Color Gamut Compression for Computer Printing

by

Brian E. Barth

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science

and

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1989

Signature of Author .....
Department of Electrical Engineering and Computer Science
May 12, 1989

Certified by .....
William F. Schreiber
Professor of Electrical Engineering
Thesis Supervisor

Certified by .....
John D. Meyer
Company Supervisor, Hewlett-Packard

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Color Gamut Compression
for Computer Printing

by

Brian E. Barth

Submitted to the
Department of Electrical Engineering and Computer Science
on May 12, 1989 in partial fulfillment of the requirements
for the Degree of Master of Science

Abstract

Exact colorimetric reproduction of a color image is often impossible due to the color gamut limit of the output device. When printing digital color images, pixel values may specify colors that are unprintable on a given printer with the given inks. In order to avoid gross printing errors, all pixel values must be transformed into the range of printable colors. When colors are changed, the goal is to minimize the impact on the overall appearance of the image. It is very difficult to quantify image appearance accurately.

In order to gamut-compress an image, first the lightness range of the image is reduced using spatial filtering in the CIE psychometric lightness domain to attenuate the low frequency lightness component of the image. This allows the contrast of the high frequencies to be maintained although the dynamic range is reduced. Second, adaptive three-dimensional gamut compression is performed in which image chroma is smoothly compressed as a function of lightness, hue angle, and chroma. Adaptive gamut compression warps the shape of the image's three-dimensional color gamut solid to better fit the device's color gamut solid. By comparison, the previous technique of global compression

shrinks the image gamut until it fits entirely within the device gamut, thus usually wasting part of the device's gamut capability. Another previous technique, pixel clipping, chops off the image gamut wherever it exceeds the device gamut. Adaptive scaling eliminates the loss of detail from pixel clipping yet retains the colorfulness of the image by using the entire device gamut.

Thesis Supervisor: Prof. William F. Schreiber

Title: Professor of Electrical Engineering

Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to thank my manager at Hewlett Packard Laboratories, John Meyer, for recommending this research topic, for being a ready source of expert advice in the field of graphic arts, and for his experienced judging of the many processed images. I would also like to thank Ricardo Motta for providing expert advice in the field of color and for proofreading parts of this thesis.

Thanks to Gary Dispoto for programs such as the clustered-dot dither used to print the gray scale images for this document. Thanks to Andy Fitzhugh for knowing every little detail about UNIX and C, and to Rich Baer for his efforts to get a second printer working.

I would also like to thank Bill Lloyd, my department manager, for making this thesis possible, and Professor William Schreiber for being my thesis supervisor.

Chapter 1

Introduction

1.1 Background

1.1.1 Problem

Each imaging device, medium, and storage format has its own associated range of representable colors. In general, this range is quite different for a printer, video monitor, scanner, and transparency projection. There often are significant differences between the capabilities of different devices within the same functional category. The range of colors a particular device or medium handles is called its *color gamut.* Calibrated colorimetric measurements are necessary to have meaningful comparisons between various color gamuts. The difference between color gamuts can create a problem whenever an image is rendered or translated between devices. If the destination gamut is more restricted than the source gamut in some region, then the colors in that region must somehow be compressed.

This project deals with the color gamut compression problem in the specific context of printing digital color images from a computer. The purpose of this project is to transform, in a controlled way, all non-printable colors in an image into printable colors. In the process of changing out-of-gamut colors, it may be desirable to also change in-

gamut colors slightly. If the color gamut of a printer is viewed as a continuous solid in some three-dimensional color space, then each pixel value must lie inside the solid for the image to be printable. When transforming an image to lie within a particular device's color gamut, the ultimate goal is to preserve the perceptual appearance of the original. This objective is difficult to quantify, and consequently methods for gamut compression tend to be rather ad hoc. Some approaches to the gamut compression problem involve defining fixed device-to-device mappings; the mappings explored here depend on both the particular image and rendering device. For this project, the desired appearance of the image is assumed to be specified correctly. These algorithms do not explicitly attempt to correct or enhance poor quality image data.

1.1.2 The Goal: Appearance Matching

The goal in gamut compression is to preserve the appearance of the original image while using fewer colors. Ideally, we could calculate the optimal output of the gamut mapping procedure by minimizing the change in appearance between the original and gamut-constrained image while imposing the color gamut constraint. This approach is not practical because image appearance is difficult, if not impossible, to quantify precisely. Some methods have achieved good results for small amounts of compression by trying to match appearance by minimizing the change in each pixel value as measured in a uniform color space. This approach does not take into account the many perceptual phenomena that affect appearance. Perceptual phenomena are as important to image appearance as is the spectral power distribution that reaches the eye. Light energy is easier to measure than appearance, but trying to compare images in terms of light energy is not really comparing appearance.

One example of a perceptual phenomenon that makes appearance matching difficult is *brightness constancy* [28] [9]. Brightness constancy explains the fact that the brightness of an object is not proportional to the level of illumination falling on it. It is as if our visual system attempted to judge the reflectance of objects, rather than simply measure

the amount of light energy coming from them. Because of brightness constancy, it makes sense to normalize the measurement of an image with respect to a white object in the scene. We judge the lightness of objects by comparing them to a white point in the scene, not by the absolute amount of light energy coming from the object. In Section 2.6, the white point is used to explicitly normalize the color space used for gamut compression.

A second example of a perceptual phenomenon that makes appearance matching difficult is *color constancy* (also called *hue constancy*) [9] [5]. Color constancy is the effect by which an object illuminated by two spectrally different light sources, like daylight and tungsten, appears to be the same color. According to colorimetry, discussed in Chapter 2, the tristimulus values of an object are related to the product of the spectral reflectance of the object and the spectral power distribution of the source. In general, this means that the tristimulus values of an object will change when the light source changes. Although a measurement with a colorimeter under two different light sources can give very different results, humans are relatively insensitive to such changes and usually judge the colors as having the same appearance. The lack of models describing this mechanism makes the measurement of appearance difficult.

