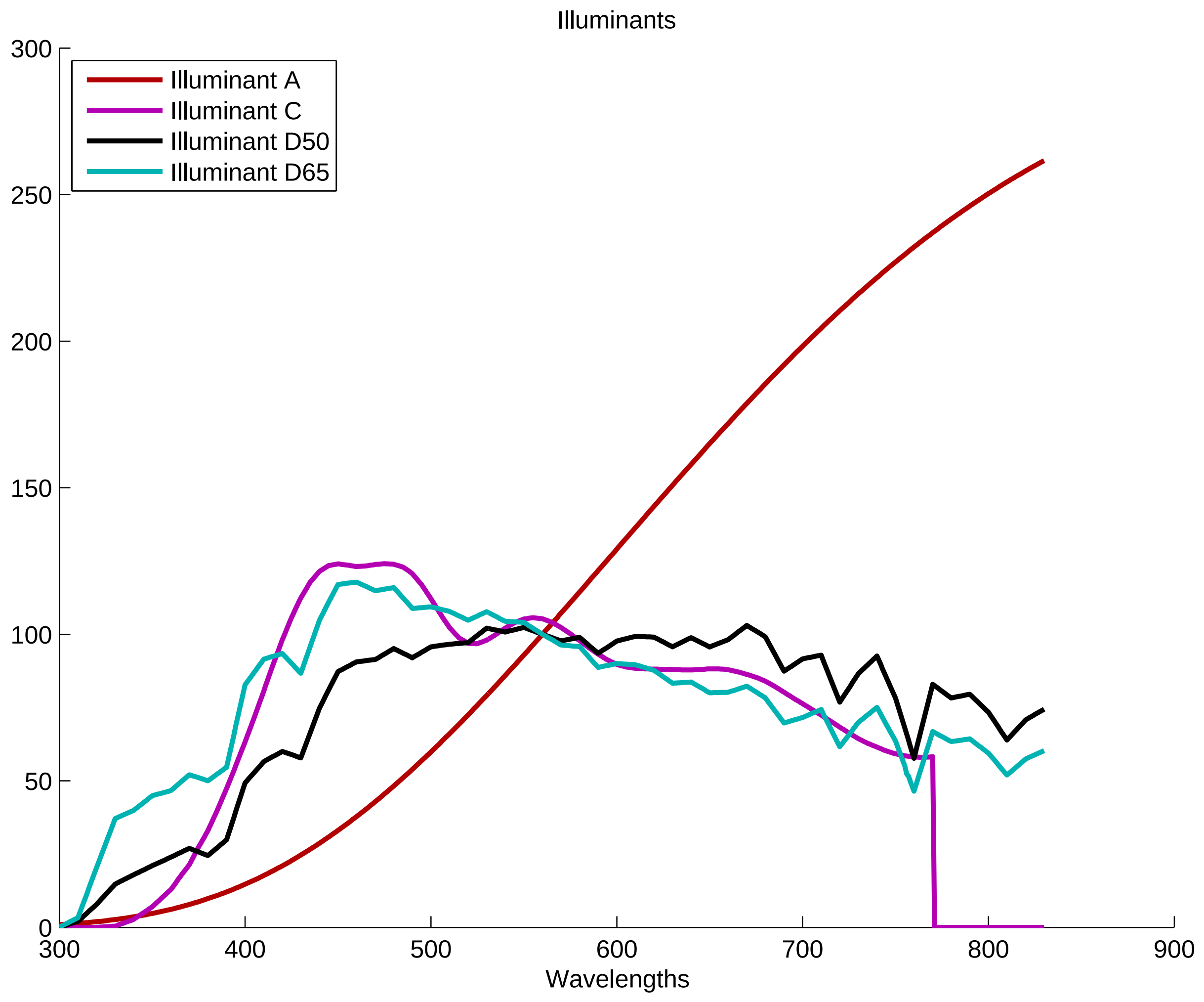


Figure 3.5: The CIE Illuminants A, B, D50 and D65

3.4 CIE Color Space

With the use of the color matching functions and taking into account the light source S_λ and reflectance R_λ, it is possible to obtain three values that represents the the perceived color from a certain spectra [Pee93] [WHD03].

The applied formulas to obtain the tristimulus values are:

$$X = k \sum_{\lambda=380}^{780} R_\lambda S_\lambda \overline{x}_\lambda$$

$$Y = k \sum_{\lambda=380}^{780} R_\lambda S_\lambda \overline{y}_\lambda$$

$$Z = k \sum_{\lambda=380}^{780} R_\lambda S_\lambda \overline{z}_\lambda$$

$\overline{x}$, $\overline{y}$, $\overline{z}$, are the matching functions. k is a normalizing factor set by taking the white reference.

$$k = \frac{Y}{S(\lambda)R(\lambda)\overline{y}(\lambda)}$$

And setting $Y = 100$ if the object is ideal white.

Chromaticity diagrams are taken with the use of the following parameters:

$$x = \frac{X}{X+Y+Z}$$

$$y = \frac{Y}{X+Y+Z}$$

$$z = \frac{Z}{X+Y+Z}$$

The difference in the CIE 1931 and CIE 1964 diagrams is small, but the wider field of view, from 2 degree of the CIE 1931 to 10 degree of CIE 1964.

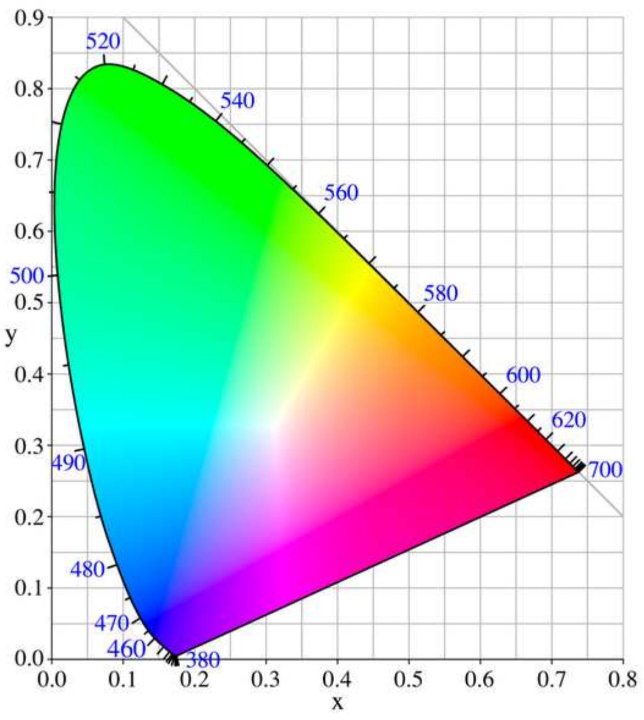


Figure 3.6: CIE Color Diagram

RGB values are the common way to represent the images on the computer, they are obtained by linear transformation from the X, Y, Z values.

The transformation from XYZ space to RGB space is achieved by:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 2.3647 & -0.8965 & -0.4681 \\ -0.5151 & 1.4264 & -0.0887 \\ -0.0052 & -0.0144 & 1.009 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

The opposite transformation from RGB to XYZ space is not used in this thesis either the software, it is made by the use of the following transformation:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.4899 & 0.31 & 0.2 \\ 0.1769 & 0.8124 & 0.01 \\ 0 & 0.01 & 0.99 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

3.5 CIE L a* b*

Lab is the abbreviation of CIELAB (CIE 1976 L*a*b*) (or Hunter L, a, b).

Both spaces are based on the CIE 1931 and CIE 1964 XYZ space. The main aim of both color spaces is to provide a correspondence between the changes in the color and the perceived color. This relay in the property of perceptual linearity, it means that if we make changes in the properties of a color, the perceived color should represent the same amount of changes, it should have the same visual impact or importance. This property can improve the reproduction of tones [Pee93] [WEV02] [WHD03] [HRV97].

Both spaces are relative to the white-point of the XYZ data they were obtained from. The LAB values do not represent absolute colors unless we provide the white-point reference. It is very common that the white-point is assumed to be a standard, often it is relative to CIE standard illuminant D50 [HRV97] [MG80].

The LAB color model is obtained from the XYZ space with the use of the following transformations:

$$\begin{aligned} L* &= 116 f\left(\frac{Y}{Y_0}\right) - 16 \\ a* &= 500\left[f\left(\frac{X}{X_0}\right) - f\left(\frac{Y}{Y_0}\right)\right] \\ b* &= 200\left[f\left(\frac{Y}{Y_0}\right) - f\left(\frac{Z}{Z_0}\right)\right] \end{aligned}$$

$$f(x) \ = x^{1/3} \qquad\qquad \text{if} x \geq 0.008856$$

Where X_0, Y_0, Z_0 are the tristimulus values of the white reference.

It is easier to work with CIELAB space than with RGB or CMYK, because the chromacity and luminance are separated [JG78] . With CIELAB it is possible to represent colors that are not possible in other spaces, which is useful for color manipulation of the pictures.

CIELAB is more similar to real human vision than RGB representation. One of the analysis that can be made to the spectral images is the segmentation based on L*a*b* coordinates, it gives fast results and solves the problem of working with the big amount of data the spectral images.

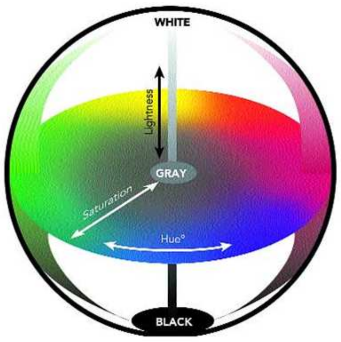


Figure 3.7: CIE Color Space Sphere, courtesy of [Wik06]

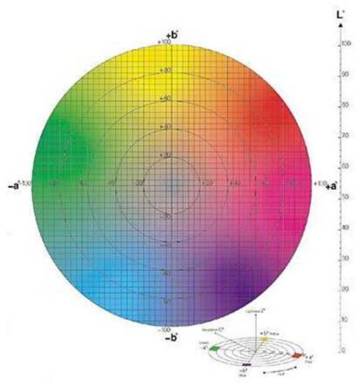


Figure 3.8: CIE Color Space Circle, courtesy of [Wik06]

Chapter 4

Segmentation

We call segmentation in the context of image analysis to the partitioning of the image into several regions following certain criterions. The pixels that belong to any area need to have some properties. The properties are defined by a set of features. The aim of the segmentation is to distinct between parts of the images. Clustering deals with finding a structure in a set of data with certain properties [MLAB06].

When we try to make a segmentation of a spectral image we need to consider which features we want to highlight before our first attempt to solve the problem [AS07].

Clustering is defined as the partitioning of objects into groups based on a given base of features. The use of those features give us the ability of decide when an object belongs to a group or not. In our case we have a wide variety of situations to use our software. Because of that we implemented three different algorithms to make the segmentation [MLAB06] [OO06].

4.1 Spectral Image Segmentation

Previous works in this area tried to apply different algorithms to the segmentation of color images or greyscale images. Since this is done in 2D space with a matrix that represents every point, it is relatively easy to attach the problem.

The segmentation algorithms rely on the detection of features that can make possible to say that one pixel belong or not belong to a certain group [KVV04] [KM06].

The selection of the features is the most important aspect of the segmentation. The results depend on the selection of the properties that the pixels should have and therefore the results can be slightly different depending on the set of features [OO06].

To make a good feature selection we should think about what we want to see in the segmentation. In our case we use three different algorithms that use different features. We will go more in deep in the next sections, but previously we will present some applications of the segmentation, particularly with spectral images [KVV04] [KM06] [Cha].

Segmentation usually is used in RGB images or black/white images. This makes everything more easily since we need to work with a two dimensional matrix and we only need to add some amount of features to distinct between the different groups.

The segmentation is useful to highlight regions on the image with some properties based on features. The features to be used with the algorithms are based on the image information, but sometimes that information can be pre-processed to get some specific set of features.

In greyscale images it is very common to use algorithms [KVV04] to highlight the shape of objects and the features to be the pixel position and pixel luminance. It is also possible to discover regions with some characteristic pattern [ZCH$^+$04] [Cha]. In our case we will have very powerful information, that is the spectral information of the pixel.

Unlike the greyscale images, with spectral images we have information about different light wavelengths for each pixel [KVV04], this provides us with more information than standard RGB color images. We can search in the image for pixels with certain properties.

In the spectral image we have several approaches. Because we have a multi dimensional matrix with the pixel position and an array of spectral values for different wavelengths, it looks that it will be much more difficult to apply segmentation algorithms [KM06] [ZCH$^+$04].

But again everything depends on the approach taken to make the algorithms. The features that represent a spectral pixel are based on the different wavelengths that conform the pixel. We can separate the spectra in different ranges of wavelengths each of them represented by the mean value of the spectra in the selected wavelength range, this mean a preprocessing of the spectral data before aplying an algorithm.
In this thesis we will use a different preprocessing of the spectral data. If we use the wavelengths that conform the spectra as features that represent the pixels, we have a very high amount of features.

If we have spectral data coming from 400 nanometers to 700 nanometers in 5 nanometer steps, we have a total of 61 wavelength values. Indeed this wavelength values represent very well the pixel information and characteristics, but to work with such amount of features have drawbacks.

The first drawback is the computational time to apply certain algorithms, for example the sequential clustering algorithm that we will use later, it is high [KVV04] it can take from couple of hours to several days depending on the spectral image size.
Second the features do not provide a good way to classify the pixel in all cases. If we try to determine similarities between pixels according to the color information from the spectral image we should be careful with the clustering algorithms because they may not provide good classification [KVV04] [ZCH$^+$04].

We will see in the following example that the classifiers can provide us different results than we expect.

Lets have a set of pixels with three features, they represent three different wavelengths of a specific spectral image. We select three wavelengths because it is easy to represent graphically in three dimensional space.

Let S to be a set of pixels with three features:

$$S = [0,0,0], [1,0,0], [0,1,0], [0,0,1], [1,0,1], [0.2,0,0.2], [0,0.2,0], [0.2,0.2,0.2]$$

Lets represent the features as spectral data, we obtain the figure (Fig. 4.1).

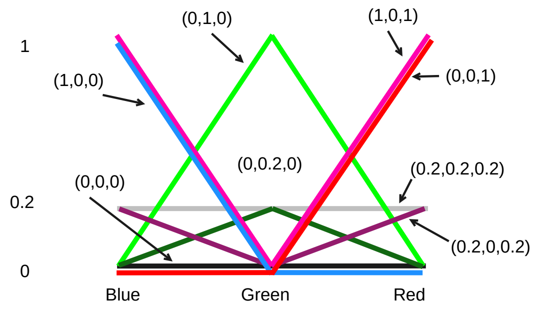


Figure 4.1: Overlap of Spectral Features

And the visual representation in three dimensional cardinal axes is shown in figure (Fig. 4.2), most of the clustering algorithms use Euclidean distance to make the clusters, in cardinal axes we can see the distance of the pixel features.

Depending on the clustering algorithm we have several groups as result, in the case of a clustering algorithm based on Euclidean distance the natural results are in general quite disappointing.

Basically the clustering algorithms try to generate groups based on distance between the central weight of the groups. We will see more in deep the clustering algorithms in the next sections. If we apply a clustering algorithm such as IsoData, the algorithm will find clusters with features that are close to each other. This kind of algorithms are highly affected by the Euclidean distance from the groups.

The results using clustering algorithm like IsoData are the following groups:

- $A = [1,0,0]$
- $B = [0,1,0]$
- $C = [0,0,1]$
- $D = [1,0,1]$
- $E = [0,0,0], [0.2,0,0.2], [0,0.2,0], [0.2,0.2,0.2]$

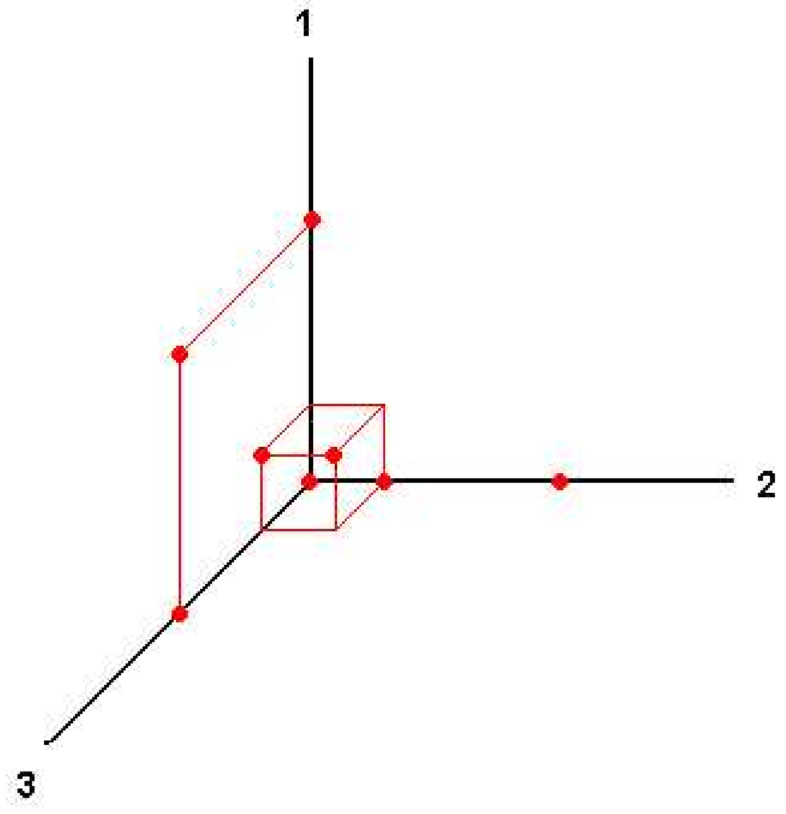


Figure 4.2: Cardinal Axes With Data Points

Isolated features will belong to their own cluster, features that are near other features will belong to another cluster. Because we are studying color differences and similarities to make the groups we should obtain the following results (Fig. 4.3):

- $A = [1, 0, 0]$
- $B = [0, 1, 0], [0, 0.2, 0]$
- $C = [0, 0, 1]$
- $D = [1, 0, 1], [0.2, 0, 0.2]$
- $E = [0, 0, 0], [0.2, 0.2, 0.2]$

Each group represents certain color. Euclidean distance between pixels is not the main similarity property we look for, Euclidean distance provide good results to find luminance differences but for color or spectral shape it is not useful.

This is an example about how the clustering algorithms can provide solutions that may not be good for our task. The features may need some pre-processing before use them with the algorithms. In spectral images we have a large amount of wavelengths, not all of them are good to know which cluster the pixels belong to.

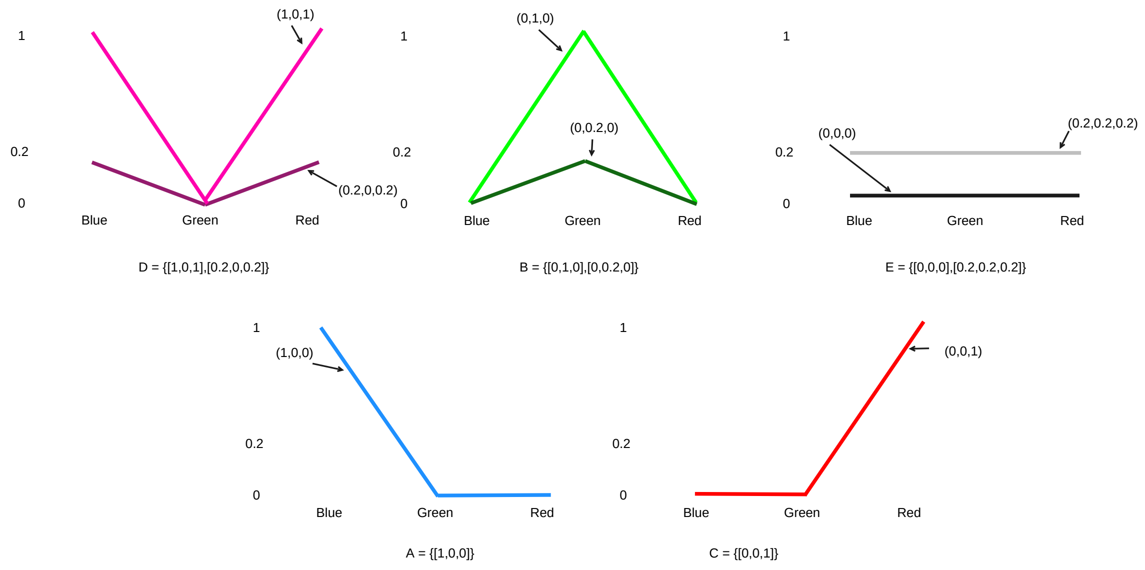


Figure 4.3: Correct Grouping With Color Differences

4.2 Clustering

The first task to be made before the designing of the clustering algorithm is to make a good selection of features that fully represent the proper characteristics we want to highlight. The number of features or the so called curse of dimensionality can be very large, in the order of hundreds.

Because most of the features do not provide a good information for classification because of the high mutual correlation existing between them. We need to create a feature vector good enough to represent clearly the different groups we want to obtain.
In our case we not only need to have a good classification scheme but also good performance in the algorithm. Our time is limited and the results should be shown to the user in few seconds. We need to retain the features that provide more discriminatory information [ST03].

At this moment we have a spectral image where each point is formed by a large number of wavelengths. If we use those wavelengths as individual features we will have a total number of dimensions that is quite large and the computational time for the algorithms to complete the calculations will take several hours. So it is mandatory to pre-process the features in order to increase performance. We will see that quality of the clusterization will be increased also.

Previously to show to the reader the final solution used, we will think about some possibilities that were discarded. It is clear that we can not use all the wavelengths and use them as features without some filtering or pre-processing.

Not all features are good enough to distinct between different clusters. In the case of spectral image we need to establish which properties make two points belong to the same cluster or not, that is the key to construct the classifier and also it will restrict the different solutions [ST03].