One further complication to the simple minimization of appearance error is that the preferred reproduction of many familiar objects is different from their true colorimetric values [14]. In other words, some color "errors" could be be perceived as enhancements to the image. For example, the preferred colors of sky blue and caucasian skin are different from their colorimetric values. This makes the calculation of appearance error even more difficult because it is necessary to know what an object is in order to know how a color change will be judged. Furthermore, for some special objects it is important to have the colors look just right, while for other objects that are less familiar, accuracy does not matter as much. For example, a hue shift in the reproduction color of a person's face is usually much more noticeable than the same hue shift in the reproduction color of a telephone. The many factors that influence image appearance are difficult, if not impossible, to model accurately.

1.2 Some Previous Approaches to the Color Gamut Problem

Color gamut compression is not a new problem. Virtually all color imaging methods have an associated technique for gamut compression, be it explicit or implicit. It is useful to review some of the previous approaches, along with their benefits and drawbacks, before discussing the proposed improvements.

1.2.1 Graphic Arts

Graphic artists and photographers have a long history of experience with color gamut compression. Consequently, the graphics arts industry is a valuable source of ideas about color gamut compression. Understanding the graphic artist's methods for gamut compression is complicated by the fact that it is often performed in conjunction with other steps of the reproduction process, such as color correction and tone scale modification. Graphic artists commonly use one of two methods to accomplish color gamut compression: photographic processing and color scanners. Scanners offer the most control and flexibility. The scanner operator can adjust controls to specify the parameters of functions such as the tone reproduction curve, unsharp masking, and color balance of the image. In photography, masks and separations are made of the original image using special films and exposure techniques to produce a well-balanced, high-contrast reproduction. Both of these methods require a significant amount of human interaction and judgement. In compressing the gamut, the graphic artist makes a decision about which colors are important and how these key colors should be reproduced. These judgements are often made without immediate feedback; thus, extensive experience may be required to become proficient with these methods.

1.2.2 Xerox PARC

A project explicitly involving gamut mapping was undertaken at XEROX PARC (Palo Alto Research Center) in 1986 [29]. It involved preparing color separations for offset printing of a variety of computer images for a special issue of *Color Research and Application* [30]. The system consists of a calibrated input process for converting images into CIE tristimulus values, X, Y, Z; a calibrated output process for accurately printing reproducible colors from X, Y, Z values; and a set of gamut mapping tools for linking the two. The gamut mapping tools operate in the X, Y, Z domain using various mappings to convert calibrated tristimulus values from the monitor to calibrated tristimulus values for the printer, and in the process, eliminate all non-printable colors.

The Xerox processing scheme consists of three sequential transformations. Each transformation is performed interactively by watching projections of the image and device gamuts. The first transformation, involving translation, scaling, and gray axis rotation is the main part of the gamut mapping. Its primary purpose is to adjust the achromatic image component so that the gray axes of the image and destination device are aligned and approximately the same size. This is accomplished by subtracting the blackest color in the image from all colors, multiplying by a matrix to align the gray axis of the image with that of the destination gamut, multiplying by a constant to set the overall contrast, adding the value of the printer's black, and fine tuning by small shifts along the gray axis. The accuracy of individual colors is not emphasized; by observing how the entire image gamut compares to the destination gamut, a reasonable overall mapping is found. Best results were obtained by mapping most, but not all, of the pixel values to the inside of the color gamut in this first step. The next two transformations bring the remaining colors in-gamut.

The second step, sometimes needed for highly saturated images, is called the *umbrella transformation*. The umbrella transformation is similar to the Ives-Abney-Yule compromise [13] and desaturates an image by moving the primaries used to represent the image towards the white point. A color R, G, B, given by $RR_s + GG_s + BB_s$ is mapped to

$RR_n + GG_n + BB_n$ where R_s, G_s, B_s are the source primaries and R_n, G_n, B_n are the new primaries. Hue shifts can be achieved using this transformation by allowing R_n, G_n, and B_n to deviate from the lines between the original primaries and the white point. As with the first transformation, it was found best not to map all colors inside the destination gamut using the umbrella transformation. The reason is that the desaturation is global; therefore, in order to put all saturated colors inside the destination gamut, it would be necessary to desaturate the entire image severely.

The third step, clipping out-of-gamut colors to the gamut surface, is provided so that the global operations of the first two steps are not required to over-compress the image. The clipping scheme maps out-of-gamut colors to a perpendicular projection or to the closest point on the gamut surface when a perpendicular projection does not exist. Note that the exact results obtained by clipping depend on the color space used, X, Y, Z in this case.

One important point about this project is that these tools were targeted for use in their lab by people at least somewhat knowledgeable about color science and printing. It would probably be difficult for non-technical users to get good results by using these tools, since the emphasis was placed on viewing gamut representations, rather than the transformed image itself. Representing the printed image on a calibrated monitor and observing changes directly in the image might be easier to use and might give more predictable results than observing the gamut diagrams. The approach of using soft-proofing in conjunction with interactive control of color mappings has been successfully demonstrated[23].

1.2.3 M.I.T. E.E. Thesis, Robert Buckley

In 1978, Buckley investigated the photographic reproduction of pictures with non-reproducible colors[7]. Buckley's thesis is based on the the *Evan's consistency principle* [11], which suggests that all colors in an image should be compressed equally. According to this theory, colors that are reproducible should be compressed along with the colors that

are not, so that the picture remains "internally consistent." Buckley used the CIE 1964 $U^*V^*W^*$ uniform color space.

Buckley's compression scheme involved compressing lightness (W*) and chromaticness (U*V*) independently while attempting to hold hue constant. In the first step, all lightness values are multiplied by a constant to reduce the density range and shifted to the appropriate range by adding another constant. This affine transformation roughly matches the whitest and blackest colors in the image to the whitest and blackest reproducible colors. During this first step, the chromatic components of the image are not used or changed. The second step, *global chroma scaling*, reduces the chromaticness of all colors in the image by a constant scale factor. This global constant is determined by selecting the largest scale factor which maps every color to the inside of a polyhedral solid model of the color gamut.

Buckley found that global chroma scaling resulted in excessive desaturation. To improve his results, he limited the selection of the constant to a few important areas of the picture. This forced the important areas to be completely in the gamut and imposed no restriction on the rest of the image. This technique allowed more chromaticness and improved the reproduction. Buckley also experimented with sharpening the image and found that it improved the appearance.

1.2.4 M.I.T. Ph.D. Thesis, Jason Sara

In 1984, Jason Sara did a much more involved study of several color gamut compression techniques. Sara used the $L^*a^*b^*$ color space to represent image data. Three categories of algorithms were investigated, two clipping methods and one global scaling method. In the *minimum error gamut compression* (MEGC) method, all in-gamut colors are left unchanged and all out-of-gamut colors are mapped to the closest in-gamut color, as measured by the euclidian distance in $L^*a^*b^*$ space. In the MEGC scheme, there is no differentiation between changes in hue, lightness, or chroma; all changes are considered equivalent as long as the euclidian distance is the same. In the *binary search gamut*

compression, each out-of-gamut color is moved towards a "target" color which is near the neutral axis, until the out-of-gamut color lies on the gamut surface. As with MEGC, the binary search gamut compression does not change any colors that are already in-gamut. Sara also tried a smooth gamut compression scheme that involved no clipping to the gamut boundary. Two variations of the smooth gamut compression were investigated. The first, global compression, is identical to Buckley's method. The second was to allow the compression to vary over different hues.

In Sara's opinion, the best results were obtained with the MEGC method. He found these images to be the most colorful and closest in appearance to the originals. Even so, the MEGC method had a few significant problems. Originally, shadows tended to be reproduced too lightly. In an effort to correct this problem, an emphasis was placed on the lightness match when determining the error, which in turn caused very saturated dark colors to be mapped to black. For example, when emphasis was placed on the lightness mapping, dark blues were mapped to black rather than being lightened. Another problem with the MEGC method was that smooth gradients were sometimes reproduced as steps. These artifacts appeared when many out-of-gamut colors were mapped to one color on the gamut surface. This occurred because the gamut was modeled as a polyhedral solid. The vertices of the polyhedron are more likely to be near an out-of-gamut color than other surface points because the vertices "poke out" the most. One last problem with the MEGC method was that some objectionable hue shifts occurred. In an attempt to eliminate this problem, another version of MEGC was tried in which hue was held constant. This method was not preferred by Sara because these images appeared less colorful to him.

The binary search methods had problems similar to those of MEGC. The most significant problem with the binary search methods was that shadows were usually reproduced too lightly. When lightness was held constant in an attempt to remedy this problem, too many light colors were mapped to white and too many dark colors were mapped to black. As with MEGC, the binary search methods sometimes reproduced smooth gradients as

steps.

The smooth gamut compression without clipping had the advantage that it preserved more image detail than the other methods. Unfortunately, it also had a tendency to produce images that look "washed out" as a result of excessive desaturation. Consequently, Sara did not prefer it.

1.3 Overview of the Method

There are three important issues brought forth by the previous approaches to color gamut compression: the degree of adaptiveness of the algorithm over different areas of the gamut, the color space in which the gamut compression is performed, and whether the lightness component of the image receives special attention.

The first issue, discussed in detail in Chapter 5, involves the degree of adaptiveness of the gamut compression method. Methods involving clipping, such as the MEGC method, are the most adaptive because every pixel in the image is processed without regard to the value of any other pixel. Methods such as Buckley's global chroma compression are the least adaptive across the color gamut because every pixel is processed in exactly the same way. From the previous work, we can see that either extreme has its problems. The lack of adaptiveness always results in an overcompressed image. On the other hand, too much adaptiveness, such as in the MEGC method, causes the relationships between colors to be disregarded. When several colors are mapped to the same color, image detail is lost, and false contours may appear in areas that were smooth gradients in the original. In Chapter 5, a compromise between these two extremes which emphasizes adaptiveness is developed.

The second important gamut compression issue is the choice of the particular color space chosen as the domain for the compression. This topic will be discussed in more detail in Chapter 2. In his gamut compression techniques, Sara used the $L^*a^*b^*$ color space, which has three main advantages over linear spaces such as CIE tristimulus values,

X, Y, Z. First, $L^*a^*b^*$ explicitly normalizes all color values with respect to a reference white. Some type of normalization is necessary in order to avoid having the intensity of the light source affect the measurements. For example, in the Xerox project, tristimulus values were translated between a monitor and the printer. The maximum luminance of the monitor gamut was 25 foot-lamberts, whereas the maximum luminance of the printer gamut was 526 foot-lamberts, which was dependent on the intensity of the light source used to measure the print samples. Consequently, the calibrated color values on the monitor could not be used on the printer without using the gamut compression tools, even though the colors were considered printable. The second advantage of $L^*a^*b^*$ is that it has good perceptual uniformity. The Xerox gamut clipping scheme did not clip colors to the perceptually closest color, rather, colors were clipped to the closest color in X, Y, Z space. Had the $L^*a^*b^*$ color space been used, the clipping would have minimized a perceptual distance metric, not the X, Y, Z distance, which is poorly correlated to human perception. The third advantage of $L^*a^*b^*$ is that it is an opponent color space, and is a better model of the human visual system. For these reasons, the $L^*a^*b^*$ color space is used for this project.

The third important issue concerning the gamut compression method is whether special processing is devoted to the lightness or achromatic component of the image. Buckley's thesis and the Xerox work both paid special attention to the achromatic component of the image. In both projects the first processing step was to adjust the overall density or lightness range of the image without regard to the chromatic components of the image. This approach appeared to work well. In Sara's thesis, lightness was processed along with the chromatic components. Sara also had many problems with incorrect lightness values in his output images.

The reason why it may be advantageous to process the lightness, density, or achromatic range first is that different devices have very different dynamic ranges with respect to lightness. The printer used to print the images in this thesis can produce no more than a 20:1 ratio between the luminous intensity of the white paper compared to 100%

black. Other devices have much greater dynamic ranges. A good reflection print, for example, may have a 100:1 ratio of intensities, and an image projected in a dark room may have a 1000:1 ratio of intensities[25]. Real scenes viewed by the human eye often encounter intensity ranges which are much greater still. The lightness range of an image must be compressed in order to reproduce it on a device with a smaller dynamic range. Otherwise, only a small range of lightnesses in the image will be distinct, the others are reproduced either as white or black. Buckley's approach, the affine transformation, is a simple method for reducing the lightness range of an image. Unfortunately, the image contrast is reduced along with the lightness range. This is a serious disadvantage of the affine transformation because high image contrast is very important to producing quality reproductions.