The definition of clustering leads directly with the definition of "cluster". If the feature vectors are viewed as points in the l-dimensional space, the clusters are defined as the areas of that space with high density of points. Clusters defined at this way are often called "natural clusters". To give us what clustering is we will see some formal definitions. Let have a set of objects X with certain features [ST03].

$$X = \{x_1, x_2, ..., x_N\}$$

We define cluster as a $m - clustering$ the partition of X into m sets $C_1, ..., C_m$ so that the next three conditions are meet:

$$C_i \neq 0,\ i = 1, ..., m$$

$$U_{i=1}^{m} C_i = X$$

$$C_i \cap C_j = 0, i \neq j, i, j = 1, ..., m$$

The vectors that belongs to a cluster C_i are more similar to each vector that belongs to the same cluster. Notice that one vector can only be part of one cluster C_i. In this thesis we use proximity measures.

A *dissimilarity* measure (DM) d on X is a function $d : XxX \to R$ where R is the set of real numbers, such that:

$$\exists d_0 \in \Re : -\infty < d_0 \leq d(x, y) < +\infty, \forall x, y \in X$$

$$d(x, x) = d_0, \forall x \in X$$

and

$$d(x, y) = d(y, x), \forall x, y \in X$$

If in addition

$$d(x, y) = d_0$$

if and only if

$$x = y$$

and

$$d(x, z) \leq d(x, y) + d(y, z), \forall x, y, z \in X$$

$$d(x,z) \geq 0$$

Finally, the minimum dissimilarity level for two given vectors is achieved when they are identical.
A *similarity* measure (SM) s on X is defined as:

$$s : XxX \rightarrow R$$

such that

$$\exists s_0 \in \Re : -\infty < s(x,y) \leq s_0 < +\infty, \forall x, y \in X$$

$$s(x,x) = s_0, \forall x \in X$$

and

$$s(x,y) = s(y,x), \forall x, y \in X$$

If in addition

$$s(x,y) = s_0$$

if and only if

$$x = y$$

and

$$s(x,y)s(y,x) \leq [s(x,y) + s(y,z)]s(x,z), \forall x, y, z \in X$$

s is called a *metric* SM
In our clustering algorithms we use Euclidean distance d_2

$$d_2(x,y) = \sqrt{\sum_{i=1}^{l} (x_i - y_i)^2}$$

The Euclidean distance is a metric for dissimilarity distance.

4.3 Categories of clustering algorithms

The classification that we will use in our software should be based on color differences rather than individual wavelengths differences. We will implement in the software three different types of algorithms. In this section we will present the two clustering algorithms that we use in the software [ST03].

Clustering algorithms provide different ways to set partitions in $l - dimensional$ space. The results differ depending on the algorithm and criteria used.
Clustering algorithms can be divided into the following categories:

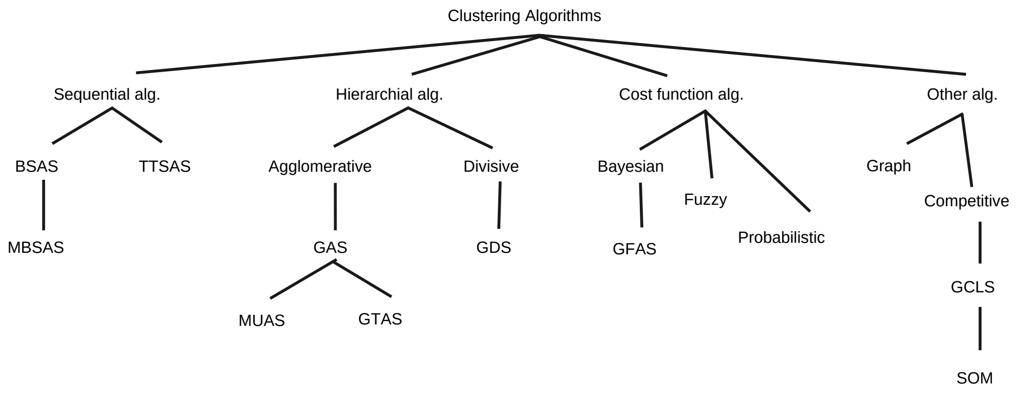


Figure 4.4: Clustering Algorithms

Sequential algorithms: produce single clustering. In this approach the vectors are used several times in the algorithm and the results depends on the order that the vectors are presented to the algorithm. These algorithms produce compact and round shaped clusters, although the results may vary depending on the distance metric used [ST03].

For that we will use the CIE L*a*b* space in order to find the features. Once the pixel spectral information is converted to L*a*b* coordinates, we have many information about the color scheme of the pixel with just three features.

Because the CIE L*a*b* only show information about visible wavelengths we need to add a fourth feature for the non visible range of wavelengths, the ones higher than 780 nanometers infrared-light. This fourth feature only is used when the software detects infrared wavelengths, so that for standard analysis within the visible range the calculations are much faster. Use of another dimension for infrared values slow the calculations.

The infrared data is preprocessed and we obtain a mean value of wavelengths from 780nm and above. If the pixel has infrared data and another pixel with the same L*a*b* values do not have infrared data, one will belong to a different group than the other. Final grouping depends also on the number of clusters of the final result selected by the user.

4.4 Hierarchical clustering algorithms

The hierarchical clustering algorithms can be divided in two approaches, agglomerative and divisive. We use in the software the agglomerative.

Agglomerative clustering algorithms: at the first iteration they produce a sequence of clusters. From that set of initial clusters they try to merge the clusters that match a certain criteria, for us the merging criteria is the Euclidean distance.

They can have some parameters to fine-tune the merging, in this thesis we use the type of distance to be used in the merging (mean, minimum, maximum), and the link type (single, complete). These algorithms are appropriate to detect elongated clusters.

Clustering algorithms based on cost function optimization. In these algorithms there exist a quantitated cost function. Usually the number of clusters is fixed to some static value, the algorithm tries to produce successive clustering and in each step the value of cost function is recalculated in order to fin the optimum value. They are also called iterative function optimization schemes [ST03].

4.5 Matrix Updating Algorithmic Scheme

Matrix Updating Algorithmic Scheme (M.U.A.S.) clustering algorithm rely on the hierarchical algorithms.

The general definition for hierarchical algorithms is [ST03]: Def: A clustering $\Re_1$ containing k clusters is said to be *nested* in the clustering $\Re_2$ which contains $r(< k)$ clusters, if each cluster in $\Re_1$ is a subset of a set of clusters that belongs to $\Re_2$ and also there exist at least one clusters from $\Re_1$ that is a proper subset of $\Re_2$.

Hierarchical clustering algorithms produce a hierarchy of nested clustering. The hierarchical algorithms produce N steps $N \leq$ (number of vectors) [ST03]. In every step a new set of clusters are formed based on the previous selection. M.U.A.S. Is an agglomerative algorithm so that it merges the clusters following a criteria.

The algorithm can be presented as follows:

```
begin
    N = Number_of_clusters;
    C = Void; //set of clusters
    for (i=1, i less or equal N, i = i + 1)
        //$\Re_0 = \{C(i) = \{x(i)\}$
        C = Select(C(i), x(i));
    end; //for
loop
$t = t + 1$
```

```
//From all possible combination of clusters in pairs $(C_r, C_s)$ $in$ $\Re_{t-1}$
// take the pair of clusters that has the following properties:

//
```

$$g(C_i, C_j) = \begin{cases} \min_{r,s} g(C_i, C_j), & if\ g\ is\ a\ dissimilarity\ function \\ \max_{r,s} g(C_i, C_j), & if\ g\ is\ a\ similarity\ function \end{cases}$$

```
// Define $C_q = C_i \cup C_j$ and generate a new clustering
// $\Re_t = (\Re_{t-1} - \{C_i, C_j\}) \cup \{C_q\}$

C = Mix_Clusters(C);

//Continue until all vectors lie in a single cluster.
if length(C) == 1
    break;
else
    continue;
end; //if
```

The linkage type defines the dissimilarity measure $d(C_q, C_s)$.

- $d(C_q, C_s) = \min\{d(C_i, C_s)\ ,\ d(C_j, C_s)\}$
- $d(C_q, C_s) = \min\{d(C_i, C_s)\ ,\ d(C_j, C_s)\}$
- Where element (i, j) of $P(X)$ dissimilarity matrix, is the dissimilarity $d(C_i, C_j)$ between clusters C_i and C_j.

The M.U.A.S. algorithm has some differences respect the general algorithm. We define X as a vector containing the similarity values $s(x_i, x_j)$ from each pair of $l - dimensional$ vectors.

The input is now a N x N dissimilarity matrix, $P_0 = P(X)$, derived from X. At each step the size of the matrix decreases because two clusters are merged into one.
The general algorithm for M.U.A.S. is the following:

```
//Initialization
//$\Re_0 = \{x_i, i = 1, ..., N\}$
C = void; //init set of clusters
N = Number_of_clusters;
Create_array (R, N);
//$P = P(X)$
P = Init_vector();
//$t = 0$
t = 0; //init counter

Loop
    $t = t + 1$ // increase counter
```

```
//From all possible combination of clusters in pairs $(C_r, C_s)$ $in$ $\Re_{t-1}$
// take the pair of clusters that has the following properties:
//
        $g(C_i, C_j) = \left\{ \min_{r,s} g(C_i, C_j), \quad \text{if} g \text{ is a dissimilarity function} \right.$

pair_clusters = Take_pair_clusters(C);

//merge $C_i, C_j$ into a single cluster $C_q$ and create $\Re_t = \Re_t - 1 - C_i, C_j \cup C_q$

C = Merge_clusters(C,pair_clusters);

//Define the next proximity matrix $P_t$ based on $P_t - 1$ matrix.
P = Define_proximity_matrix(C,P);

//Finish when $\Re_N - 1$ have only one cluster (all clusters merged).
if Length(C) == 1
        break;
else
        continue;
end;
```

These algorithms have a very high computational cost. At each level t there are $N - t$ clusters. The number of pairs of clusters that are going to be analyzed for merging at $t + 1$ step is:

$$\left(\begin{array}{l} N-t \\ 2 \end{array} \right) \equiv \frac{(N-t)(N-t-1)}{2}$$

And the total amount of operations neccesary to complete the algorithm is:

$$\sum_{t=0}^{N-1} \left(\begin{array}{l} N-t \\ 2 \end{array} \right) = \sum_{k=1}^{N} \left(\begin{array}{l} k \\ 2 \end{array} \right) = \frac{(N-1)N(N+1)}{6}$$

And that represents a total number of 560,000 millions of operations for an image of 190x100 pixels. Because this represents very long time to finish the calculations, the software includes a scaler. This scaler merges pixels in the order of n by n creating bigger squares and therefore reducing the resolution of the spectral image.

4.6 Hard Clustering Algorithm

The hard clustering algorithms are so called because a vector containing a set of features or properties belong only to a single cluster, unlike some probabilistic approaches. The membership coefficients $u_i j$ represents the probability that the vector of features x_i belongs to the cluster C_j.

In the hard clustering algorithms the coefficient u_ij is 1 if the vector x_i belongs to the cluster C_j and 0 for all the other clusters, $C_k, k \neq j$. It can be viewed as a special case of fuzzy algorithms. The cost function is no longer differentiable respect to θ_j:

$$J(\theta, U) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij} d(x_i, \theta_j)$$

4.7 Isodata Clustering Algorithm

The Isodata or K-Means is a very common clustering algorithm. It is a case of the generalized hard clustering algorithms using point representatives and setting the distance between vectors x and clusters θ_j to squared Euclidean.

$$J(\theta, U) = \sum_{i=1}^{N} \sum_{j=1}^{m} u_{ij} \left\| x_i - \theta_j \right\|^2$$

For this equation the cluster θ_j represents the mean vector of the $j - cluster$. The algorithm converges to a minimum of the cost function. The results of this algorithm are compact clusters but not always with a absolute minimum cost function.

The general algorithm for the Isodata clustering algorithm is:

```
begin
//Select a random estimation $\theta_j(0)$, $j = 1, ..., m$
T = Generate_random();

Loop
    //For $i = 1$ to $N$ find the better representative $\theta_j$, for $x_j$
    for i=1;i less or equal N //N size of T; i=i+1
        //set $b(i) = j$
        b(i) = j;
    end; //for
    // For $i = 1$ to $m$
    for i=1; i less or equal m; i=i+1
        // Update parameters: $\theta_j$ = mean of the vectors $x_i \in X, b(i) = j$
        Update(T)
    // end For
    end; //for
        // Repeat until no change in $\theta_j$ during two iterations
    if (Nochange(T) and TwoIterations()
        break;
    else
        continue;
    end;
```

The advantage of this algorithm relies on the computational simplicity. For our task it is very important the speed in the calculations and we should give the user fast response times. This algorithm is suitable for spectral images of all resolutions and will give results in few seconds.

4.8 Segmentation Based on L*a*b* Space Data

The previous segmentation algorithms were based on clustering algorithms. The segmentation based on L*a*b* space is another segmentation technique used in our software. It give us information about pixels having similar hue and saturation.

This method creates only one cluster and it is useful to distinct between pixels that have similar color properties and the ones that have totally different color. The luminance information is used in order to highlight regions with similar properties to the selected pixel.

For this algorithm the user needs to select one dot of the image. After that the color coordinates of the dot appears in the L*a*b* diagram. The algorithm is very simple:

```
begin
    // User selects one dot $pixel_1$ from the RGB picture
    pixel = getpixel(image);

    // Find CIE L*a*b* coordinates
    coordinates = calculate_lab_coordinates(pixel);

    // Highlight with a black dot the position of the dot in the CIE L*a*b* coordinates
    draw_coordinates(coordinates);

    // Find $pixel_i$ CIE a*b* coordinates
    image_coordinates = get_ab_coordinates(image);

    // For $pixel_i$ all pixels in the image
    for i=1;i less or equal to Total_number_pixels; i = i + 1
        // Measure Euclidean distance $d$ between $pixel_1$ and $pixel_i$
        distance = Euclidean (pixel(i), pixel);
        // If distance $d$ is lower than "threshold a*b*" $t$ set output pixel as black
        if distance les_or_equal than threshold_ab
            set_black(image(pixel(i))); // total coincidence
            // Otherwise create gray pixel with L* data and "threshold L*"
        else
            set_grey(image(pixel(i)), threshold(L));
        end; //if
    end; //for
```

As the reader can see this algorithm detects similarities on colors and the luminance data is used only in the case that the distance d between the pixels in a*b* coordinates is higher than a given threshold.

Because we give more importance to the color information in the detection it is possible that parts of the image with different luminance but same color belongs to the cluster. It is very common that dark areas of the image can have the same color properties (spectral shape) than bright ones.

With the use of the "threshold L*" and "threshold a*b*" parameters in the software GUI, it is possible to fine-tune the detection. The gray levels of the resulting image refers to pixels that not match the given criterions but are close to them.

4.9 Segmentation Based on Mean Spectral Data

While the clustering algorithms provide a good way to distinct between different color shades, they are not suitable for detecting other aspects.

Because of the nature of the spectral to L*a*b* space transformation, we loose a lot of information about the spectra. Indeed the results of the application of the clustering algorithms work the overall of the color characteristics of the image. In the case that we need to detect very specific characteristics within the spectral data we should use a slightly different approach.

In the segmentation based on the mean spectra we try to find areas of the image with similar spectral data. There are some approaches to achieve this goal. The spectral can be viewed as a curve with a specific shape.

The similarities of two different spectras could be viewed as similarities in shape or in absolute values. There are many ways to decide if two spectral shapes are similar or not. Distance d between two spectra can be viewed as the linear difference between two spectras or difference between the overall shape.

We decided in this software not to use shape comparison. The reasons are that the shape represents the color information of the pixel and that can be obtained in a more easy way from the color space diagrams. So the shape comparison will take long time and the results may not be as good as the color and hue comparison.

There is a case that the shape comparison could improve the results of other algorithms used in this thesis. The existence of a narrow peak within the wavelengths of the spectra may not be detected properly with the other methods. In fact the spectral to CIE L*a*b* transformation loses data about the spectra, the color matching functions give the same values in the case that a narrow peak exist in the spectra or not.

In order to detect this case it is needed to apply a totally different method prepared to detect narrow peaks. In this thesis we use spectral differences by mean. The user selects a square and the mean of the spectra of all the dots inside the square are compared with the dots of all the rest of the image.

The differences in the wavelengths are sum and the final value is compared with the given threshold to create the segmentation.
The algorithm can be shown as follows:

```
begin
    // User selects one dot $pixel_1$ from the RGB picture
    pixel1 = getpixel(image);

    // User selects one dot $pixel_1$ from the RGB picture
    pixel2 = getpixel(image);

    //From the square created by $pixel_1$ to $pixel_2$ obtain the mean value of all spectras
$mean\_spectra$
    mean_spectrum = mean_spectrum(pixel1,pixel2)

    // For $pixel_i$ all pixels in the image
    // Find $pixel_i$ spectra
    for i = 1;i less or equal last_pixel; i = i + 1;
        //Compare with function $f$ $pixel_i$ spectra and
        //$mean\_spectra$ and obtain distance $d$
        distance = distance_spectra(mean_spectra, spectra_image(pixel(i)));

        //If distance $d$ is lower than "threshold spectra" set output pixel as black
        if distance is_less_or_equal than threshold1
            image(pixel(i)) = black;
        // Otherwise create gray pixel with threshold two, three and
        //four times "threshold spectra" as distance
        else
            image(pixel(i)) = grey_level(threshold1, threshold2);
        end; // if
    end; //for
```

As we can see in the algorithm we have a 4 level greyscale resulting image. The purpose of this tolerance is to highlight the pixels that do not match the given criterion but are close.

Chapter 5

Spectral Software manual

For this thesis we developed three different software. The Segmentator, Spectral Painter and Spectral Data Reader. We will present the features and characteristics of all with some examples of use.

5.1 Segmentator: Manual

The Segmentator (Fig. 5.1) is a software developed in order to analyse spectral images. It is done in MATLAB 2006a. It is capable of loading, saving, cropping and some other features that will be explained in this chapter. The key feature is the segmentation. This segmentation is done using three different algorithms, each of them capable of highlight different properties of the image.

The Segmentator can load spectral images in the following formats, "mat" Matlab files, "dat" binary files from ImSpector, "aix" MUSP files and "spb" Spectral Binary Files.

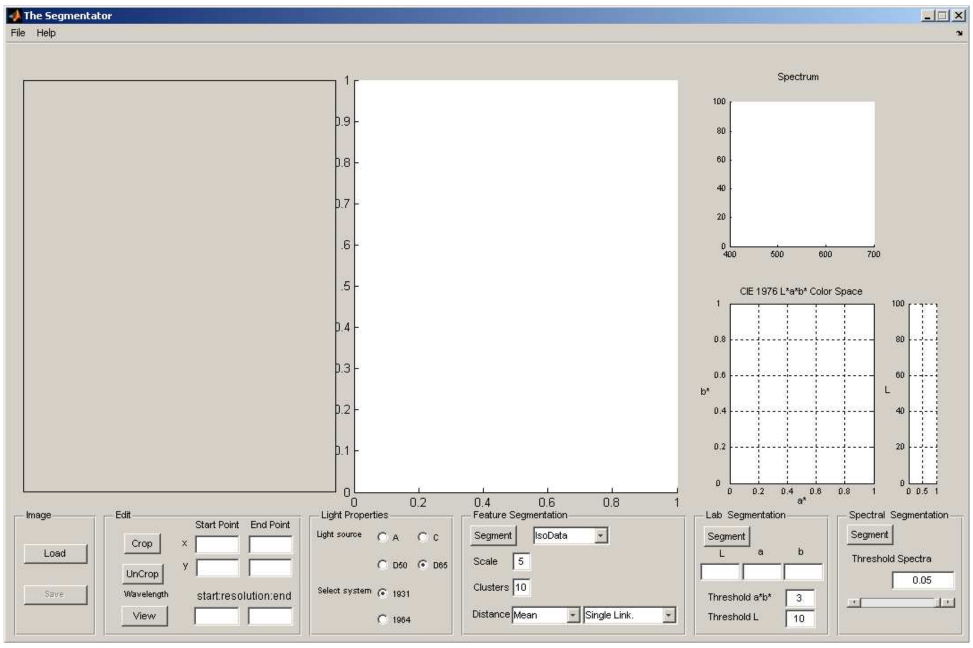
The saving of the spectral images is done in "mat" file format. The user interface is as simple as possible, there are only few parameters for the algorithms and the GUI is protected so that it is not possible to introduce wrong values that could crash the program.

The software developed for this thesis have two main parts. On the top we can see the original image aspect in sRGB format. We can find also information about the spectral of the selected pixel of the image, the L*a*b* coordinates of the selected pixel and the resulting image from the algorithms application.

At the bottom we can find the controls of the program, we can specify the value of the parameters of the algorithms and also make use of some basic editing tools. The aim of this layout is that the visual information is on the top, and the technical information is on the bottom.

When the original spectral image is loaded, it is converted to sRGB, aplying the correct color system and the selected iluminant. The RGB value of the pixels of the image are recalculated if the color system or the iluminant are changed.

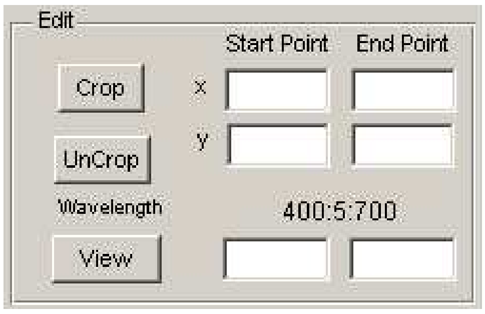
At the same time the L*a*b* coordinates are recalculated for all the pixels within the



Light source: ○ A ○ C ○ D50 [✓ D65]
Select system: [✓ 1931] ○ 1964
IsoData
Scale 5
Clusters 10
Distance Mean, Single Link.
Threshold a*b* 3
Threshold L 10
Threshold Spectra 0.05

Blank Segmentator

Figure 5.1: Blank Segmentator



Edit Options

Figure 5.2: Edit Options

image. This calculation involves a high computational cost in terms of time. It can take some seconds to make the calculations for all the pixels and update the L*a*b* coordinates and sRGB image in the user interface. This characteristic is included y default because it is needed to know the actual values of the pixels to apply the segmentation algorithms.

This software has some basic editing tools built in (Fig. 5.2). In the "edit" area we can crop the image. For cropping we need to specify two points that will define the area or square to be cropped (Fig. 5.3). For this we should simply click with the mouse button in the sRGB image area to define the two points.

We can crop the image all the times we need recursively, the image will be smaller each time we crop. The "uncrop" button will return to the original full sized image.

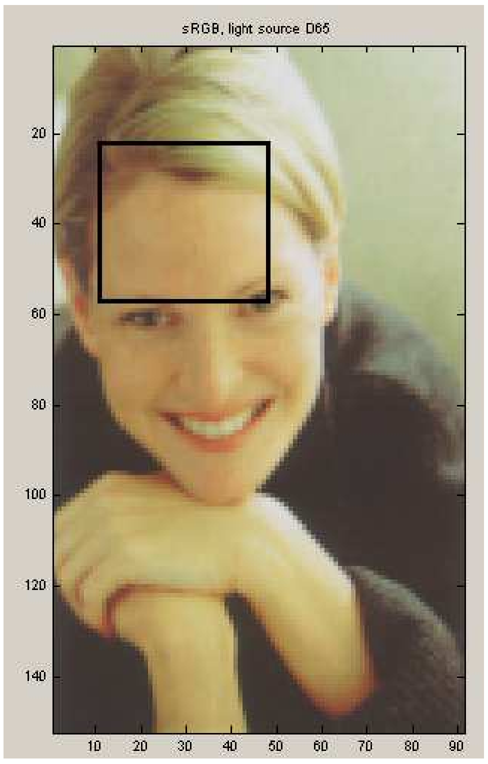


Figure 5.3: Cropping Square

The cropping is useful in cases that we are only interested in one area or region, the "save" button will save the cropped image. Cropped images have less number of pixels than the original full sized image. This mean less work for the algorithms and the calculations will be faster.

Another useful option in the "edit" menu is the "view" button. With the "view" button we can visualize any range of wavelengths of the spectral image. As we can see in the example image (Fig. 5.4), we can find very interesting data in some wavelengths like horizontal bands that can not be viewed using all the wavelengths.

In this example we visualized only the range between 400 nanometers and 405 nanometers. We can appreciate some horizontal bands that can not be viewed in the full wavelength image. They are probably caused by the spectral camera while scanning the image.

In the "Light Properties" menu we can change the light source and color system to be applied to the spectral image. The sRGB representation and the L*a*b* coordinates are affected by light properties changes. The results of the segmentation can slightly be affected by changes in the light properties.

In the central part of the user interface we have the resulting image from the application of the segmentation algorithms. It is a greyscale image that represents the clusters and areas highlighted by the algorithms.

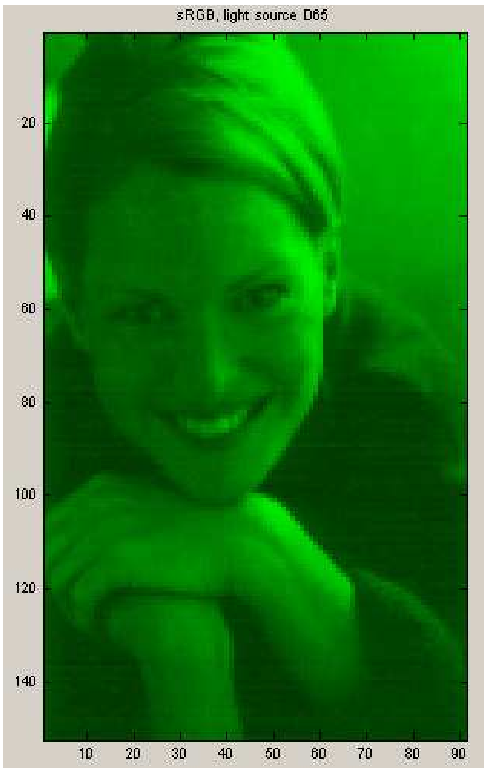


Figure 5.4: Image Visualization from 400nm to 405nm

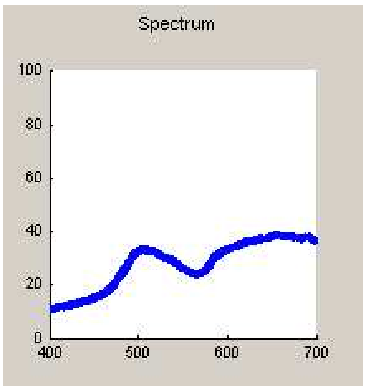


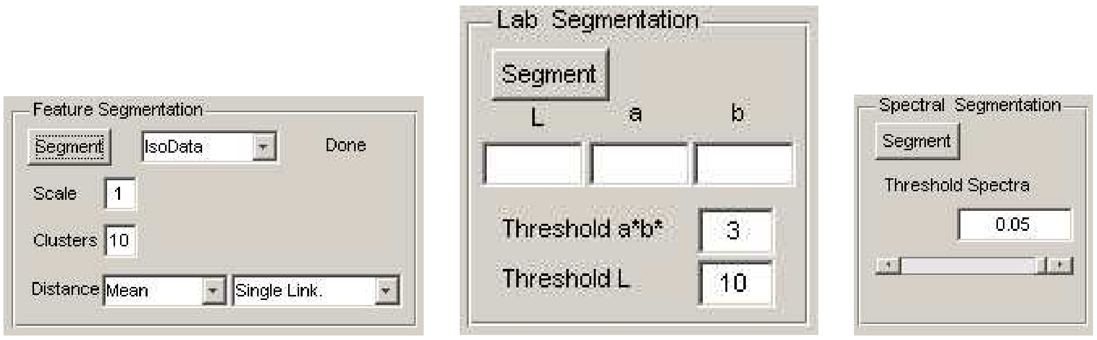
Figure 5.5: Spectrum from one image dot

In the right part we have the spectrum from the selected dot (Fig. 5.5) and the L*a*b* coordinates. The spectrum is uptated when the user selects any point from the RGB image.
The list of features from left to right are:

- Load and Save buttons
- Edit properties, start and end point to define the square (for later cropping or seg-

mentation), cropping button and uncrop button, and finally the range of wavelength to visualize.

- Light properties, where the user can select the proper illuminant.
- Feature Segmentation options, mainly this are the options to select the clustering algorithm with few options for each algorithm.
- Lab Segmentation properties with the coordinates of the reference dot and threshold for chromacity and luminance.
- Spectral Segmentation, this part only needs one parameter that is the threshold of similarity for the spectral comparison.



IsoData
Done
Scale 1
Clusters 10
Distance Mean Single Link.

Threshold a*b* 3
Threshold L 10

Threshold Spectra 0.05

Figure 5.6: Feature Segmentation, L*a*b* Segmentation,Spectral Segmentation

5.2 Segmentator: Segmentation Based On Clustering

The first segmentation technique is based on clustering algorithms. The in the ”Feature Segmentation” menu we can find some properties to select the clustering algorithm and other properties. There are two clustering algorithms available, M.U.A.S. and IsoData, both already explained in Chapter 3.

For both algorithms we have the option to scale the original image. We should give a factor to scale and the software will internally resize the original image grouping the pixels. A factor of 2 will merge pixels of squares of 2x2, making a resulting image of half the size of the original and 4 times less pixels. This is useful in the case of big images. The time to compute the results can be very long if he amount of pixels is too high.

For the IsoData clustering algorithm we can specify the number of clusters to be used. The algorithm will try to divide the image in the specified number of clusters. Large amount of clusters will involve more time to compute. The resulting image is a greyscale image with the clusters having different gray values.

The M.U.A.S. clustering algorithm, apart from the number of clusters, type of distance and link type. The type of distance defines if the distance between clusters will be the distance between the mean value of the cluster, the minimum distance between the clusters or the maximum distance between the clusters. Single link favours elongated clusters whereas complete link favours compact clusters.

5.3 Segmentator: L*a*b* Segmentation

For the L*a*b* segmentation the user needs to specify the coordinates. It is possible to just select the L*a*b* coordinates by clicking on the sRGB image. The software will show the coordinates in the L*a*b* coordinates area with a black dot.

It is possible to edit these values manually. The algorithm works with the information about color and luminance. The clustering is made with color information and luminance taking into account the given thresholds for the color and luminance coordinates. The results is shown in the central part of the interface as an greyscale image.

This algorithm is pretty basic, the computational time is mainly used to calculate the L*a*b* coordinates and the distances between the selected coordinates and the rest of the image points coordinates.

5.4 Segmentator: Segmentation based on spectral data

The user has only one option to set this algorithm. First the user should select two different points of the image. A black square will appear and the user needs to set the threshold. The software will calculate the mean value of all the spectras within the black square. After that the software will calculate the differences between the mean spectra and the spectra of the rest of the pixels.

In the resulting image dark areas corresponds to areas where the pixels are quite similar to the mean spectra selected. White areas corresponds to regions where the pixels are not similar to the mean spectra selected.

Notice that the mean spectra could not match with the spectra of all the pixels inside the selected square. For example if we have a square where half of the pixels are red and half of the pixels are blue, the mean spectra will correspond to a purple color. Depending on the threshold selected the pixels inside the square could not have similar spectra than the mean spectra of all of them. In this case the pixels inside the selected square will appear in the resulting image as white or grey.

5.5 Spectral Painter: Software manual

The Spectral Painter is a piece of software developed for this thesis. The aim of this software is to create spectral images. The spectral images can have different ranges of wavelengths, but the software limits the range of wavelengths to be draw from 400nm to 700nm, other wavelengths will be discarded.

With the use of the Spectral Painter it is possible to test different segmentation algorithms by using artificial spectral images. By using artificial spectral images the user can compare results of the segmentation algorithms.
The Spectral Painter can draw squares in spectral images. The color used for drawing can be created with the use of slider. The sliders represent wavelengths from 400nm to 700nm in 10 nanometer steps, 31 sliders in total. The values are interpolated in order to draw a spectral image with the range selected by the user. Higher wavelengths than 700nm and lower than 400nm will be draw with plain black, all wavelength values will be equal to zero.

The spectra can be also read from a given picture, the user can pick up a spectra just clicking with the mouse over the spectral image, after pushing the "Read" button on the "Spectra" panel, the sliders will adquire the spectral from the image.

To draw a square in the Spectral Image, the user should select two pixels of the image to define the square. The user can select the pixels both with the mouse or specifying them with a number in the corresponding text box in the "Edit" panel. When the user push the "Draw" button in the "Edit" panel the squere will be draw in the image. It is possible to mix the existing data with the new data. By selecting the "Soft" button in the "Edit" panel, the result will be a mix between the previous spectral data in the spectral image and the new spectral selected by the user.

In the "Edit" panel the user can select the "Undo" option. There is only 1 step back available, so the user can go back to the previous state of the image.

It is possible also to draw L*a*b* coordinates of the image, change the illuminants and load and save existing images.

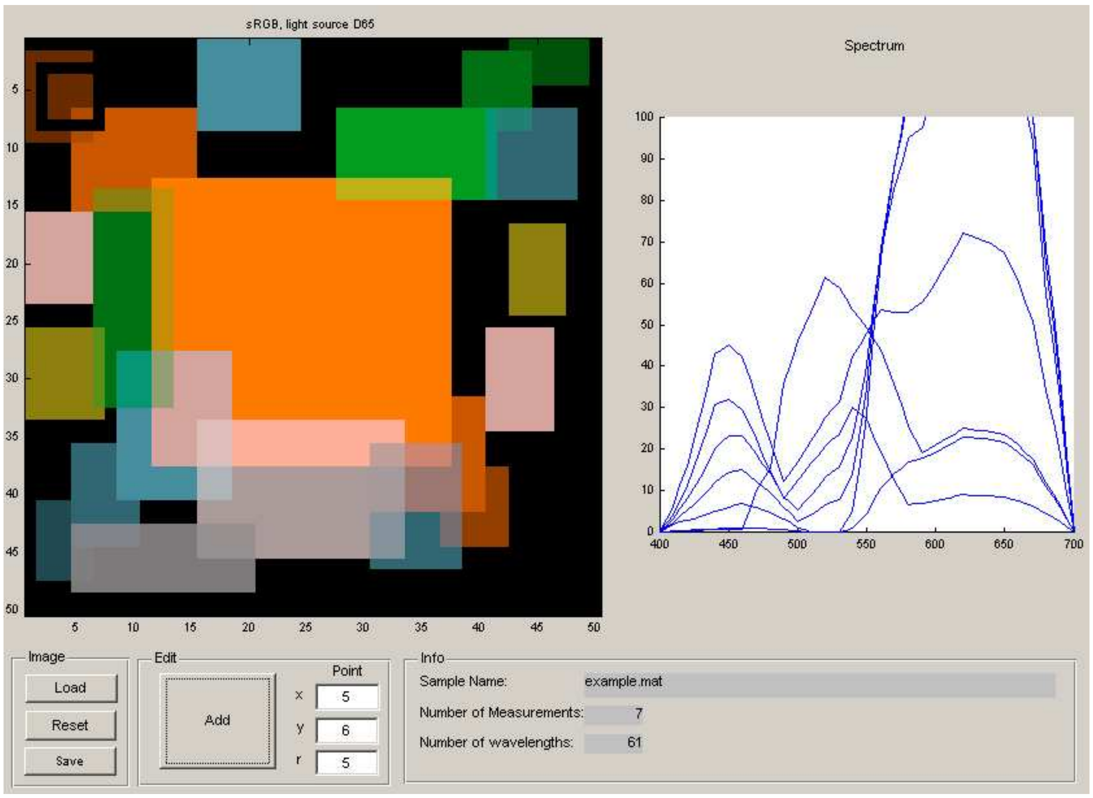
5.6 Spectral Data Reader: Software manual

Spectral Data Reader is a software developed for this thesis. The aim of this software is to study the spectral characteristics of the images. When the user clicks with the mouse over the image, the software obtain the spectral data of the pixels within the square defined by the pixel coordinates and the square size. The L*a*b* and sRGB information of the pixels are also calculated, and the output is stored in an array of values. The user can select as many areas and he wants, each area will be saved in an array with the following format:

```
    Results = Array of Result

    Result = {mean_spectra,
                  mean_L_coordinate,
                  mean_a_coordinate,
                  mean_b_coordinate,
                  mean_R_value,
                  mean_G_value,
                  mean_B_value}
```

This software recollects the data from the spectral images in an easy way. For this thesis it was needed to make a comparison test between different paper sample. Each paper sample has certain areas that should be compared to other paper images.



Point
x: 5
y: 6
r: 5

Sample Name: example.mat
Number of Measurements: 7
Number of wavelengths: 61

Figure 5.7: Spectral Data Reader

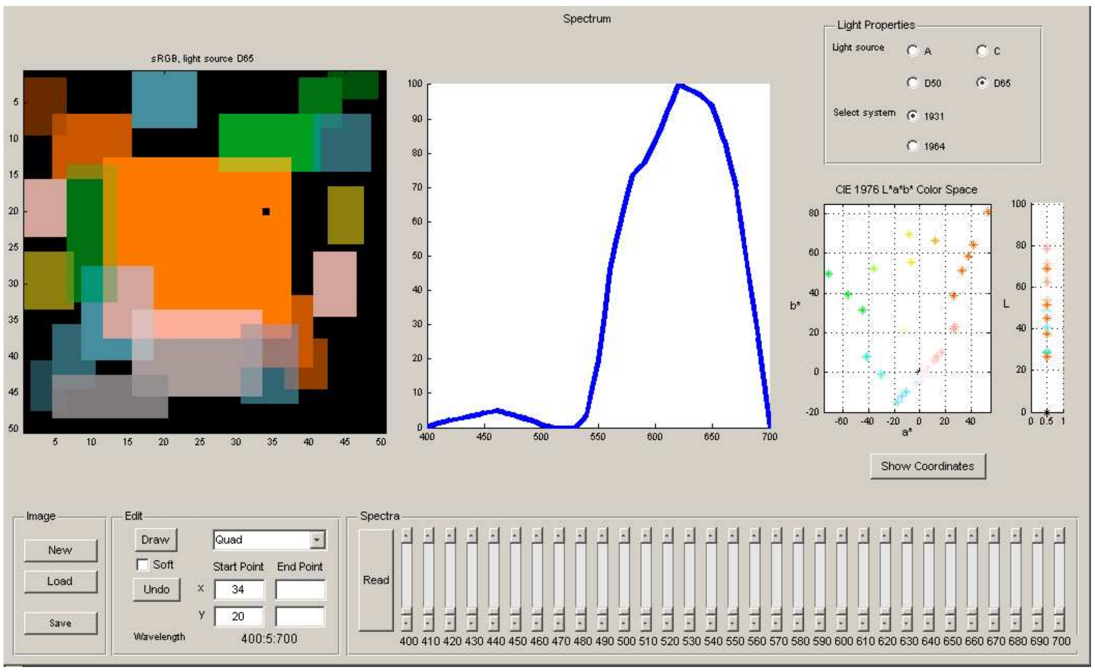

Light source: A C D50 [✓ D65]
Select system: [✓ 1931] 1964
Quad
Start Point X 34, Y 20
Wavelength 400:5:700

Figure 5.8: Spectral Data Painter

Chapter 6

Results Of The Experiments With The Software

6.1 Segmentation Results

The Segmentator has three different segmentation algorithms. Two of them use color differences in the pixels to make the clusterization and the last one compare spectras from different pixels.

The M.U.A.S. algorithm gave quite similar results than ISODATA algorithm but the running time of the process is much longer with M.U.A.S. taking more than 50 minutes with a 190x100x61 spectral image (height, wide, wavelengths). We used a 4, 6 and 10 clusters configuration in scale 1 (Fig. 6.1 Fig. 6.2), no resizing of the image. The color of the skin of the girl is detected and the hands as well.

The dark clothes are detected in all the configurations in a similar way. The hair is the most problematic area. There are several colors involved from dark ones to bright, and from brown to yellow and white. The background is detected in a similar way in all the configurations.

It is interesting that the algorithms did not difference between dark regions and bright regions as far as they belong to the same hue. This is an important feature because this way the shades will not affect the segmentation and the final results give clear areas. The scaling option is only useful to obtain previews of the segmentation results or to use with M.U.A.S. algorithm (Fig. 6.3 and Fig. 6.4).

In the L*a*b* segmentation we selected two areas. The first one was the dark clothes of the girl. The aim of this is to find the result of the segmentation algorithm with dark areas. Because it try to find areas with similar hue, we obtain that the dark clothes are selected in the resulting images (Fig. 6.5), but also some regions like the dark areas of the hair. If we increase the distance in L*a*b* coordinates we find that almost all the image except the skin of the girl is highlighted. This is because the hue of the skin is quite different in hue than the rest of the image.

If we select the mouth of the girl, we obtain clearly defined the area of the mouth and only a some small areas near the hand and head (Fig. 6.7). If we increase the distance the selected area extends itself and shows the rest of the skin in the hands and in the face (Fig. 6.8).

For the spectral distance based segmentation we find interesting results. The selected area was in the mouth of the girl. The algorithm uses the selected by the user square to find the mean spectra of the area. After that it calculates the distance from the mean spectra to the spectra of every point. The threeshold distance changes the darkness of the result. As we can see in figure (Fig. 6.9), if the distance is small we obtain only few areas highlighted by the algorithm.

The result is a 4 grey level image from white to black (Fig. 6.10). Even in the mouth we dont obtain totally black areas. That is because we use the mean spectra and that spectra can be different from the single spectra of the pixels in that area. If we increase the distance we obtain darker results with more black areas. The results are quite different from the L*a*b* segmentation algorithm.