There are many reasons to support the idea that the achromatic component of the visual signal is inherently different from the chromatic components. Studies of the human visual system have found that the retina encodes the achromatic and chromatic signals very differently [6]. Spatial perception is almost entirely based on the achromatic signal component. The fact that the chromatic components of an image are relatively unimportant to spatial perception has been applied in numerous applications such as color television, color facsimile, and color image coding. Since the spatial details of the chromatic components of an image are relatively unimportant, the chromatic image components can often be lowpass filtered without significant degradation. The lowpass filtering reduces the bandwidth required to transmit or store the color signal. Since television and facsimile were originally designed for black and white signals, reducing the bandwidth of the color signal is necessary for transmission within the small available bandwidth. For details on such a facsimile system and a discussion of the psychophysical basis for chrominance spatial filtering, see [26].

There are many more applications and examples which illustrate the difference between the achromatic and chromatic signals. For example, most image sharpening techniques only process the achromatic component of the image. Processing the chromatic

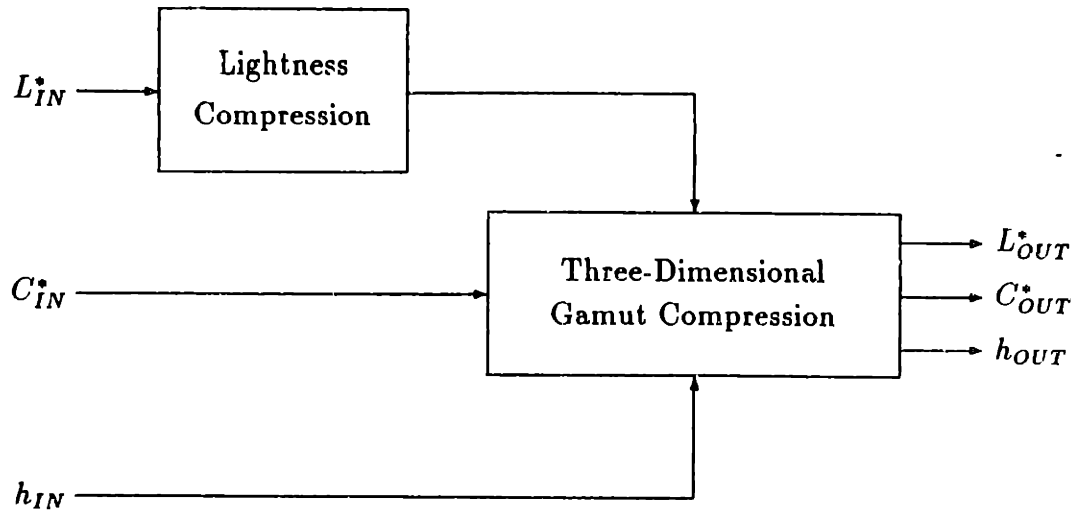


Figure 1-1: General method for color gamut compression

components usually does not enhance the image. One last simple example of the unimportance of spatial accuracy in the color signal can be seen in comic book images. Comic books rely on line drawings for the spatial detail, and allow inaccurate alignment of the color printing. The inaccurate color printing is not a significant problem since the human visual system is relatively insensitive to chromatic spatial detail. The reason why the line drawings are printed in black, and not yellow or blue, is that black lines create the largest changes in lightness and thus add the most to the spatial content of the image.

Since the achromatic component is sufficiently different in nature from the chromatic components of the color signal, it is processed independently as the first step of gamut compression. After the image lightness component has roughly the correct range, the three-dimensional gamut compression method is applied. Figure 1-1 illustrates the general approach. Chapter 4 discusses the lightness processing step, and Chapter 5 discusses the three-dimensional gamut compression step in more detail.

Chapter 2

Measuring Color

2.1 Tristimulus Values

There are many possible methods for measuring visible light. Humans can see wavelengths of light between about 400 and 700 nm, so, for color measuring purposes, we are only interested in wavelengths within this range. One way of measuring light energy at various wavelengths is to use an instrument called a spectroradiometer. Quantities from a spectroradiometer are not immediately useful for measuring color because the human eye adapts to different overall levels of illumination and detects relative changes of light energy in a scene. A spectrophotometer is another light measuring instrument which is similar to a spectroradiometer except that a spectrophotometer measures relative quantities, such as reflectance or transmittance, as a function of wavelength. A spectroradiometer can be used as a spectrophotometer by normalizing measurements with respect to the light source. Despite the normalization, the human eye measures light differently than a spectrophotometer.

The eye has two main categories of light receptors. Rods are used primarily for night levels of illumination, called scotopic vision. Cones are used primarily for day levels of illumination, called photopic vision. There are three types of cones, each with a different spectral sensitivity. Having three types of daylight sensors is what enables us to see

color. Because there are three types of cones, human color perception is inherently three dimensional [1]. Unlike cones, rods come in only one type, which is why we cannot see color at night. Since human color perception is based primarily on the cones, only vision due to the cones will be discussed.

The difference between the three types of cones is that each is most sensitive to a different range of wavelengths of light. Each cone is most sensitive either to short, medium, or long wavelengths of visible light. The peak sensitivities are located at about 450, 530, and 560 nm. Roughly speaking, the three categories of cones can be thought of as being most sensitive to either red, green, or blue light. All three are sensitive over a fairly wide range and there is considerable overlap in the ranges of wavelengths that each type of cone is sensitive to. The main difference between a spectrophotometer and measurements made by the cones is that a spectrophotometer measures light over many very narrow frequency bands. In order to model the color measuring behavior of the eye, the many narrow bandwidth measurements from the spectrophotometer can be scaled and combined to find the amount of energy over the wide frequency bands that cones would measure. A general model of this process is that cones behave as bandpass filters on the spectral power distribution to produce three signals. Figure 2-1 illustrates the model. Since the cones are packed together closely in the fovea, three signals are available to describe the color of each small area in the scene being viewed.

In Figure 2-1, $E(\lambda)$ is the spectral power distribution of the light. $H_s(\lambda)$, $H_m(\lambda)$, and $H_l(\lambda)$ are linear bandpass filters corresponding to the short, medium, and long wavelength types of cones. R, G, and B are the filter outputs which correspond to the raw signals measured by the eye. These cone "filter responses" or some linear transformation of them are often called *color matching functions* or *color mixture curves* (CMCs). The eye's bandpass filtering operation, illustrated in Figure 2-1, can be expressed:

$$R = \int E(\lambda)\, H_l(\lambda)\, d\lambda \tag{2.1}$$

[1] There is a small range of illumination where both rods and cones are active. In this *mesopic* range, color perception is based on four types of receptors.

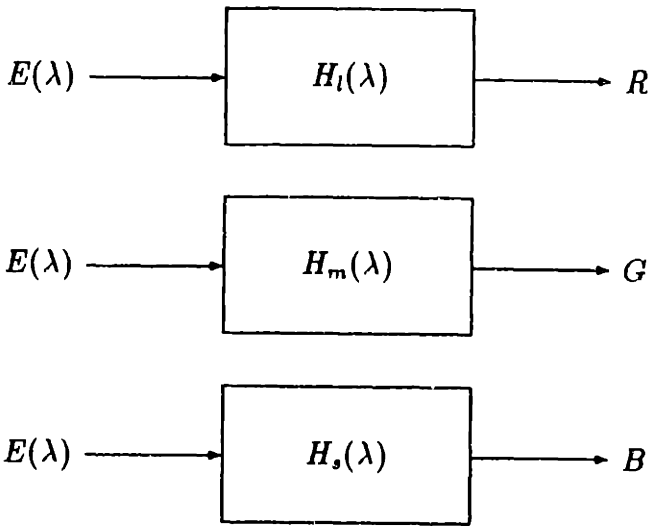


Figure 2-1: Block diagram model of the cones' linear bandpass filtering behavior.

$$G = \int E(\lambda)\, H_m(\lambda)\, d\lambda \tag{2.2}$$

$$B = \int E(\lambda)\, H_s(\lambda)\, d\lambda \tag{2.3}$$

The important point here is that color perception is based on only three signals: R, G, and B, which are called *tristimulus values*. Since the eye has only three different types of color receptors, the color signal has only three degrees of freedom. The exact nature of the spectral power distribution is not important. Consequently, only three numbers are needed to represent a color. Compared to a spectral power distribution, tristimulus values greatly simplify the specification of color.

Since tristimulus values come from the integral sum of a function, the spectral power distribution that produces a particular color sensation is non-unique. Tristimulus values can be determined from the spectral power distribution but the spectral power distribution cannot be determined from the tristimulus values. In order to match two colors for a given set of detectors with sensitivities $H_s(\lambda)$, $H_m(\lambda)$, and $H_l(\lambda)$, it is only necessary to match the tristimulus values obtained at the outputs of $H_s(\lambda)$, $H_m(\lambda)$, and $H_l(\lambda)$, not the spectral power distributions. In fact, tristimulus values only tell us whether two

colors match under similar viewing conditions. If two colors are different, tristimulus values do not describe how the colors are different. The vast majority of color reproduction processes rely on matching tristimulus values since matching spectral power distributions is difficult and unnecessary.

2.2 Linearity

The fact that the bandpass filtering of the cones is essentially a linear process and that combining light energy is a linear process has some important consequences. In the field of color, *Grassman's laws* is the term used to state that light energy and tristimulus values, which are linear functions of light energy, are linear.

One very important consequence of linearity is that, given almost any three spectral power distributions as primaries, we can produce a match to most any color. There are two requirements that must be met to guarantee that a color match can be produced. First, the tristimulus vectors of the primaries must be linearly independent. One way of expressing linear independence is that no linear combination of any two vector primaries can produce the third. The second requirement is that negative amounts of the primaries must be allowed. Although this presents no problem mathematically, negative light energy is not physically possible and thus imposes a real constraint on the gamut of colors that can be produced with a given set of primaries.

Color matching with three primaries can be demonstrated by shining lights on two separate white screens. The color to be matched is projected on the first white screen. The three linearly independent primaries are projected on the second white screen. By adjusting the intensity of each primary, the color projected on the first screen can be matched. When a negative amount of a primary is required, that primary can be added as a positive light on the opposite screen. As long as the three primaries are linearly independent and there is no limitation on their intensity, the two screens can always be made to match.

It is reasonable that only three primaries are necessary to match an arbitrary color because color perception has only three degrees of freedom. As long as there are three primary basis vectors which are linearly independent, then they will span the three-dimensional space of all possible tristimulus vectors. The combination of three primaries to produce an arbitrary color can be written:

$$a_1 \begin{bmatrix} R_1 \\ G_1 \\ B_1 \end{bmatrix} + a_2 \begin{bmatrix} R_2 \\ G_2 \\ B_2 \end{bmatrix} + a_3 \begin{bmatrix} R_3 \\ G_3 \\ B_3 \end{bmatrix} = \begin{bmatrix} R_x \\ G_x \\ B_x \end{bmatrix} \tag{2.4}$$

where $[R_1 G_1 B_1]^T$, $[R_2 G_2 B_2]^T$, and $[R_3 G_3 B_3]^T$ are the tristimulus primary vectors and $[R_x G_x B_x]^T$ is the tristimulus vector to be matched. The weightings of the primaries, a_1, a_2, and a_3, are simply:

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} R_1 & R_2 & R_3 \\ G_1 & G_2 & G_3 \\ B_1 & B_2 & B_3 \end{bmatrix}^{-1} \begin{bmatrix} R_x \\ G_x \\ B_x \end{bmatrix} \tag{2.5}$$

The matrix is guaranteed to be invertible since the tristimulus primary vectors are linearly independent.

Another consequence of linearity is that tristimulus values in terms of one set of three primaries can always be transformed into a new set of tristimulus values in terms of a different set of three primaries. The new set of three primaries can be any three linearly independent tristimulus vectors.