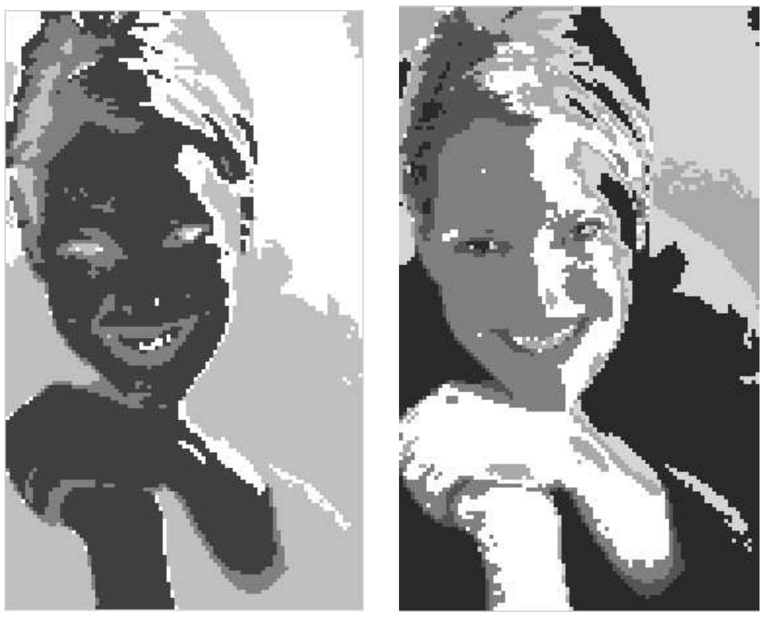


Figure 6.1: IsoData Clustering Algorithm 4 Clusters - 6 Clusters

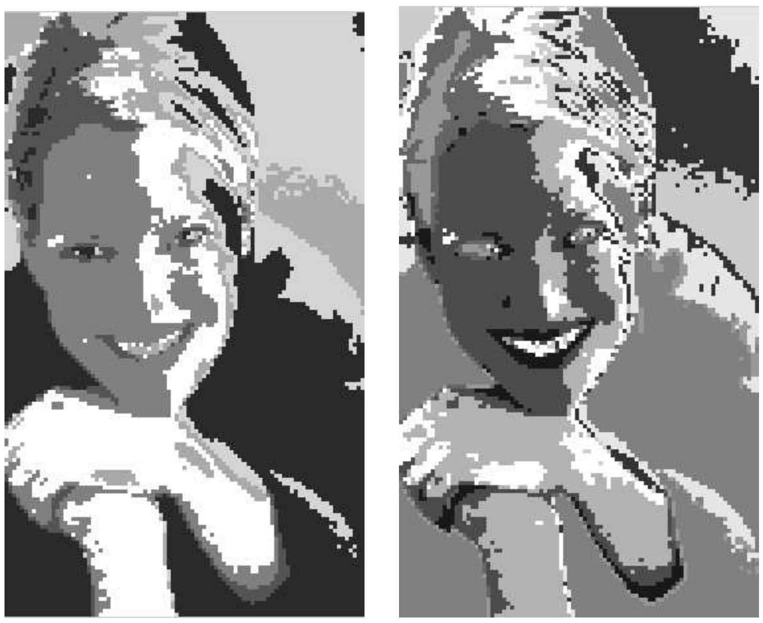


Figure 6.2: IsoData Clustering Algorithm 6 Clusters - 10 Clusters

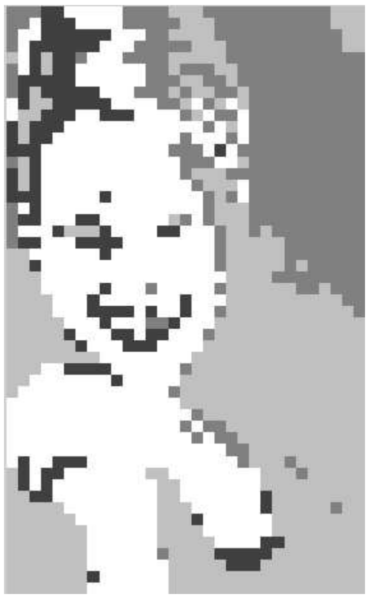


Figure 6.3: IsoData Clustering Algorithm 4 Clusters - Resize 3

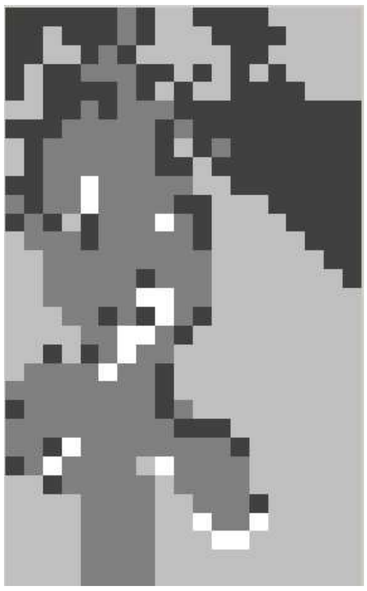


Figure 6.4: M.U.A.S. Clustering Algorithm Maximum Distance - Resize 5 - Complete Link

6.2 Paper Samples Analysis

The paper samples are sets of printing samples with different quality. The main differencesbetween the samples is the paper where they are printed and the printing density. We study different paper samples:

- MWC Gloss 65 grams

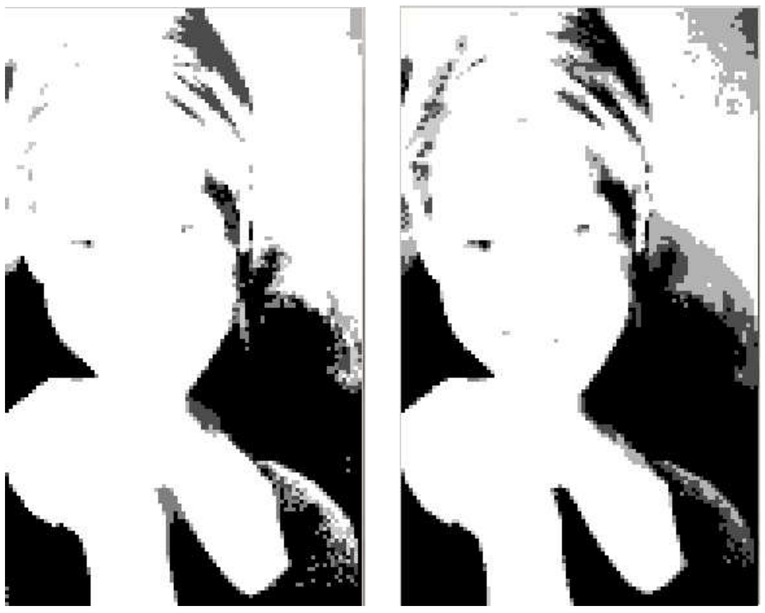


Figure 6.5: LAB Segmentation-Dark Clothes Area-Distance 3 and 5

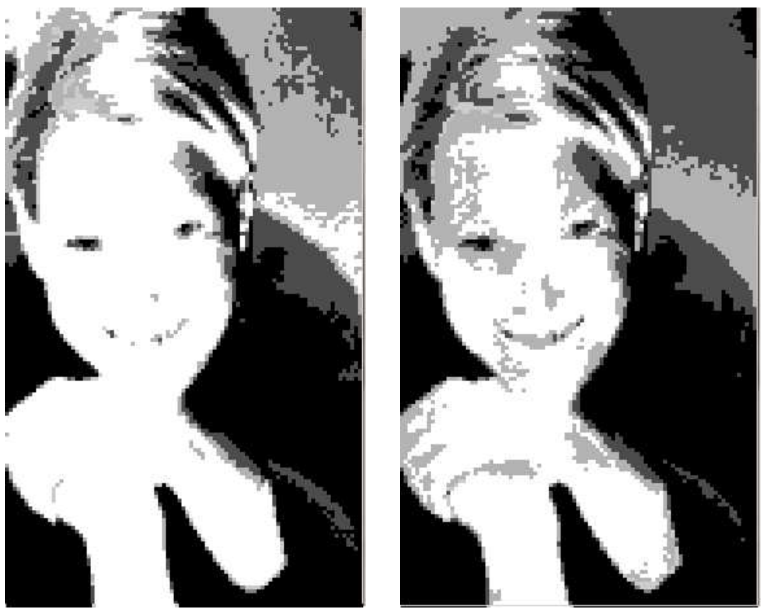


Figure 6.6: LAB Segmentation-Dark Clothes Area-Distance 8 and 10

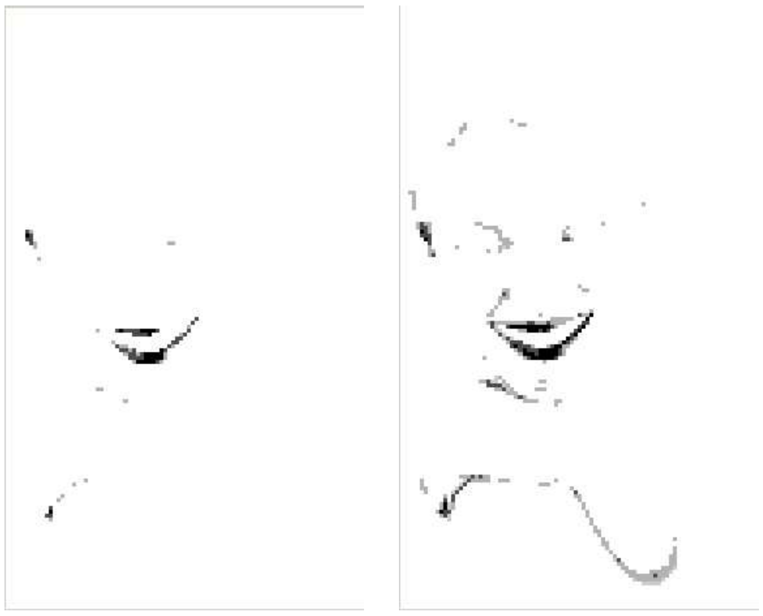


Figure 6.7: LAB Segmentation-Mouth Area-Distance 3 and 5

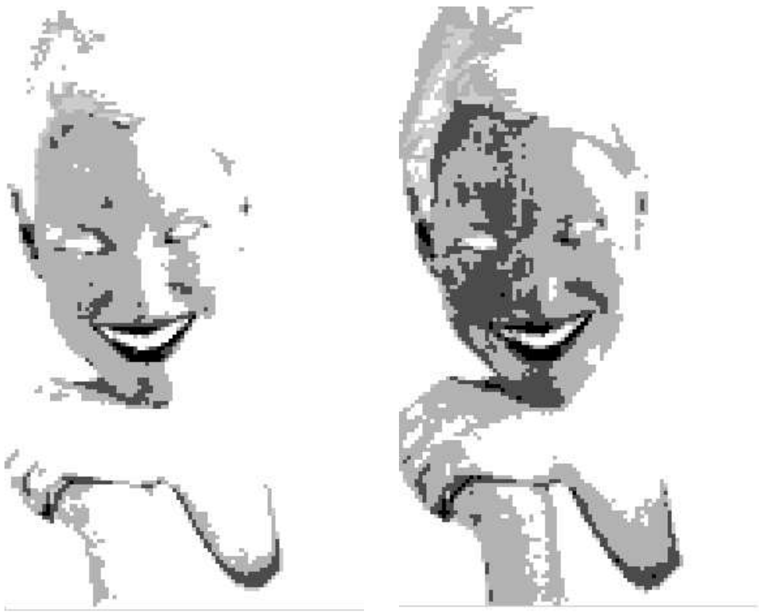


Figure 6.8: LAB Segmentation-Mouth Area-Distance 8 and 10

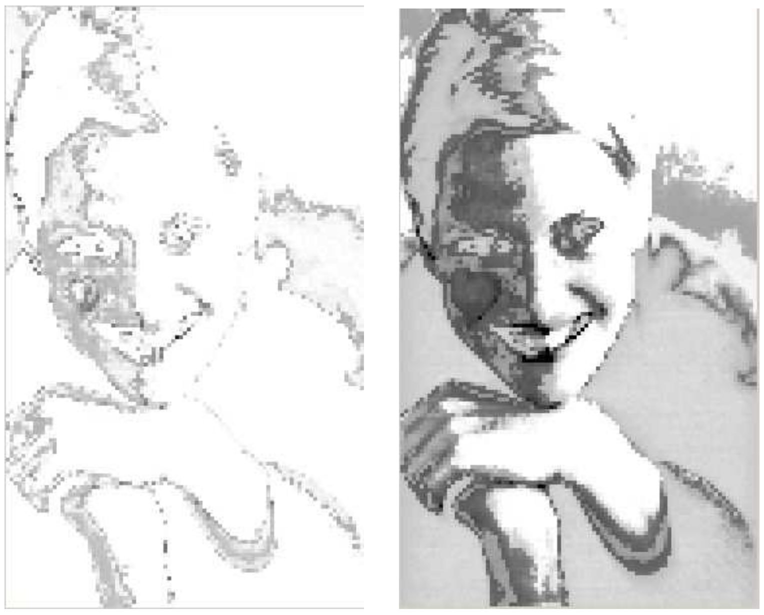


Figure 6.9: Mean Spectra-Mouth Area-Distance 0.01 and 0.05

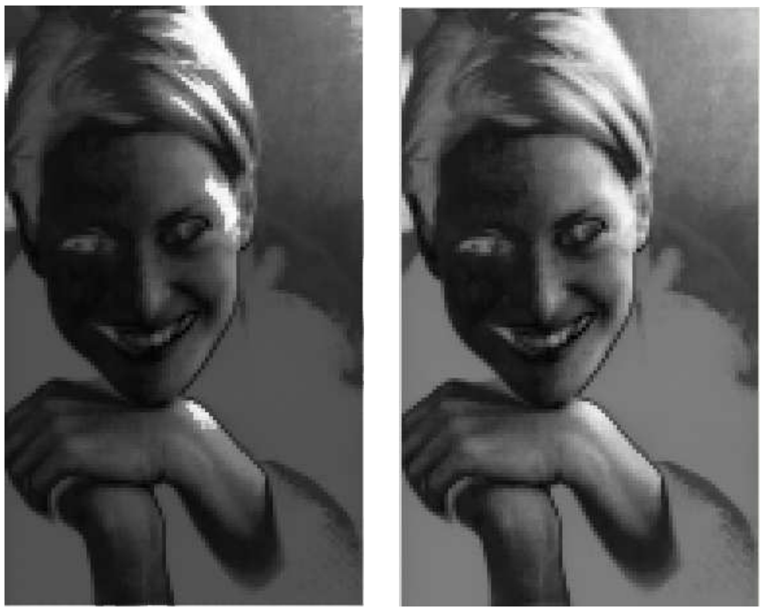


Figure 6.10: Mean Spectra-Mouth Area-Distance 0.5 and 1

- MWC Gloss 80 grams
- MWC Gloss 2 80 grams
- WFC Gloss 115 grams
- MWC Gloss 115 grams
- MWC Silk 80 grams
- MWC Silk 2 80 grams
- MWC Gloss 3 80 grams
- Hi-Brite LWC 80 grams
- Hi-Brite LWC 60 grams
- Std LWC 60 grams
- Std LWC 48 grams
- Std LWC 54 grams
- Std LWC 39 grams
- MFC 54 grams
- SC 56 grams
- UWF 80 grams
- WFC Matt 80 grams

The acronym means:

- WFC = wood free coated
- MWC = medium weight coated
- LWC = light weight coated
- MFC = machine finished coated
- SC = super calendered
- Std = standard
- Hi-brite = high brightness
- UWF = uncoated woodfree

For each paper there are three printing qualities depending on the density of the printing ink. We obtained spectral data from lower density printing, target density printing and higher density printing. The printer device uses four types of ink to generate the images, black ink, yellow ink, cyan ink and magenta ink [SCB88] [CEY84].

The samples are composed of several parts with different quantities of the inks, using one ink or mixing them to generate color areas or grey areas. In order to measure the quality of the different printing densities we will study a specific area composed of different levels of pure ink (Fig. 6.11), they are composed of one single ink without mixing them at all.

The objetives of this study are to compare lower and higher density printings respect te target density. From this study we can give some hints about the optimal quality for different paper samples and also determine printer calibration factors that can be modified to increase the quality.

With the use of the software "Spectral Data Reader" developed for this thesis, we obtained the spectral and L*a*b* values from the paper samples. We have 18 paper samples with 3 different printing qualities each one and we obtained information from yellow area, magenta area, cyan area and black area. Each area has 12 parts that represent different levels of saturation from 10parts measured is 2592.

We created three different graphs in order to present the data. The first one is made by dots that represents the absolute value of the distance from one area mean spectra to the analog area in other sample. So the level 100 represents the target density. Blue dots represents the distance between mean spectra of one area with lower density respect the analog area in the target density. The same for green dots but they represents distance from higher density to target density.

The second graph represents Euclidean distance in *a*b coordinates from lower (represented with blue lines) and higher (represented with red lines) respect the target density.

The last graph represents Euclidean distance in L coordinates from lower (represented with blue lines) and higher (represented with red lines) respect the target density.

Some interesting conclusions can be read from the results. Looking the the spectral comparison we can see that the difference of magenta, cyan and yellow spectra is minimal. The spectra of all densities are quite similar The difference between densities remains constant at about 10% respect the target.

There are big differences in the black ink. From 70can see a huge difference in the spectra. The difference increases linearly from 10different densities while color ink cyan, magenta and yellow is less affected by the density.

Studying the other two graphs we can appreciate that the distance in a*b* coordinates for the black is almost zero, while for the other inks the difference is noticeable. For the *a*b* distance comparison we can say that the yellow density affects more to the final result than for the rest of the colors. One reason can be the filtering affect of the white paper for different inks [SCB88] [CEY84].

Looking to the distance in L coordinates, we can compare the brightness of the different densities. Higher densities are darker than target density and target density is darker than lower density. But in this study we need to take into account the paper quality, weight and reflectance. For some paper samples this general increase in brightness is not so noticeable because the own paper reflectance.

We obtain some conclusions. There is no need for using target density instead lower for the following cases:

- Std_LWC_54g
- MWC_gloss_80g
- Std_LWC_48g
- MWC_gloss_2_80g
- WFC_gloss_115g

There is no need to use higher density instead of target density in the following cases:

- Std_LWC_60g

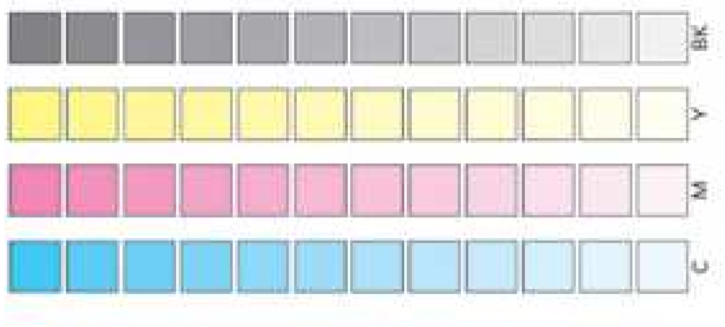


Figure 6.11: Paper Samples Color Areas

Chapter 7

Conclusions

Spectral images can provide much more information than RGB images. The study of spectral images can be done from different points of view. Segmentation techniques using spectral images can highlight areas or regions that can not be selected with the use of RGB images. The use of spectral data for segmentation involves preprocessing in order to find appropiate features and decrease the time to run the algorithms. IsoData clustering algorithm provides an efective way to make segmentation and obtain results fast. For future work it could be good to use different matching functions. The use of dynamic number of matching functions dividing the spectra in areas can give us a reasonable number of features for the clustering algorithms. L*a*b* coordinates based segmentation provides a good way to find areas with similar hue and luminance respect the desired point. Segmentation based on spectral distance can highlight regions with similar spectra to the selected one. Using spectra and L*a*b* coordinates it is possible to analyze and compare quality of paper samples in order to find quality criterions and hints that help in the printer devices calibration. Different paper affect specially the luminance of the final results. Ink density affect in the regions composed by black ink. The biggest differences in hue are in the yellow rather than magenta or cyan. To study more in depth the quality of the paper samples it is needed to compare mixed inks due to the filtering effects to the final color and different printed dot location. The use of mixed ink can give us more information about how different amounts of ink act as a filter of the paper color and other ink color.

Appendix A

Paper Samples Comparison Graphs

In this comparison we have three different graphs. The first one is a comparison of the spectral data from different areas. We measured certain regions of the paper color charts with black, yellow, magenta and cyan ink.
The amount of ink goes from 10% to 100%. So in the graphs in the x axis the position 0 to 12 represents black ink areas from 10% ink to 100% ink. The same percentages for yellow ink from 13 to 24, magenta from 25 to 36 and cyan from 37 to 48.
The other two graphs represents the Euclidean distance from lower density and higher density respect the target density paper samples in Lab space.