Given tristimulus values a_1, a_2, and a_3 in terms of primaries $[R_{1A} G_{1A} B_{1A}]^T$, $[R_{2A} G_{2A} B_{2A}]^T$, and $[R_{3A} G_{3A} B_{3A}]^T$, a new set of tristimulus values can be found in terms of primaries $[R_{1B} G_{1B} B_{1B}]^T$, $[R_{2B} G_{2B} B_{2B}]^T$, and $[R_{3B} G_{3B} B_{3B}]^T$:

$$a_1 \begin{bmatrix} R_{1A} \\ G_{1A} \\ B_{1A} \end{bmatrix} + a_2 \begin{bmatrix} R_{2A} \\ G_{2A} \\ B_{2A} \end{bmatrix} + a_3 \begin{bmatrix} R_{3A} \\ G_{3A} \\ B_{3A} \end{bmatrix} = b_1 \begin{bmatrix} R_{1B} \\ G_{1B} \\ B_{1B} \end{bmatrix} + b_2 \begin{bmatrix} R_{2B} \\ G_{2B} \\ B_{2B} \end{bmatrix} + b_3 \begin{bmatrix} R_{3B} \\ G_{3B} \\ B_{3B} \end{bmatrix} \tag{2.6}$$

Solving for the new set of tristimulus values, b_1, b_2, and b_3, gives,

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} R_{1B} & R_{2B} & R_{3B} \\ G_{1B} & G_{2B} & G_{3B} \\ B_{1B} & B_{2B} & B_{3B} \end{bmatrix}^{-1} \begin{bmatrix} R_{1A} & R_{2A} & R_{3A} \\ G_{1A} & G_{2A} & G_{3A} \\ B_{1A} & B_{2A} & B_{3A} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{2.7}$$

Note that it is not necessary for the new set of primaries to be physically realizable. Negative tristimulus values can always be eliminated if the primaries are not constrained to be realizable.

As an example of transforming primaries, suppose we are given tristimulus values in terms of the phosphor primaries of one monitor. We could convert these tristimulus values into a new set of tristimulus values in terms of the phosphor primaries of a different monitor using Equation 2.7. In practice, additional steps are necessary because monitor intensity is usually not a linear function of the input signal.

Although a color match can be produced using any set of three linearly independent primaries, an important distinction is that color cannot be measured using any set of three linearly independent primaries. An intuitive example which shows that an arbitrary set of primaries will not work, is to consider using three narrow-band filters to measure color. If a monochromatic light source was being measured, it is possible that the signal would would be missed entirely. The information is lost and there is no way to determine what the tristimulus values would have been if the spectral sensitivities of the cones had been used as the primaries.

It is not necessary, however, to use the exact spectral sensitivities of the cones to measure color. Any invertible linear combination of the cones' spectral sensitivities is essentially equivalent. We can measure a spectral power distribution colorimetrically using any set of filters $H_1(\lambda)$, $H_2(\lambda)$, and $H_3(\lambda)$ given by:

$$\begin{bmatrix} H_1(\lambda) \\ H_2(\lambda) \\ H_3(\lambda) \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} H_l(\lambda) \\ H_m(\lambda) \\ H_s(\lambda) \end{bmatrix} \tag{2.8}$$

Since the matrix **A** is invertible, $H_l(\lambda)$, $H_m(\lambda)$, and $H_s(\lambda)$ are still known.

In practice, the filters used to measure color in devices such as scanners and television cameras are often not a linear transformation of the cones' spectral sensitivities. This may be a source of significant error when measuring color. A common practice is to find a "best fit" matrix **A** in Equation 2.8 to relate the device's actual measurement primaries to a set that is linearly related to the cones' spectral sensitivities. As more emphasis is placed on colorimetrically accurate reproduction, this discrepancy between ideal and practice will probably disappear.

2.3 CIE Tristimulus Values

Tristimulus values greatly simplify color measurement because they are based on a model of the human eye. Tristimulus values also present a problem because different people's eyes are different. Fortunately, the differences are small for most people, but people with defective color vision may have very different spectral sensitivities than people with normal color vision. The problem of abnormal color vision primarily affects men. Depending on race, between 8% and 3% of males have some form of deficient color vision. About 0.4% of all females have deficient color vision[35]. Because of differences between people with normal color vision, the Commission Internationale de l'Éclairage (CIE), an international organization for color measurement, has defined a set of color matching functions for the *standard observer*[8]. The CIE color matching functions are denoted $\overline{x}$, $\overline{y}$ and $\overline{z}$. They approximately correspond to red, green, and blue wideband filters respectively. Figure 2-2 shows the CIE color matching functions [2].

Using CIE terminology, Equations 2.1, 2.2, and 2.3 become

$$X = \int E(\lambda)\,\overline{x}(\lambda)\,d\lambda \tag{2.9}$$

$$Y = \int E(\lambda)\,\overline{y}(\lambda)\,d\lambda \tag{2.10}$$

[2]Slightly different definitions of $\overline{x}$, $\overline{y}$ and $\overline{z}$ exist for different angular fields of view.

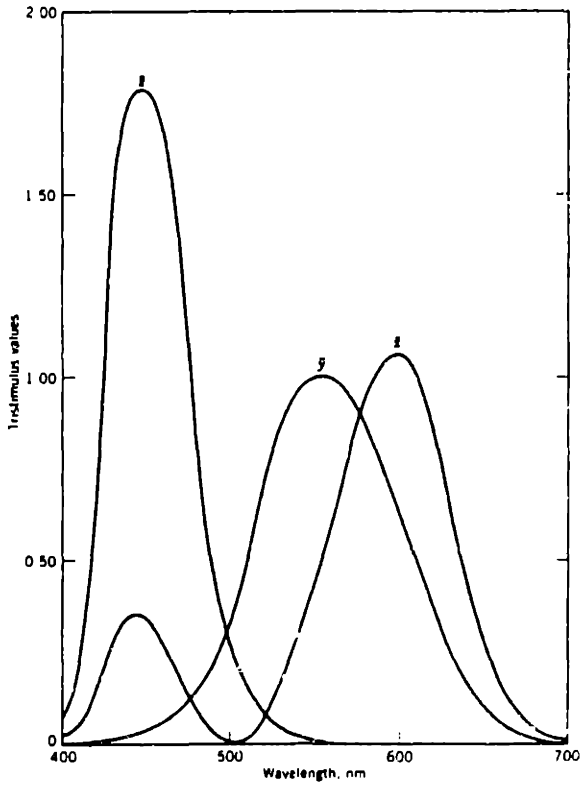


Figure 2-2: The 1931 CIE 2° standard observer [3] .

$$Z = \int E(\lambda)\,\overline{z}(\lambda)\,d\lambda \tag{2.11}$$

X, Y, and Z are called CIE tristimulus values.