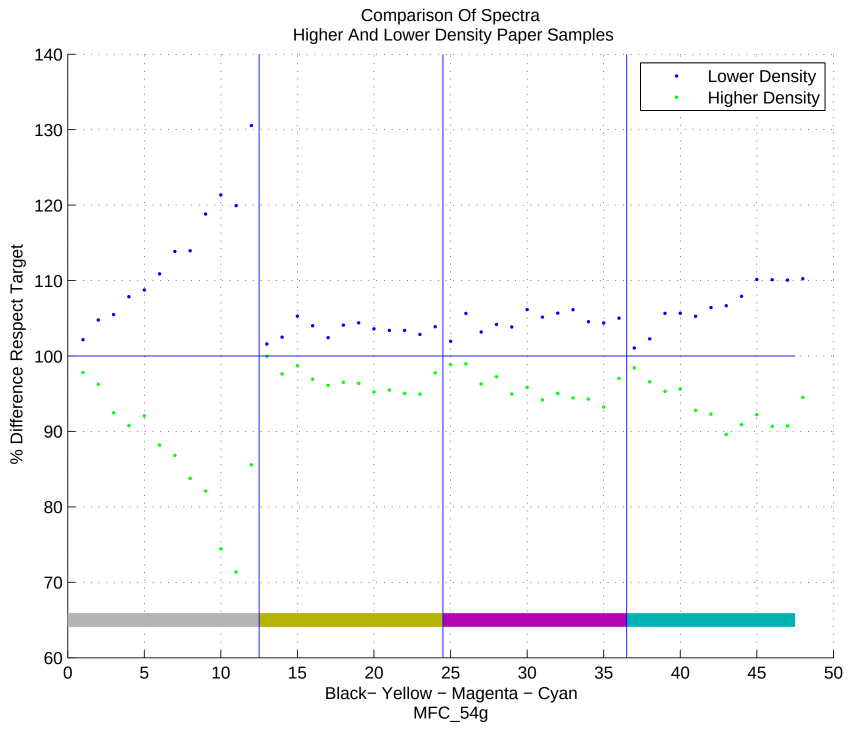


Figure A.1: Spectra Comparison - MFC 54g

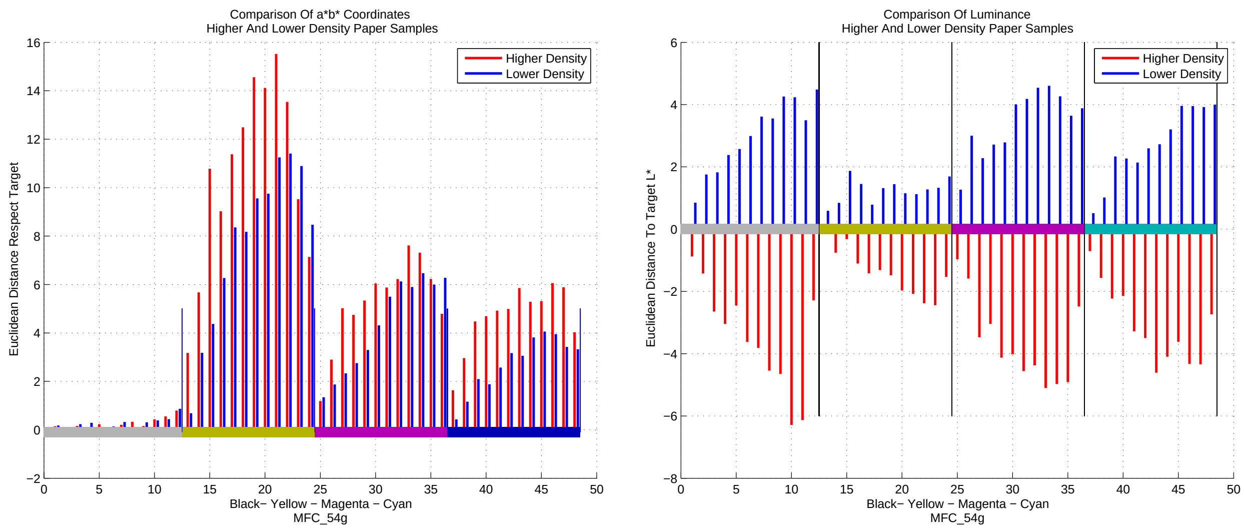


Figure A.2: L*a*b* Coordinates Comparison - MFC 54g

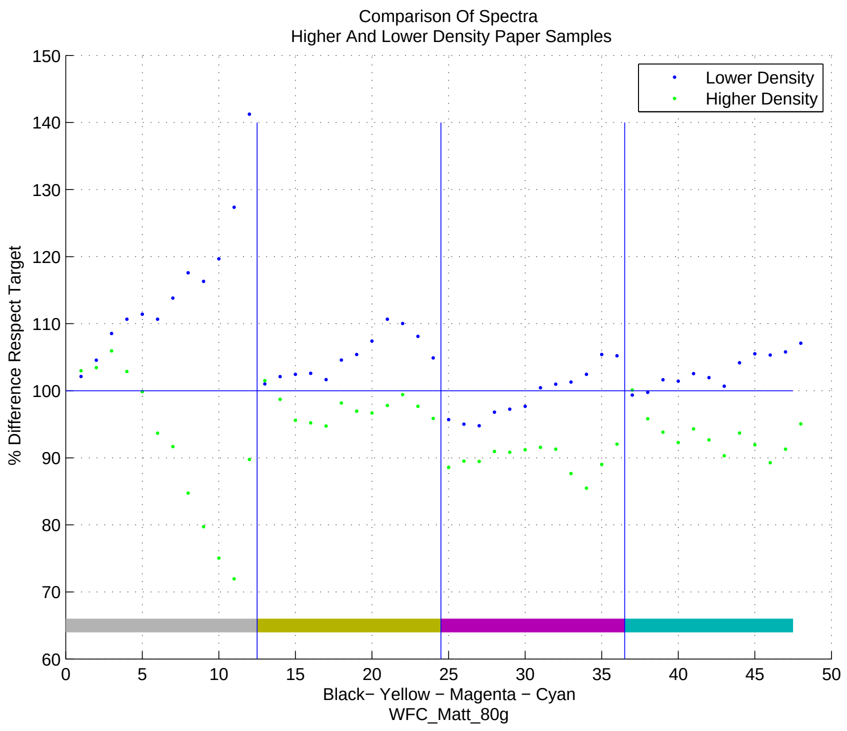
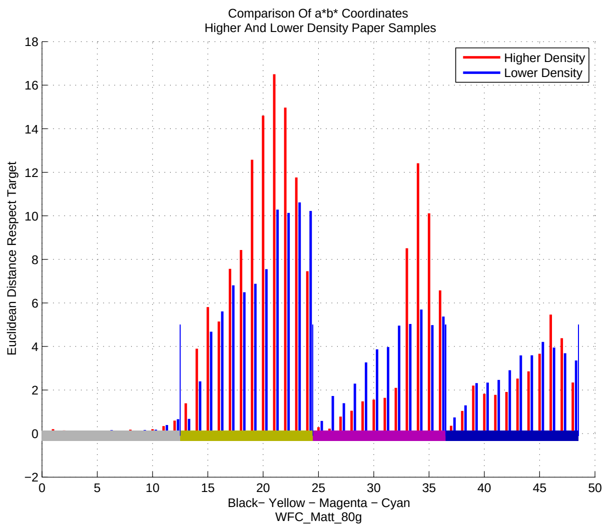


Figure A.3: Spectra Comparison - WFC Matt 80g



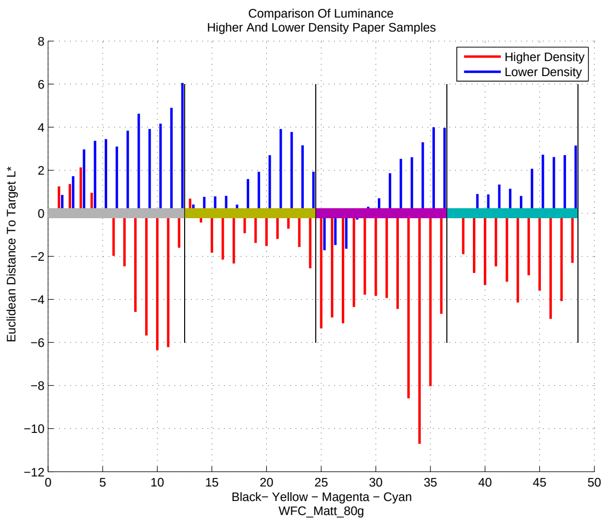


Figure A.4: L*a*b* Coordinates Comparison - WFC Matt 80g

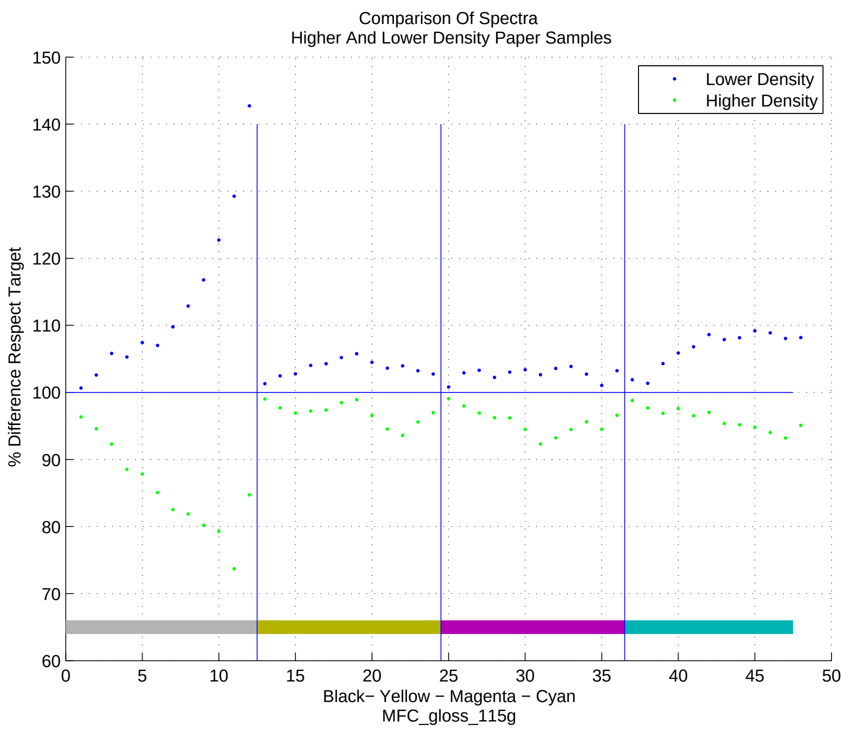
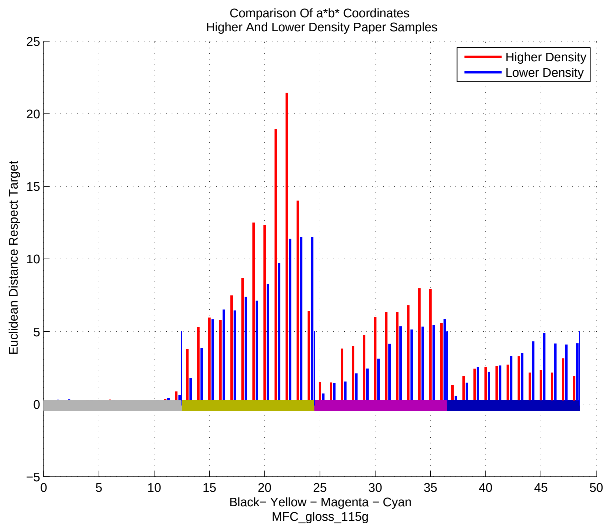


Figure A.5: Spectra Comparison - WFC Gloss 115g



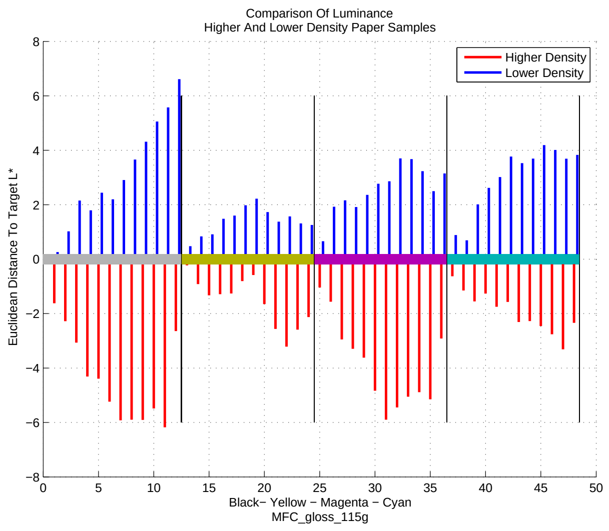


Figure A.6: L*a*b* Coordinates Comparison - WFC Gloss 115g

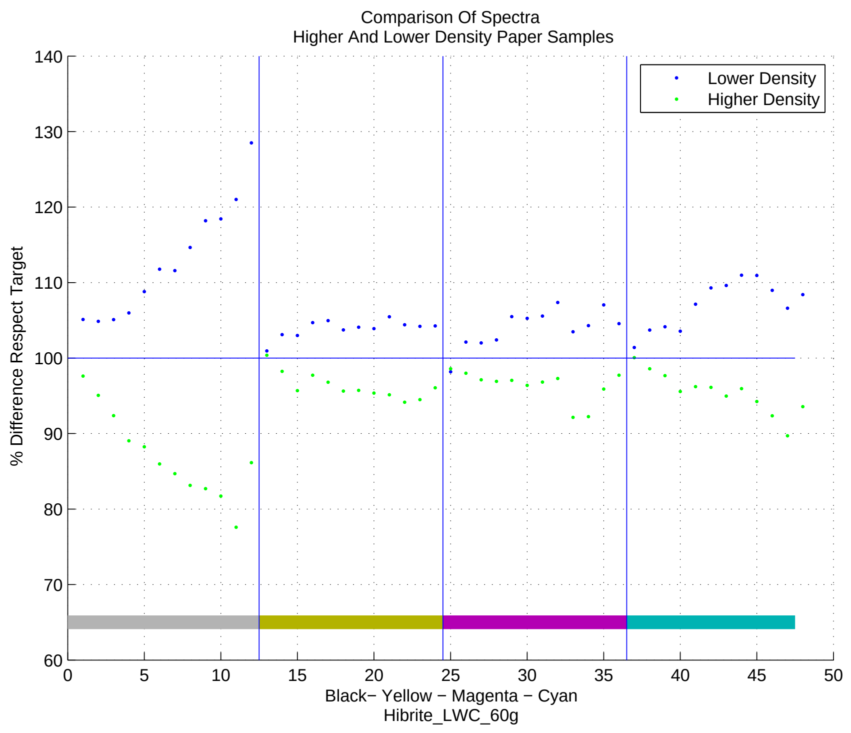


Figure A.7: Spectra Comparison - Hi-Brite LWC 60g

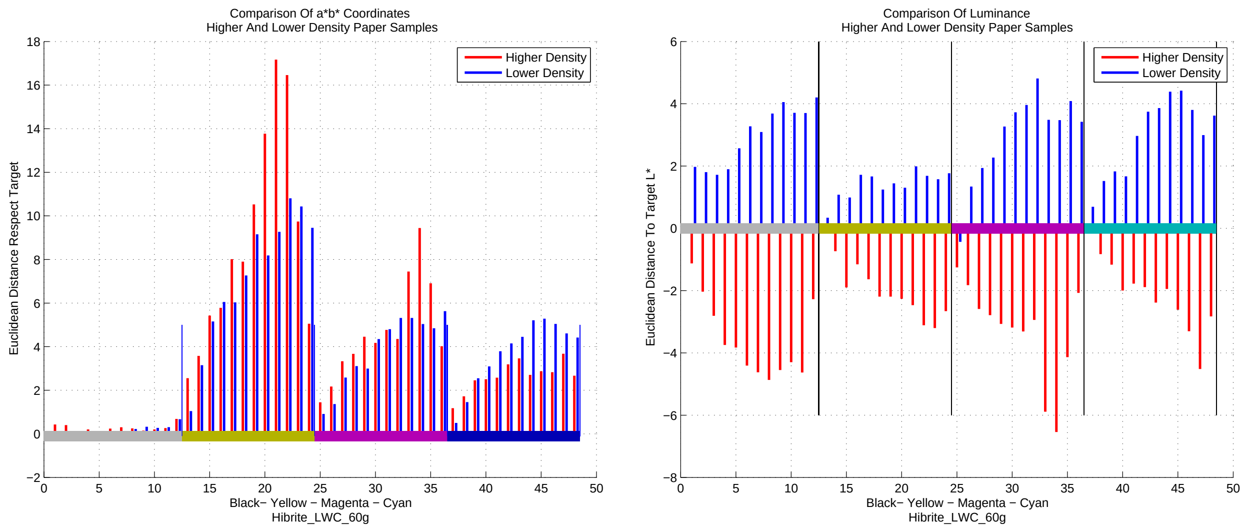


Figure A.8: L*a*b* Coordinates Comparison - Hi-Brite LWC 60g

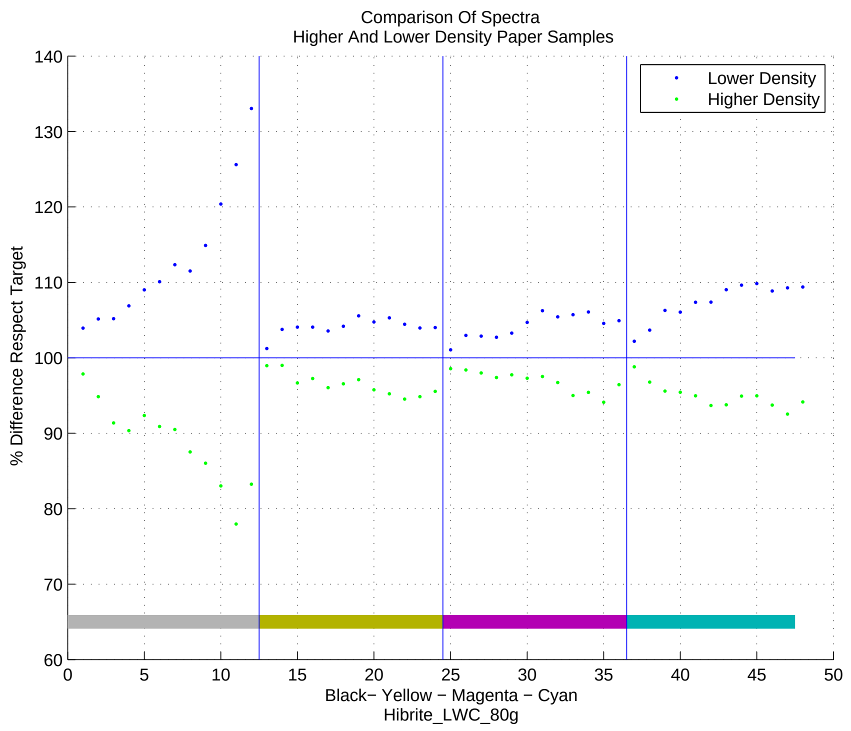
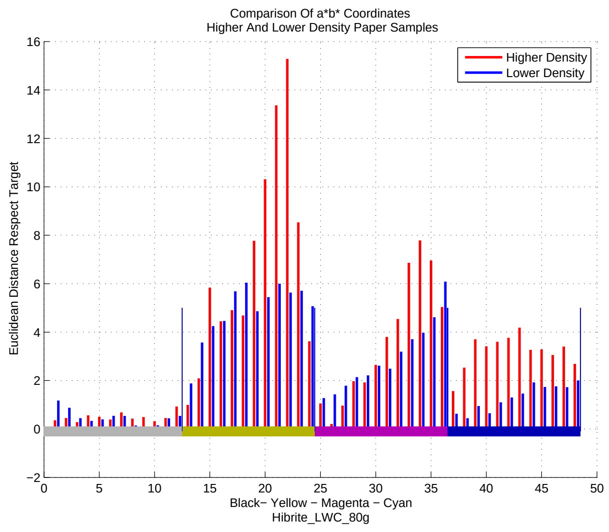