When defining the standard observer, the CIE was free to chose any linear combination of color matching functions. This particular set, $\overline{x}$, $\overline{y}$ and $\overline{z}$, achieves two goals. First, the $\overline{y}$ curve is equal to the achromatic luminous efficiency function. Thus, the CIE tristimulus value Y is equal to luminous intensity. Second, $\overline{x}$, $\overline{y}$ and $\overline{z}$ are non-negative. This is not particularly important now, but originally, it made colorimetric computations easier.

Chromaticity coordinates are a very common linear transformation of CIE tristimulus values X, Y, and Z:

$$x = \frac{X}{X+Y+Z} \tag{2.12}$$

$$y = \frac{Y}{X+Y+Z} \tag{2.13}$$

$$z = \frac{Z}{X+Y+Z} = 1 - x - y \tag{2.14}$$

Chromaticity coordinates provides a simple way to graph X, Y, and Z in two-dimensions. Since $x + y + z = 1$, only two of the three chromaticity variables need to be plotted. Usually, x and y are graphed in the first quadrant of a cartesian grid. The luminous intensity, Y is often given with x and y in order to specify a color, and is plotted as the z-axis of the chromaticity diagram.

Since chromaticity is a linear transformation of tristimulus values, functions which are straight lines in terms of tristimulus values are mapped to straight lines in chromaticity coordinates. Additive color mixtures of two primaries in the chromaticity diagram must produce a color which lies on the line between the two primaries. In general, given some arbitrary number of additive primaries, the gamut of reproducible colors is the region inside the polygon (or polyhedron in three-dimensions) which has the primaries as its vertices.

2.4 Additive and Subtractive Processes

We have explained how to measure color given a spectral power distribution, but the factors that made the spectral power distribution what it is have not been discussed. In this section, the processes between the light source and the eye which affect the spectral power distribution will be briefly explained.

The simplest process that changes a spectral power distribution is additive light mixing. Additive light mixing is simply the sum of light energy for each wavelength. Examples of devices which operate by an additive process include television monitors and multicolored discs spinning at high speed, which mix colors temporally. The result of additive light mixing is linear and thus easily predictable.

Subtractive processes may occur when light interacts with with some colored material. Surfaces or media that the light is passing through removes light energy at various wavelengths. Subtractive color processes tend to be more complicated than additive pro-

cesses because they are often the result of absorption or scattering. When a material absorbs some wavelengths of light, the energy is usually converted to heat. Scattering is caused by small particles which have an index of refraction different from that of the medium through which the light is propagating. The amount of scattering depends on the refractive index of the particle and the particle's size.

Subtractive processes are often most easily represented as multiplicative processes. The original spectral power distribution is multiplied by a spectral reflectance or transmittance function which varies between zero and one over different wavelengths, attenuating the light energy. Examples of subtractive processes include color printing, paint mixing, and dyes in a solution.

Cyan, magenta, and yellow are often used as the subtractive primaries, whereas red, green, and blue are often used as the additive primaries. This is not necessary. Nothing inherent in these processes dictates that particular primaries must be associated with a particular process. Other primaries could be chosen, but usually the color gamut will decrease in size.

One last interesting process that modifies a spectral power distribution is the fluorescent process. Fluorescent processes absorb light energy at one wavelength and emit it at another, longer wavelength. Typically, light energy is absorbed in the ultraviolet range, which is not visible, and emitted in the visible range. Fluorescent materials are often used as whiteners. Fluorescent whiteners can make an object appear brighter than an object with 100% reflectance if there is sufficient ultraviolet energy in the source, since more visible light comes off of the fluorescent object than was incident.

2.5 Measuring the Color of a Reflective Object

In the previous sections, we discussed how to measure the color of light given its spectral power distribution. In this section we will explain how to measure the color of an object that does not emit light. For example, we may want to measure the color of a piece of

paper. In order to measure the color of most objects, it is necessary to illuminate them. The spectral properties of the light source illuminating the sample will strongly affect the color that is measured. The light energy reflected by an object is the product of its reflectance and the energy of the source for each wavelength:

$$E(\lambda) = L(\lambda)R(\lambda). \tag{2.15}$$

$L(\lambda)$ is the spectral power distribution of the light source. $R(\lambda)$ is the spectral reflectance of the object being measured and ranges between zero and one as a function of wavelength. To measure CIE tristimulus values X, Y, and Z of a reflective sample, Equation 2.15 is substituted into Equations 2.9, 2.10, and 2.11.

In order to measure the color of a reflective sample, some light source must be used. If we know the spectrophotometric curve of a sample, we can calculate X, Y, and Z using a standard *illuminant.* An illuminant is defined by a spectral power distribution, and is not necessarily a real light source. The term *source* refers to a real physical light. It may or may not be possible to create a source which is spectrally equivalent to a given illuminant.

When measuring a reflective sample, it is important to remember that chromaticity is not an adequate specification of an illuminant. Just because two sources have the same CIE tristimulus values and match each other visually does not mean that an object viewed under the two different sources will look the same, nor measure the same X, Y, Z values. The reason is that the multiplication of reflectance by source energy must be calculated on a wavelength by wavelength basis. The integral of a product of two functions of wavelength is different from the product of the integrals.

Since tristimulus measurements of a reflective object depend on the light source, sometimes two objects will match color under one light source, but will not match under another light source. Two objects which match color only under a particular set of illuminants are called a *metameric pair* or *metamers* and are said to exhibit *metamerism.* The two samples of a metameric pair must match under at least one illuminant and must mismatch under at least one illuminant. The only case in which the colors of two objects

match, regardless of the illuminant, is when the objects have identical spectral reflectance curves. Since spectral curves are practically never matched, the problem of finding a color match is equivalent to finding a metameric pair which matches under a given illuminant.

In practice, to determine the tristimulus values of a reflective sample, the integrals of Equations 2.9, 2.10, and 2.11 are computed as summations. Tristimulus values are often computed from a spectrophotometric curve. The spectral reflectance is multiplied by $\bar{x}$, $\bar{y}$, and $\bar{z}$ at many discrete values of wavelength and then summed to determine X, Y, and Z. Discrete tables of CIE $\bar{x}$, $\bar{y}$, and $\bar{z}$ are available in many books about color[3] [36].