Figure A.9: Spectra Comparison - Hi-Brite LWC 80g



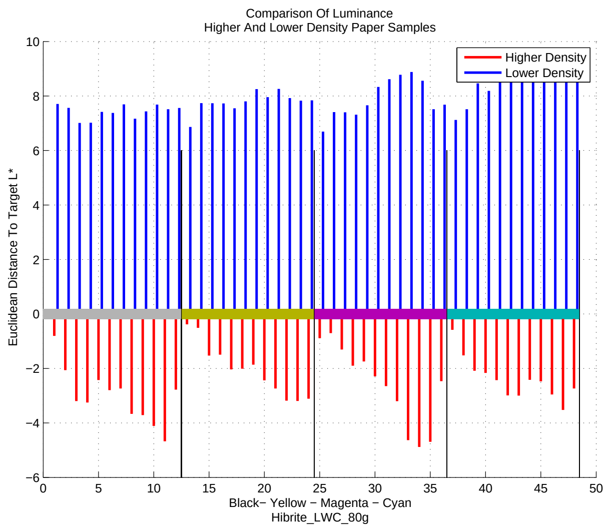


Figure A.10: L*a*b* Coordinates Comparison - Hi-Brite LWC 80g

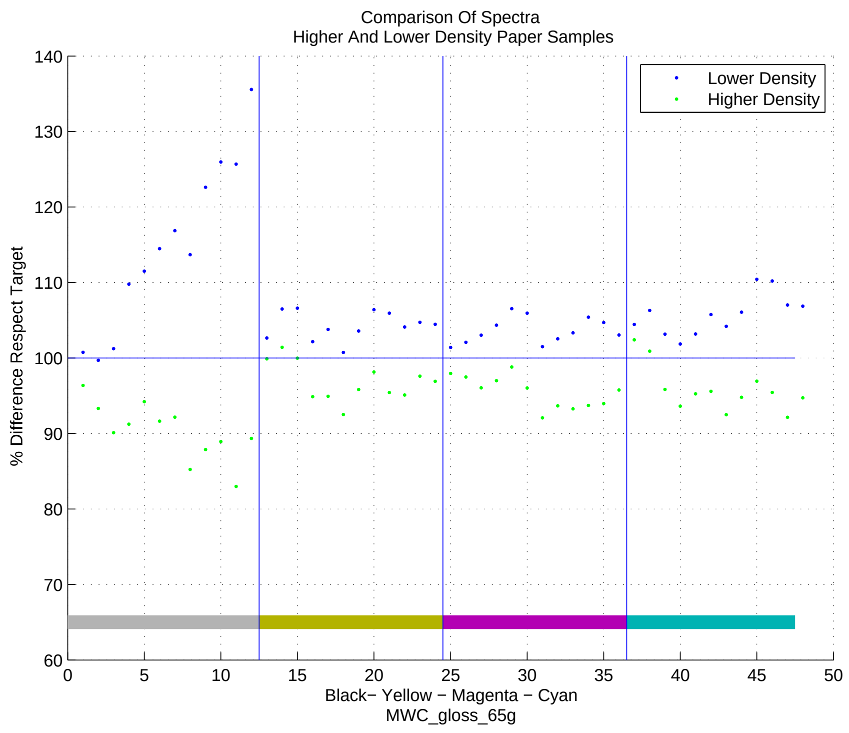


Figure A.11: Spectra Comparison - MWC Gloss 65g

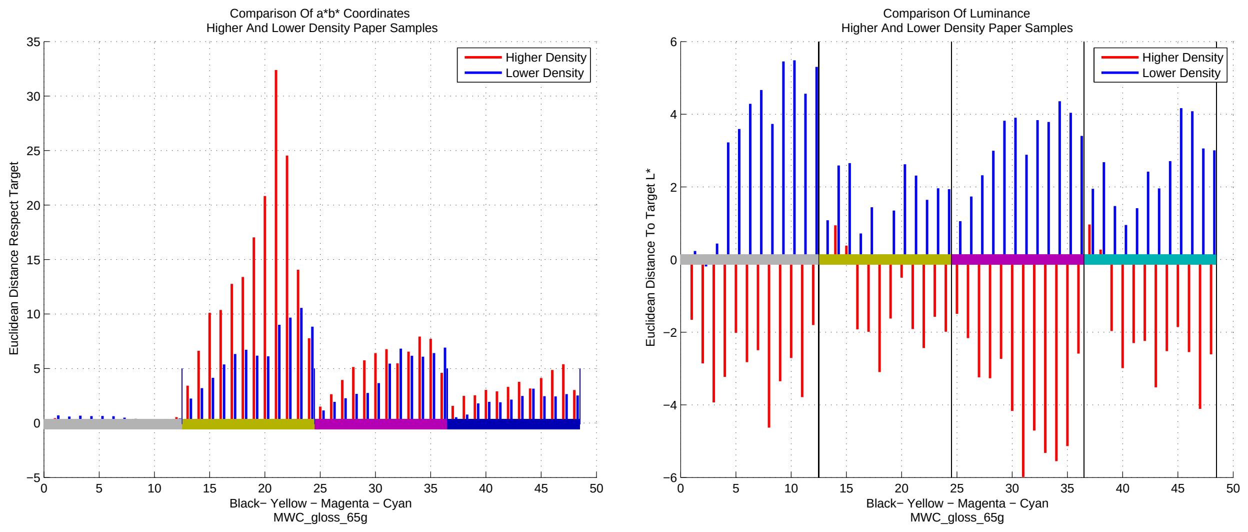


Figure A.12: L*a*b* Coordinates Comparison - MWC Gloss 65g

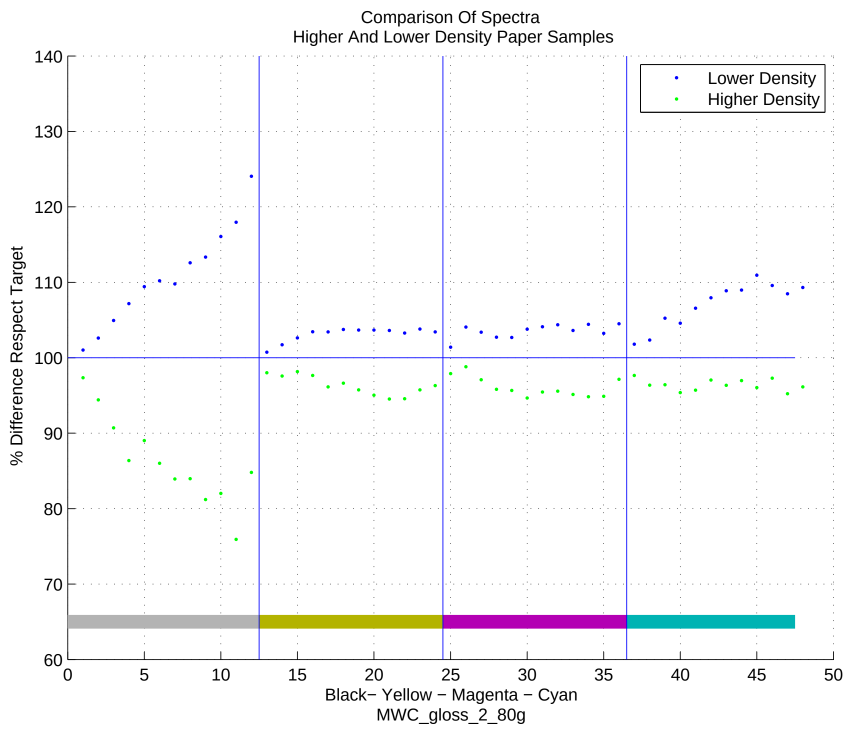


Figure A.13: Spectra Comparison - MWC Gloss 2 80g

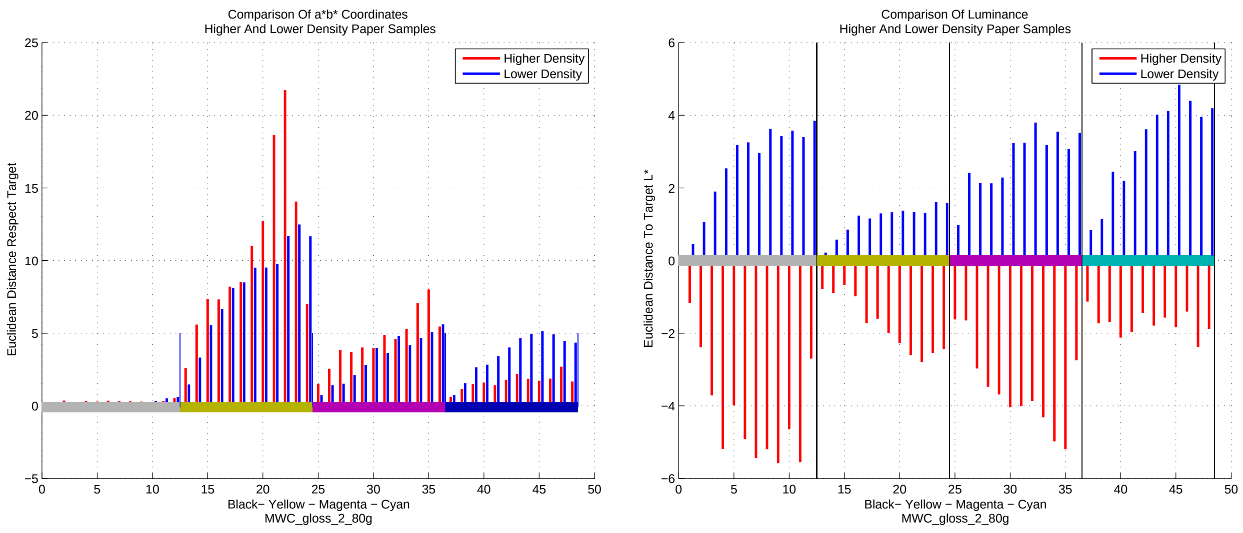


Figure A.14: L*a*b* Coordinates Comparison - MWC Gloss 2 80g

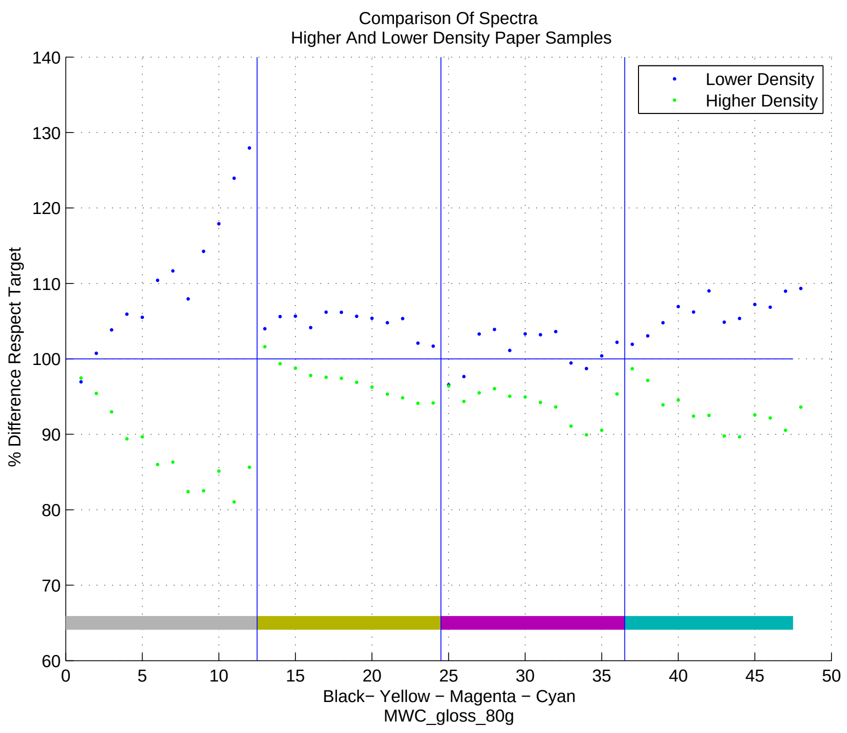


Figure A.15: Spectra Comparison - MWC Gloss 80g

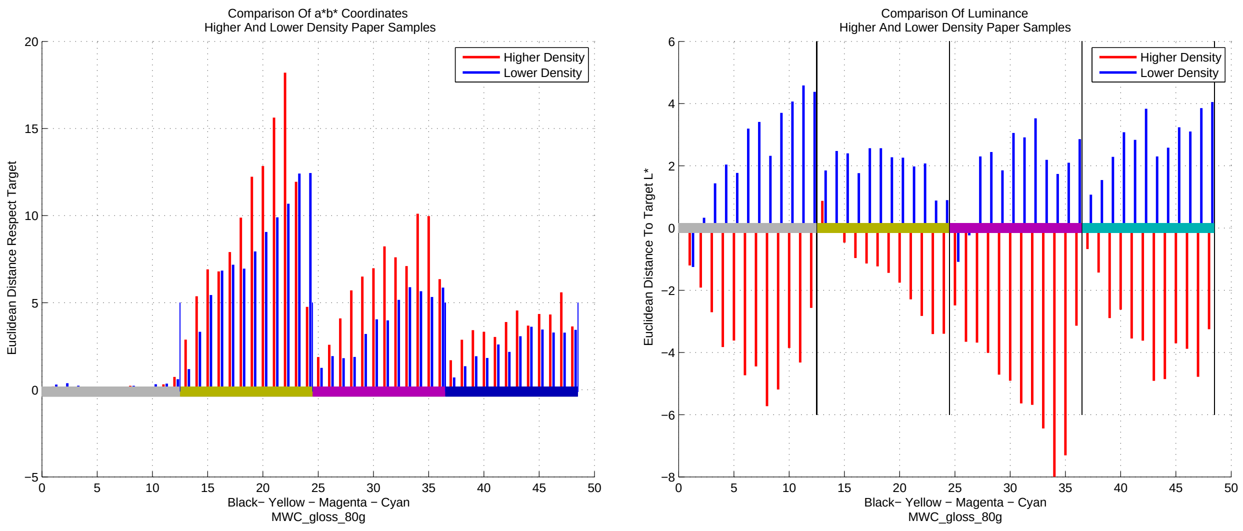


Figure A.16: L*a*b* Coordinates Comparison - MWC Gloss 80g

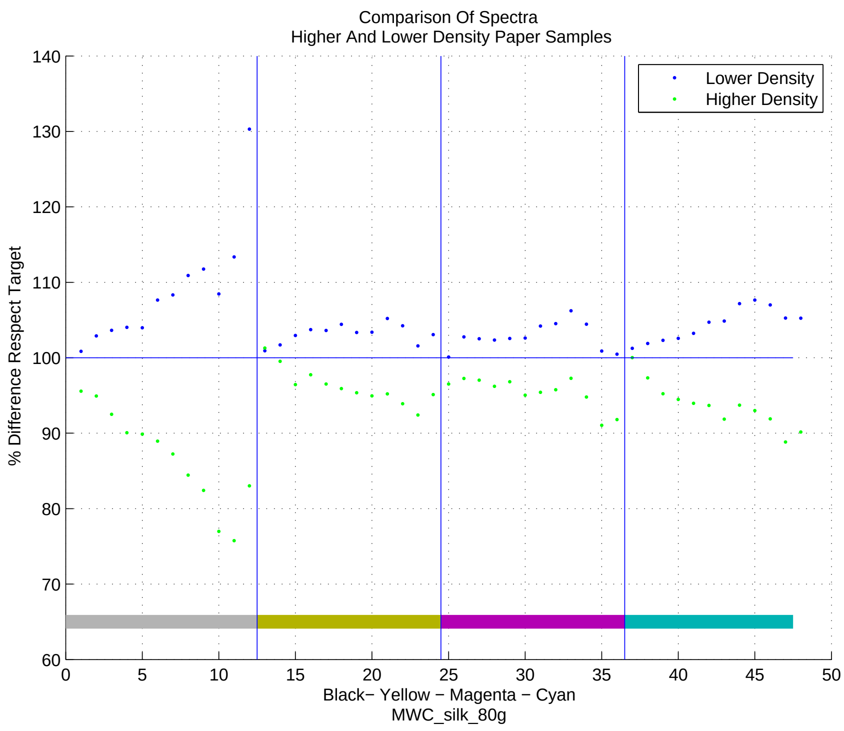


Figure A.17: Spectra Comparison - MWC Silk 80g

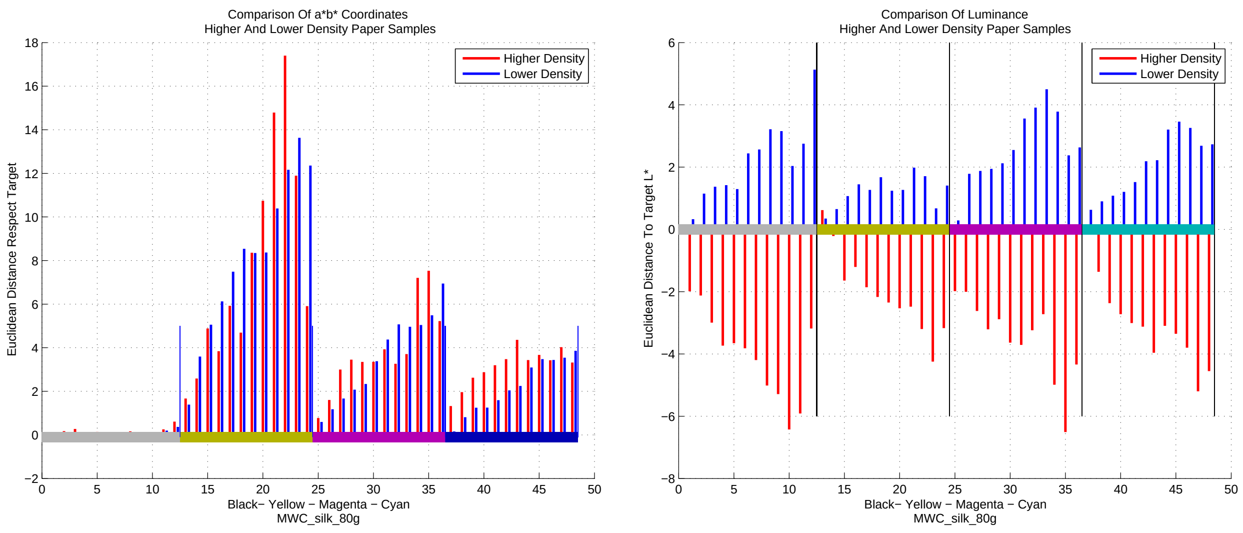


Figure A.18: L*a*b* Coordinates Comparison - MWC Silk 80g

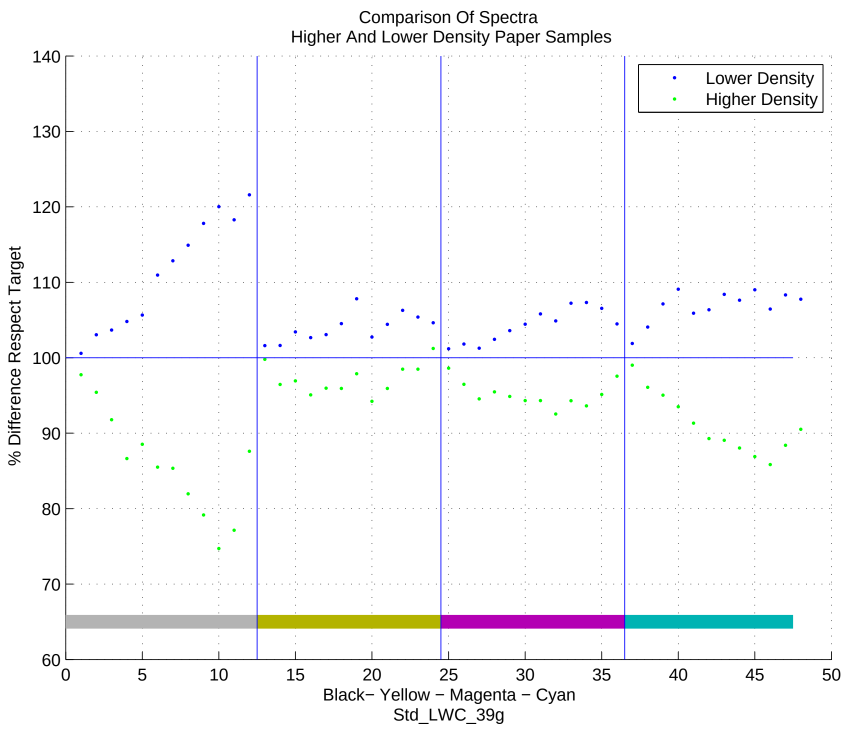


Figure A.19: Spectra Comparison - Std LWC 39g

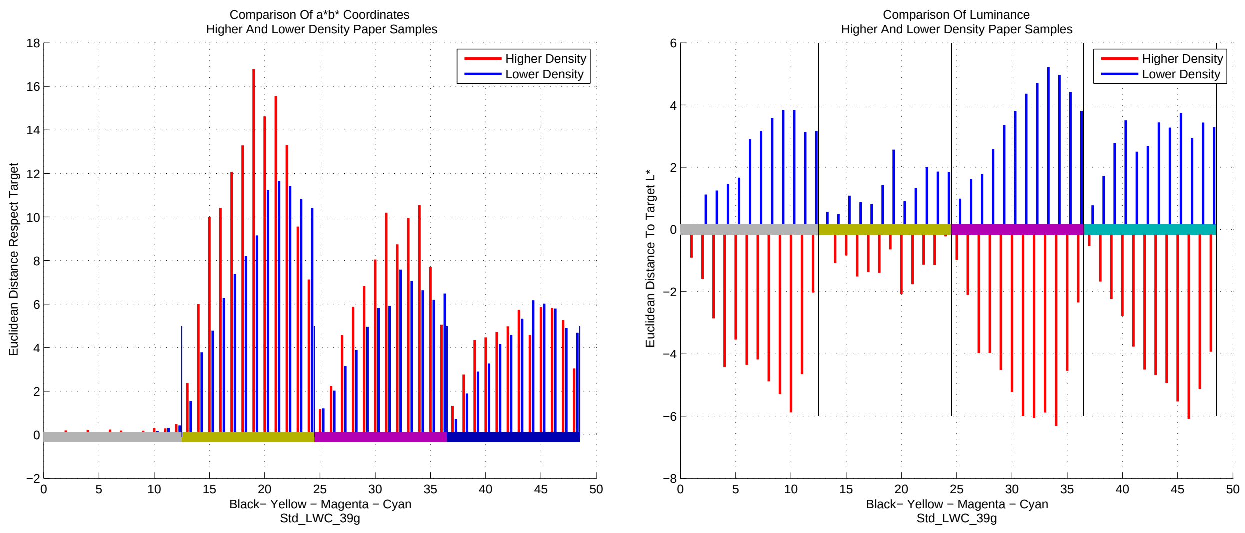


Figure A.20: L*a*b* Coordinates Comparison - Std LWC 39g

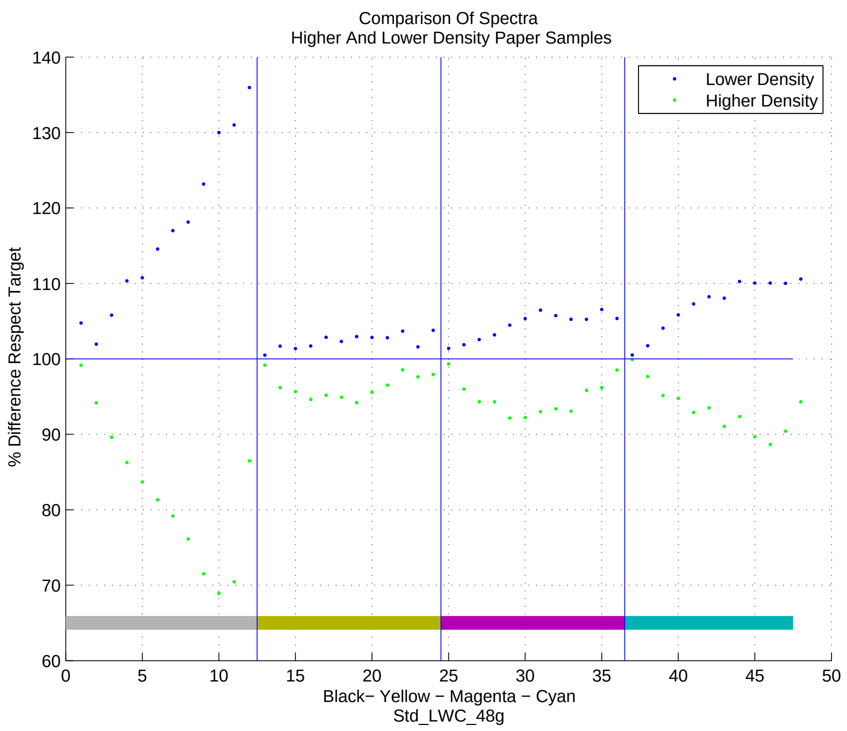
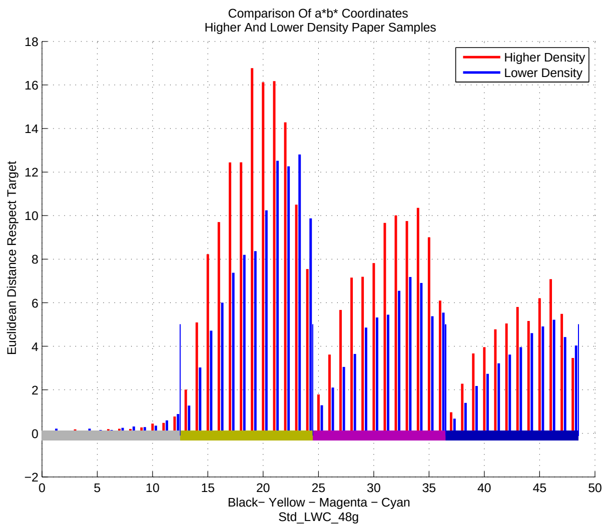


Figure A.21: Spectra Comparison - Std LWC 48g



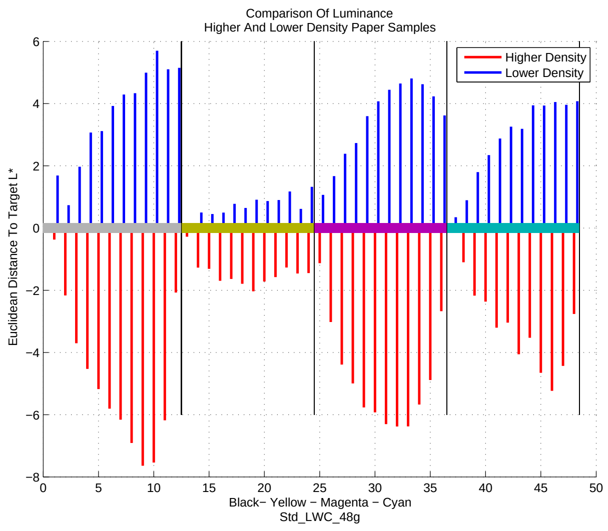


Figure A.22: L*a*b* Coordinates Comparison - Std LWC 48g

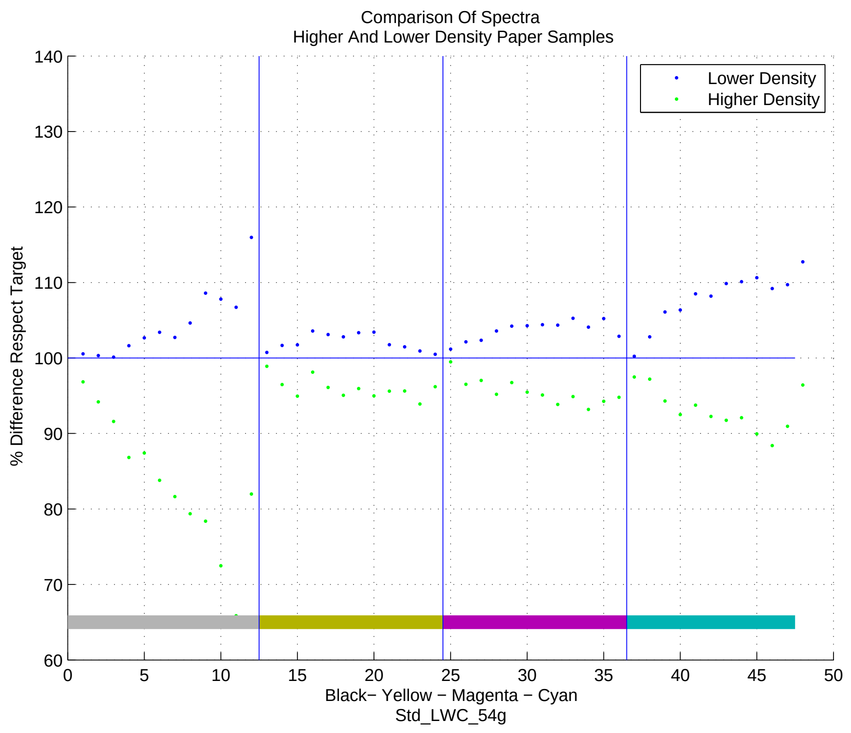


Figure A.23: Spectra Comparison - Std LWC 54g

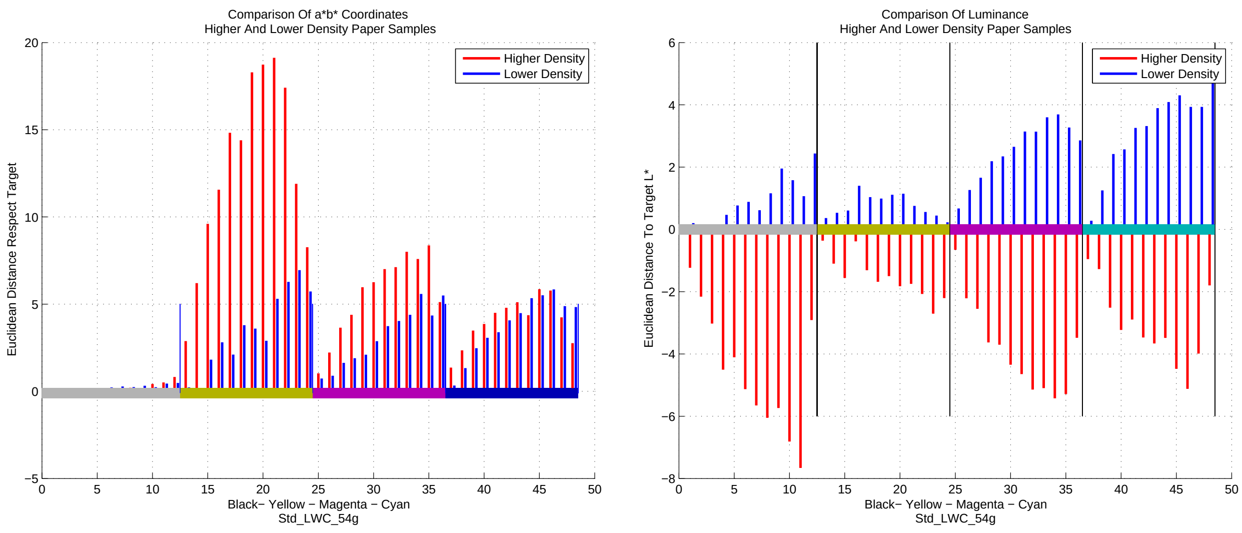


Figure A.24: L*a*b* Coordinates Comparison - Std LWC 54g

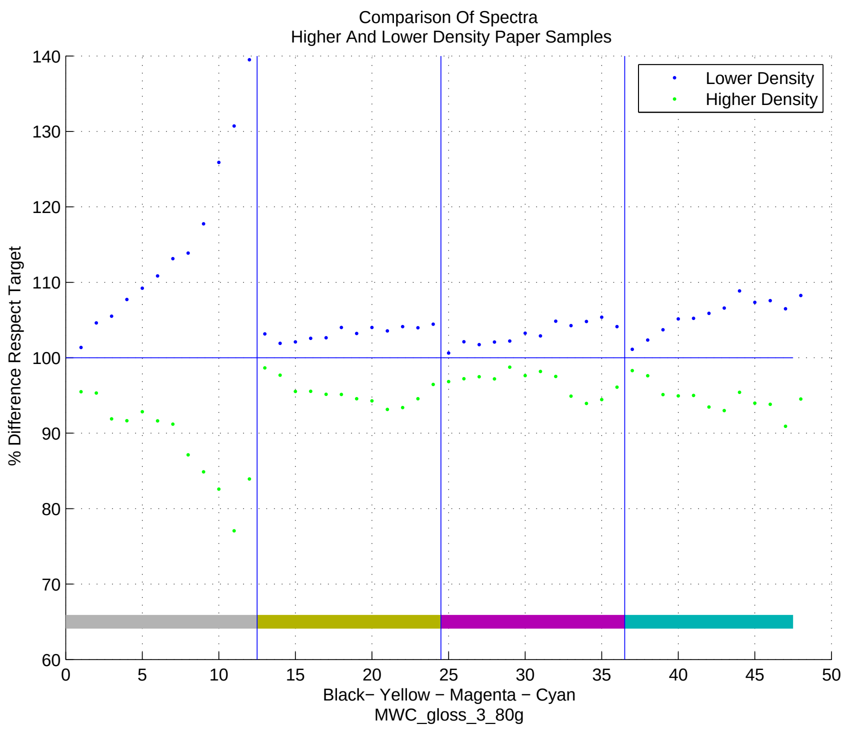


Figure A.25: Spectra Comparison - MWC Gloss 115g

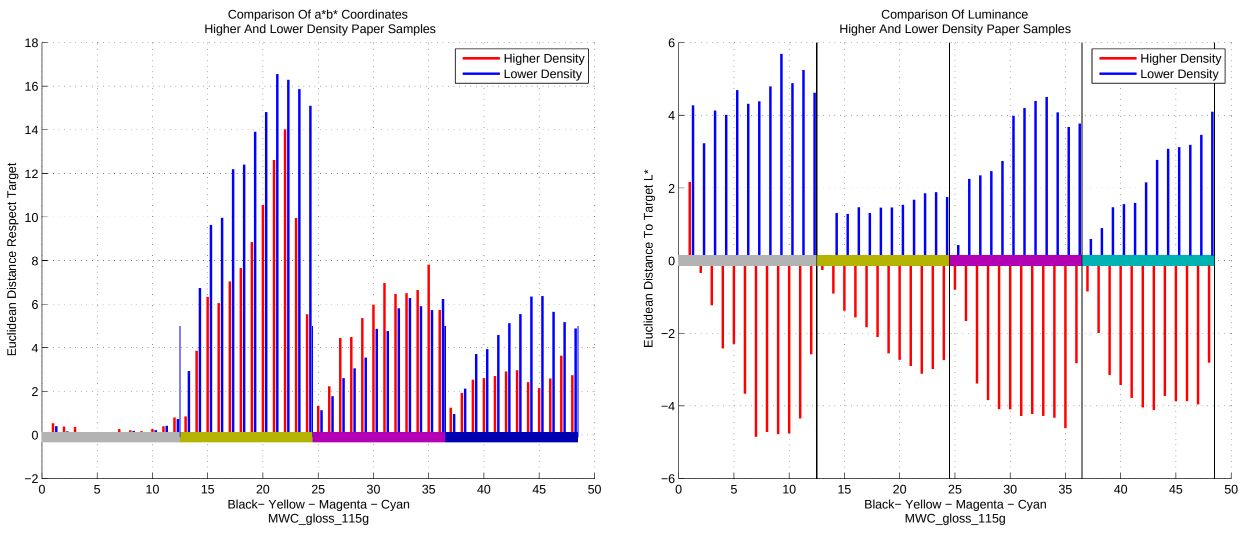


Figure A.26: L*a*b* Coordinates Comparison - MWC Gloss 115g

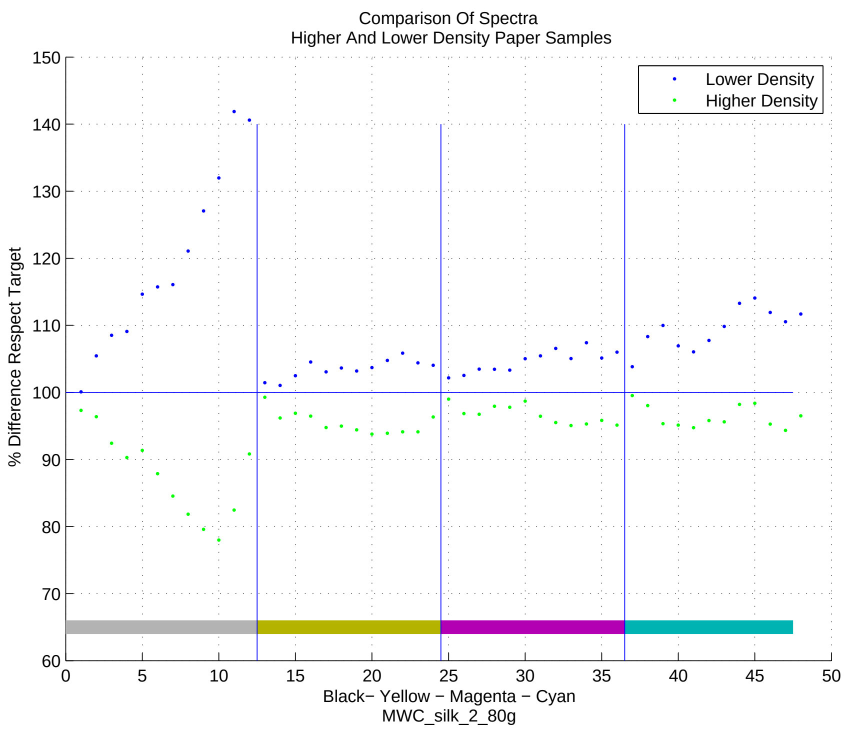


Figure A.27: Spectra Comparison - MWC Silk 2 80g

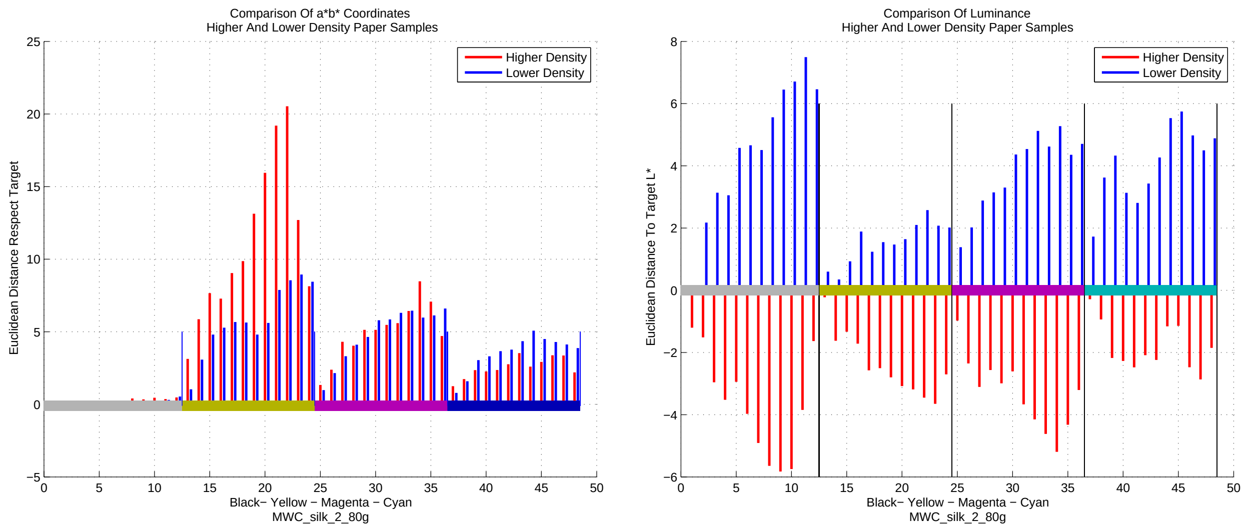


Figure A.28: L*a*b* Coordinates Comparison - MWC Silk 2 80g

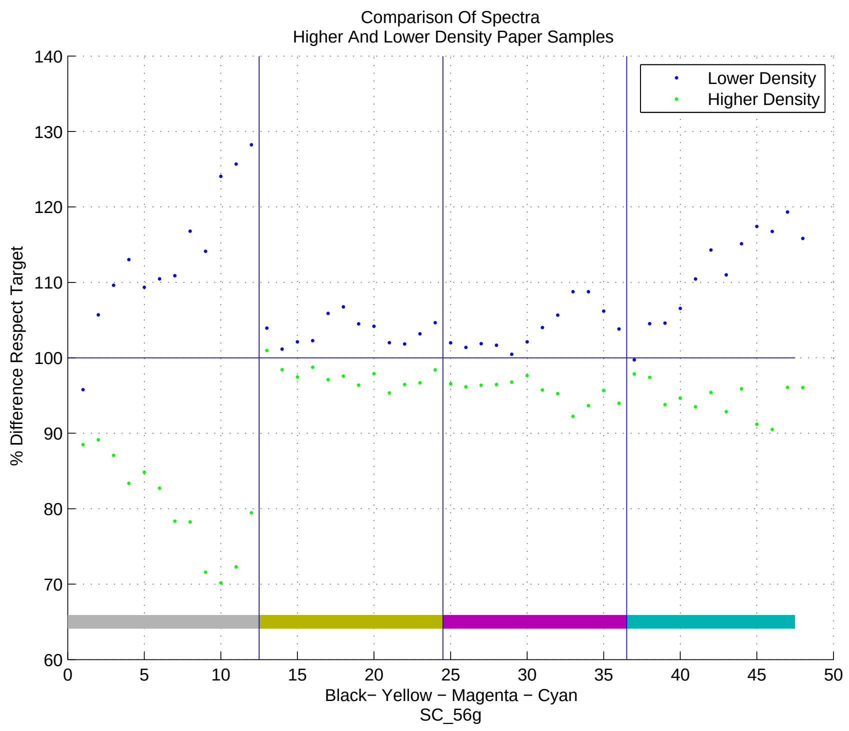


Figure A.29: Spectra Comparison - SC 56g

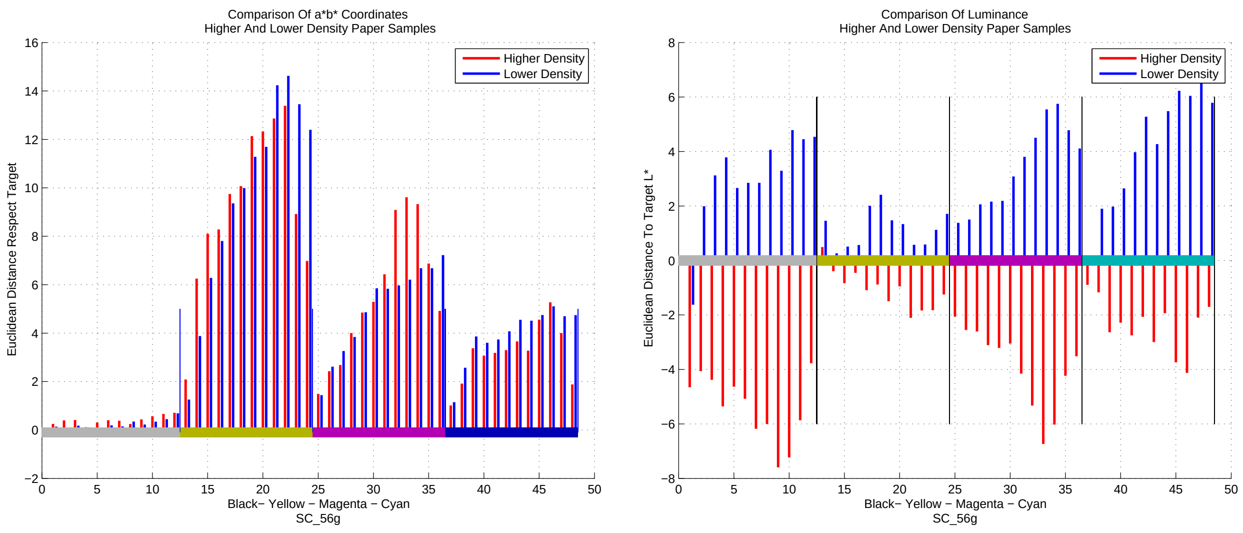


Figure A.30: L*a*b* Coordinates Comparison - SC 56g

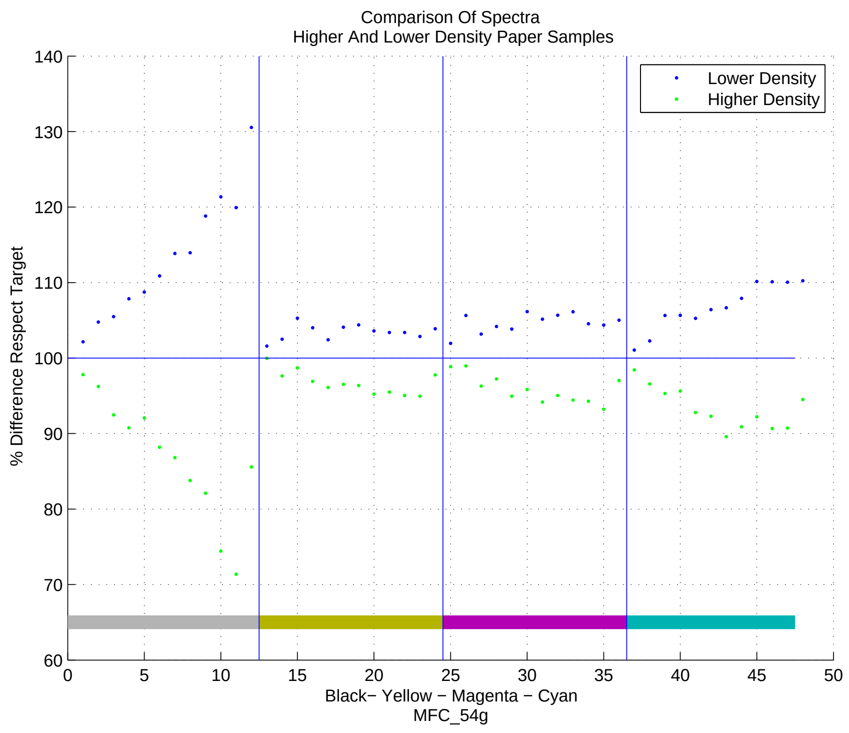


Figure A.31: Spectra Comparison - MFC 54g

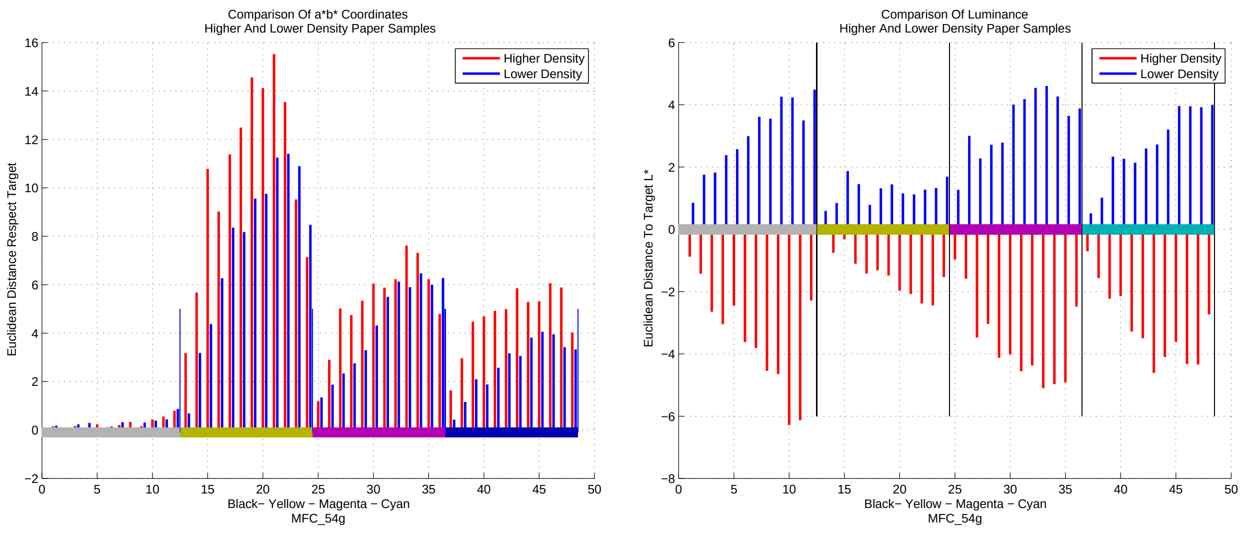


Figure A.32: L*a*b* Coordinates Comparison - MFC 54g

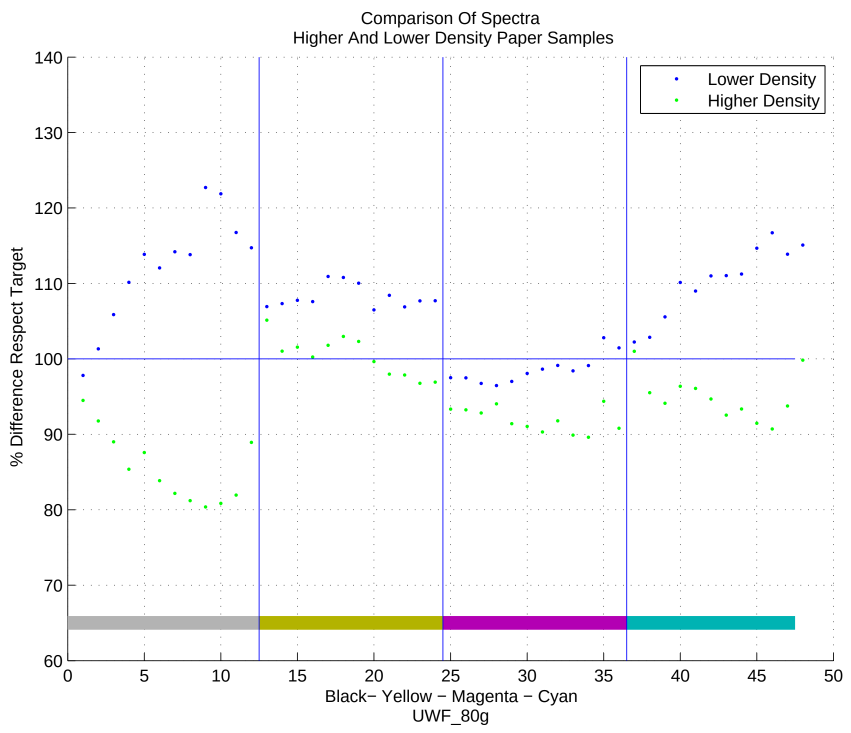
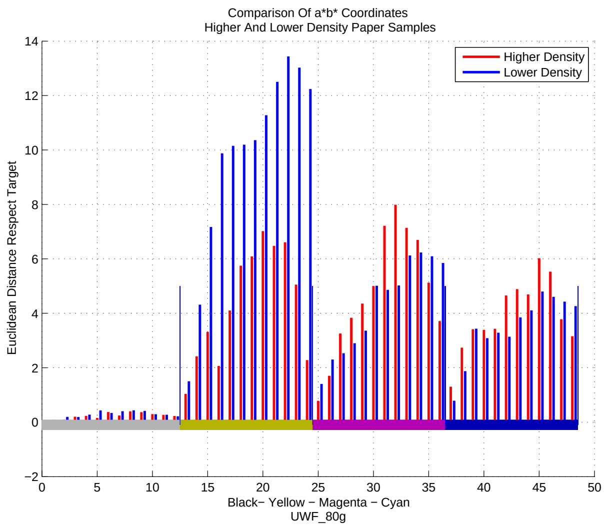


Figure A.33: Spectra Comparison - UWF 80g



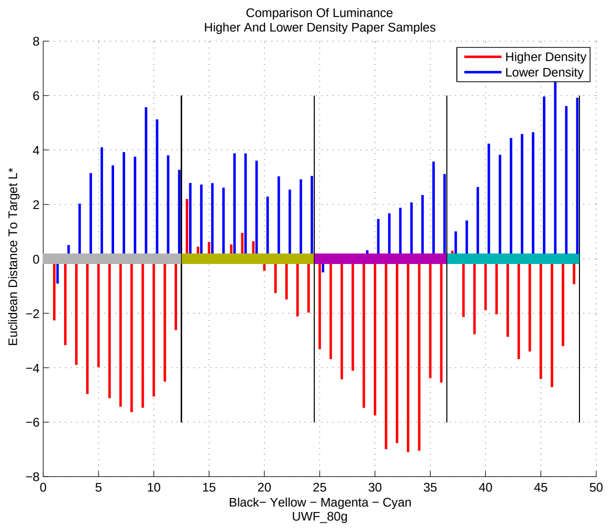


Figure A.34: L*a*b* Coordinates Comparison - UWF 80g

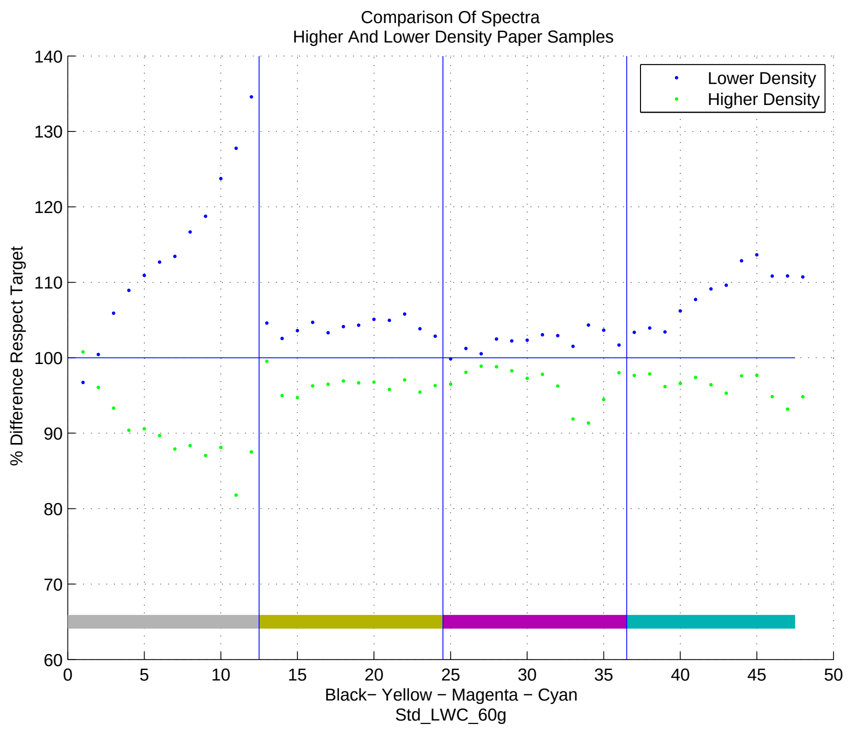
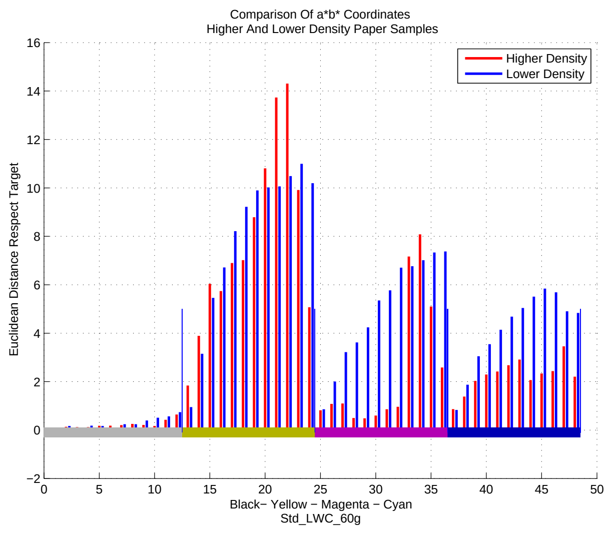


Figure A.35: Spectra Comparison - Std LWC 60g



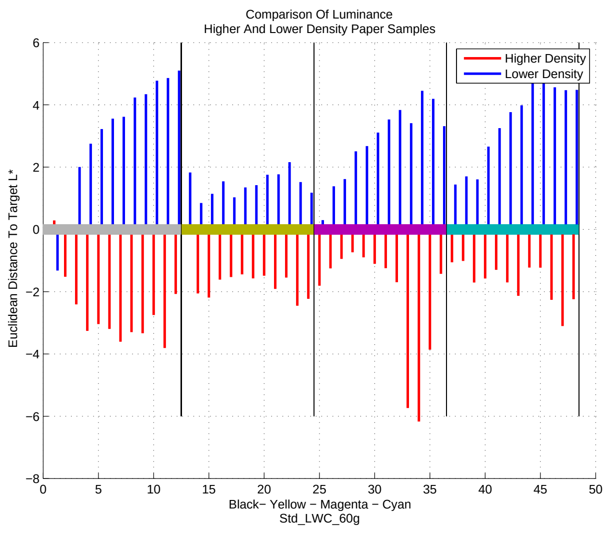


Figure A.36: L*a*b* Coordinates Comparison - Std LWC 60g

Appendix B

Spectral Image Illuminants Examples

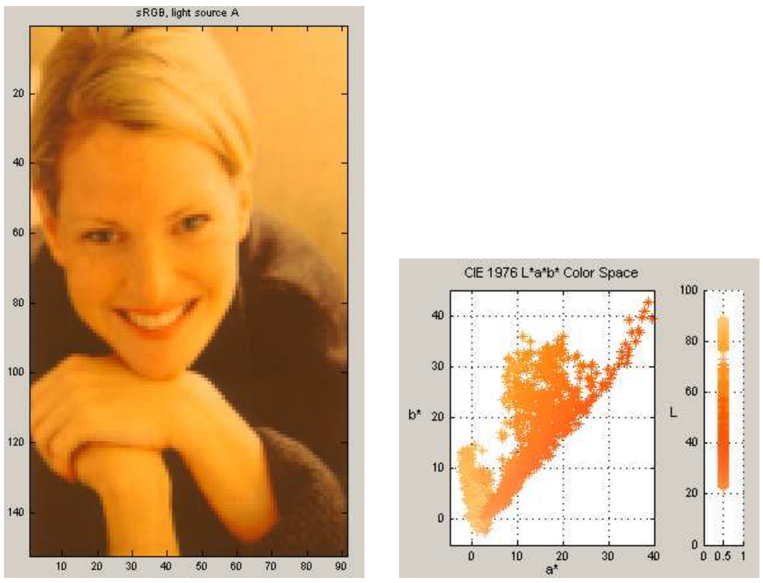


Figure B.1: 1931 Illuminant A Image - CIE Space

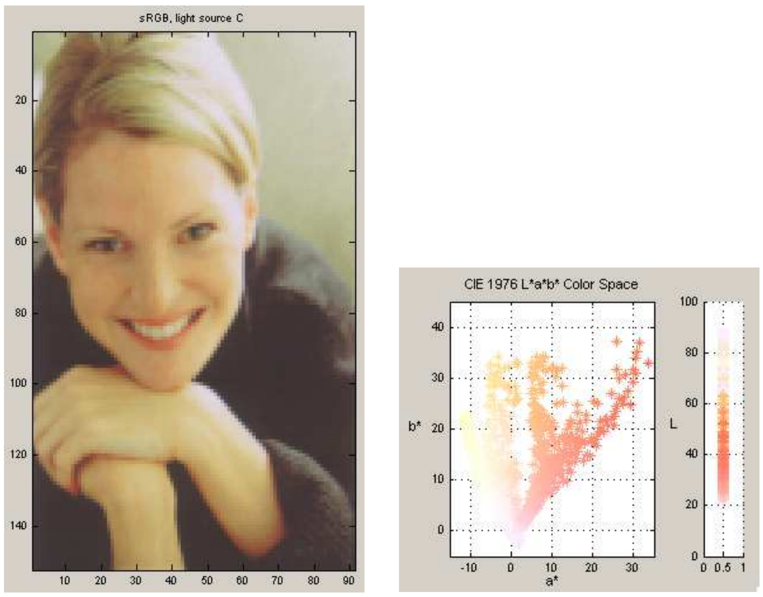


Figure B.2: 1931 Illuminant C Image - CIE Space

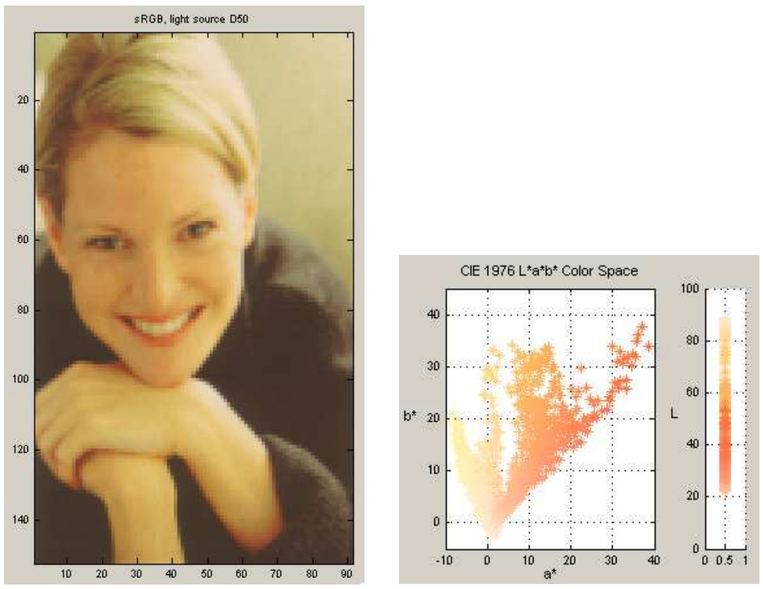


Figure B.3: 1931 Illuminant D50 Image - CIE Space

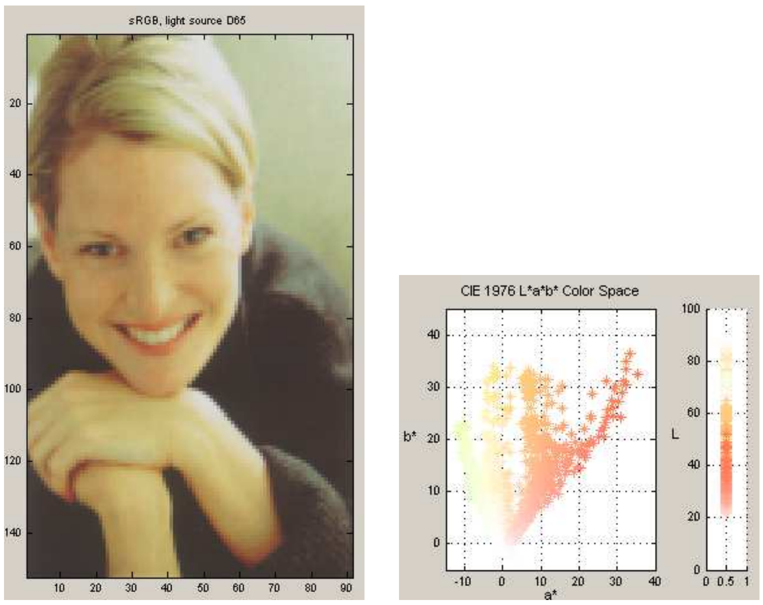


Figure B.4: 1931 Illuminant D65 Image - CIE Space

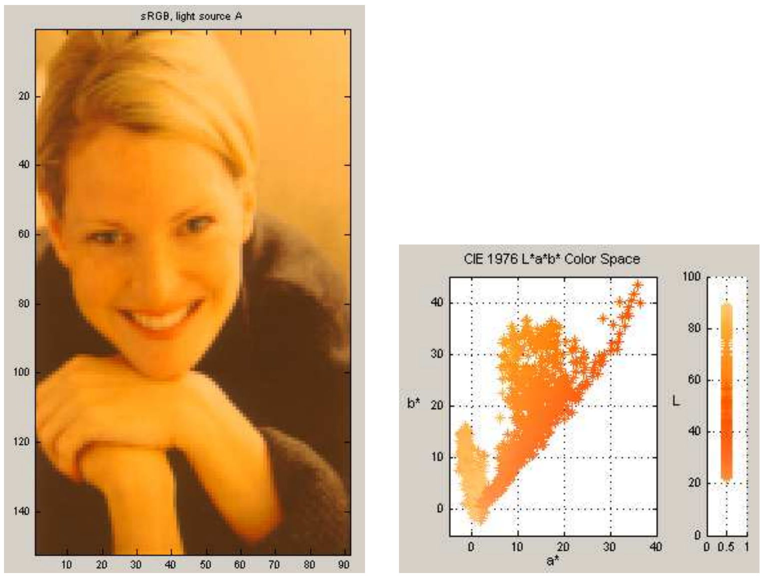


Figure B.5: 1964 Illuminant A Image - CIE Space

Figure B.6: 1964 Illuminant C Image - CIE Space

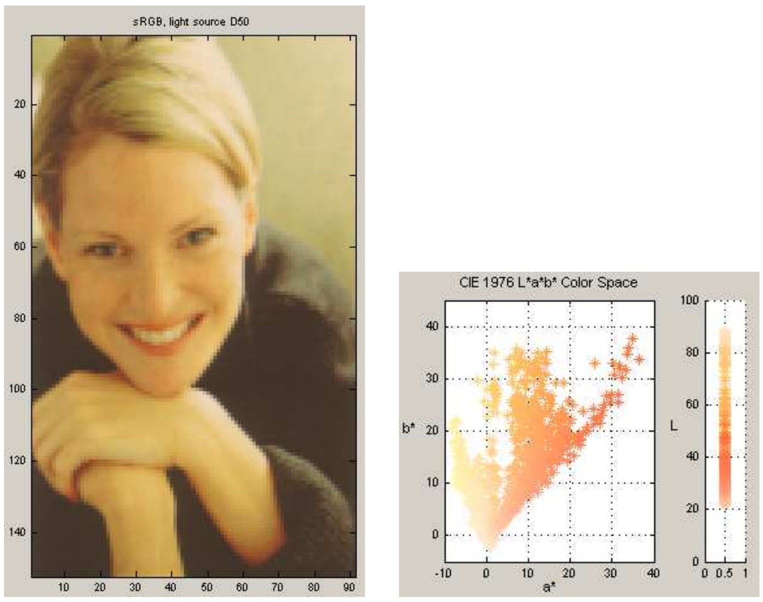
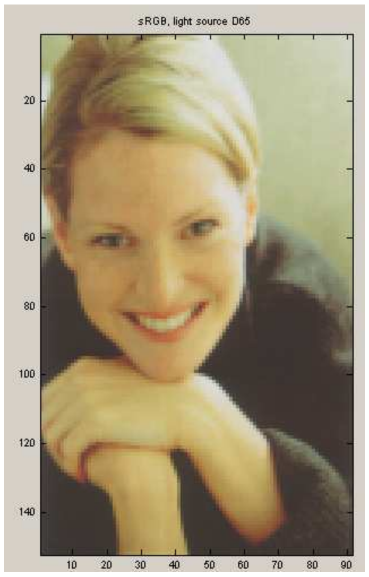


Figure B.7: 1964 Illuminant D50 Image - CIE Space



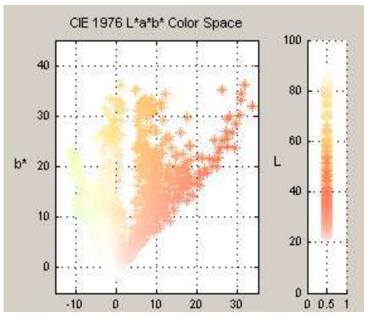


Figure B.8: 1964 Illuminant D65 Image - CIE Space

Bibliography

[AS07] Jess Angulo and Jean Serra. Modelling and segmentation of colour images in polar representations. *Image Vision Comput.*, 25(4):475–495, 2007.

[CEY84] J. L. Crawford, C. D. Elzinga, and R. Yudico. Print quality measurements for high-speed electrophotographic printers. *IBM J. Res. Dev.*, 28(3):276–284, 1984.

[Cha] Mark L. Chang. Adaptive computing in nasa multi-spectral image processing.

[GCKP01] J. Goy, B. Courtois, J. M. Karam, and F. Pressecq. Design of an aps cmos image sensor for low light level applications using standard cmos technology. *Analog Integr. Circuits Signal Process.*, 29(1-2):95–104, 2001.

[HRV97] B. Hill, Th. Roger, and F. W. Vorhagen. Comparative analysis of the quantization of color spaces on the basis of the cielab color-difference formula. *ACM Trans. Graph.*, 16(2):109–154, 1997.

[JG78] George H. Joblove and Donald Greenberg. Color spaces for computer graphics. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 20–25, New York, NY, USA, 1978. ACM Press.

[KM06] Mukesh Kumar and Douglas A. Miller. A non-parametric classification strategy for remotely sensed images using both spectral and textural information. In *SPPRA'06: Proceedings of the 24th IASTED international conference on Signal processing, pattern recognition, and applications*, pages 81–89, Anaheim, CA, USA, 2006. ACTA Press.

[KVV04] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, 2004.

[MG80] Gary W. Meyer and Donald P. Greenberg. Perceptual color spaces for computer graphics. In *SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 254–261, New York, NY, USA, 1980. ACM Press.

[MLAB06] Arnaud Martin, Hicham Laanaya, and Andreas Arnold-Bos. Evaluation for uncertain image classification and segmentation. *Pattern Recogn.*, 39(11):1987–1995, 2006.

[OO06] Alberto Ortiz and Gabriel Oliver. On the use of the overlapping area matrix for image segmentation evaluation: A survey and new performance measures. *Pattern Recogn. Lett.*, 27(16):1916–1926, 2006.

[otCIdL86] Central Bureau of the Commission Internationale de L'clairage. *Publication CIE No 15.2.* Colorimetry, Second Edition, Vienna, Austria, 1986.

[Pee93] Mark S. Peercy. Linear color representations for full speed spectral rendering. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 191–198, New York, NY, USA, 1993. ACM Press.

[Reg76] Satish L. Rege. Performance evaluation of ccd chip organizations from memory system design viewpoint. In *ACM 76: Proceedings of the annual conference*, pages 30–37, New York, NY, USA, 1976. ACM Press.

[SCB88] Maureen C. Stone, William B. Cowan, and John C. Beatty. Color gamut mapping and the printing of digital color images. *ACM Trans. Graph.*, 7(4):249–292, 1988.

[ST03] Konstantinos Koutroumbas Sergios Theodoridis. *Pattern Recognition, Second Edition.* Academic Press, 2003.

[TT99] Shoji Tominaga and Etsushi Takahashi. Spectral image processing by a multi-channel camera. In *ICIP (3)*, pages 575–579, 1999.

[WEV02] Greg Ward and Elena Eydelberg-Vileshin. Picture perfect rgb rendering using spectral prefiltering and sharp color primaries. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 117–124, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[WHD03] A. Wenger, T. Hawkins, and P. Debevec. Optimizing color matching in a lighting reproduction system for complex subject and illuminant spectra. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 249–259, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[Wik06] Wikipedia. online free encyclopedia. 2006.

[WS82] Gnter Wyszecki and W.S. Styles. *Color Science: Concepts and Methods, Quantitative Data and Formulae.* Second Edition (John Wiley and Sons, New York), 1982.

[ZCH$^+$04] Xin Zheng, Deng Cai, Xiaofei He, Wei-Ying Ma, and Xueyin Lin. Locality preserving clustering for image database. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 885–891, New York, NY, USA, 2004. ACM Press.