2.6 The $L^*a^*b^*$ Color Space

In section 2.3, two color spaces were introduced: CIE tristimulus values, X, Y, Z, and chromaticity coordinates, Y, x, y. In fact, using Equation 2.8, an infinite number of linearly related color spaces can be defined. These linear color spaces are poor choices as a domain for color gamut compression because they make no attempt to model the human eye. Consequently, numerical differences in tristimulus or chromaticity coordinates are poorly correlated to perceptual differences. In order to approach this problem, the CIE recommended in 1976 the use of the $L^*a^*b^*$ (CIELAB) and $L^*u^*v^*$ (CIELUV) color equations. By incorporating nonlinearities which are well correlated with experimental results, $L^*a^*b^*$ and $L^*u^*v^*$ provide numbers which are a better descriptor of color appearance. I chose to use the $L^*a^*b^*$ color space over $L^*u^*v^*$. One slight advantage of choosing $L^*a^*b^*$ for this thesis was that many test images specified in $L^*a^*b^*$ were already available.

$L^*a^*b^*$ is defined in terms of CIE tristimulus values:

$$L^* = 116\left(\frac{Y}{Y_o}\right)^{\frac{1}{3}} - 16 \tag{2.16}$$

$$a^* = 500\left[\left(\frac{X}{X_o}\right)^{\frac{1}{3}} - \left(\frac{Y}{Y_o}\right)^{\frac{1}{3}}\right] \tag{2.17}$$

$$b^* = 200\left[\left(\frac{Y}{Y_o}\right)^{\frac{1}{3}} - \left(\frac{Z}{Z_o}\right)^{\frac{1}{3}}\right] \tag{2.18}$$

Equations 2.16, 2.17, and 2.18 are valid for $(\frac{X}{X_o}) \geq .01$, $(\frac{Y}{Y_o}) \geq .01$, and $(\frac{X}{X_o}) \geq .01$. $L^*a^*b^*$ is defined for smaller values of X, Y, and Z, but these definitions are not needed for most image processing applications because such small values are practically never encountered. X_o, Y_o, and Z_o in Equations 2.16, 2.17, and 2.18 are the CIE tristimulus values of reference white.

The variable L^* only depends on CIE Y, the luminous efficiency function, and is called lightness. Evans defines lightness as the visually apparent reflectance of a surface under a given set of conditions [11]. There are many possible ways to define a perceptual lightness scale that is well correlated to the physical variable luminance. L^* is the CIE psychometric lightness scale and takes on positive values less than 100. More accurate lightness scales are possible but they are also more complicated. As mentioned in Section 1.1.2, brightness also depends on the viewing field, and attempts to account for it can make the function even more complicated [2]. The logarithm function is sometimes used as a lightness scale, and provides interesting mathematical justifications for some image processing techniques, but it is not as accurate as L^* in Equation 2.16.

The variables a^* and b^* are organized as an opponent color system. The opponent color system is based on the idea that somewhere in the visual signal transmission path between the eye and the brain, color is coded into a red–green signal and a yellow–blue signal [6]. In the $L^*a^*b^*$ color space, a^* is the red-green signal and b^* is the yellow-blue signal. The variables a^* and b^* are signed real numbers where positive values of a^* and b^* are redish and yellowish, and negative values of a^* and b^* are greenish and blueish respectively. For example, in this system orange would be in the $+a^*$, $+b^*$ (red, yellow) quadrant, and purple would be in the $+a^*$, $-b^*$ (red, blue) quadrant. All achromatic colors, such as white, black, and gray have $a^* = b^* = 0$.

$L^*a^*b^*$ is graphed in three-dimensional cartesian coordinates with a^* corresponding to the x-axis, b^* corresponding to the y-axis, and L^* corresponding to the z-axis. In many

image processing applications it is useful to have the L^* image component separate from the chromatic image components, a^* and b^*. For example, in color image enhancement, often only the lightness component needs to be processed, and in color image coding, the coding requirements for the lightness image component are different from the coding requirements for the chromatic components.

One of the main advantages of $L^*a^*b^*$ for color gamut compression is that it has good perceptual uniformity while being relatively simple to calculate. A good measure of the perceptual difference between two colors can be calculated as the straight-line euclidian distance between the two color points in $L^*a^*b^*$ space:

$$\Delta E_{ab} = \sqrt{\Delta L^{*2} + \Delta a^{*2} + \Delta b^{*2}} \tag{2.19}$$

An accurate perceptual distance metric is important for color gamut compression because it provides a simple means for estimating the perceptual impact of a color change.

In cylindrical coordinates, the $L^*a^*b^*$ cartesian color coordinates become the $L^*h_{ab}C^*_{ab}$ color coordinates, which are used in Chapter 5. The definitions of metric hue angle (h_{ab}) and metric chroma (C^*_{ab}) depend only on a^* and b^* and are given by the standard rectangular to polar conversion:

$$h_{ab} = \arctan\left(\frac{b^*}{a^*}\right) \tag{2.20}$$

$$C^*_{ab} = \sqrt{a^{*2} + b^{*2}} \tag{2.21}$$

Through the rest of this document, the ab subscripts are assumed. Lightness, L^*, is the z-axis in both $L^*a^*b^*$ and L^*hC^*. Hue angle, h, is the main determinant of a color name. Hue tells whether a color is, for example, green, yellow, or orange. Chroma indicates how much of a hue is in a color; whether it is pale or deep, weak or strong. Chroma is correlated to the vividness, purity, chromaticness, or colorfulness of a color. The less black, white, or gray a color has, the higher its chroma. Chroma is also related to saturation, although saturation is not defined in $L^*a^*b^*$ space, because saturation is only defined for linear transformations of CIE chromaticity coordinates. In the $L^*u^*v^*$ color space, saturation is equal to chroma divided by lightness.

When using Equation 2.19 to measure the color error of a particular color change, it is sometimes useful to separate the error into components. The component errors ΔL^*, Δa^*, Δb^*, and ΔC^* are simply the arithmetic difference between the two values of the variable. Hue error cannot be calculated this way since it is an angle measure. The hue angle error is given by

$$\Delta h = \sqrt{\Delta E^2 - \Delta L^{*2} - \Delta C^{*2}} \tag{2.22}$$

In summary, color should be measured in the same way that the human eye does. The fact that the human eye reduces spectral power distributions of light to a three-dimensional signal makes color matching practical. The methods for measuring color are well defined in terms of the standard observer. There are many candidate color spaces for performing color gamut compression. The $L^*a^*b^*$ color space is one good choice. In Chapter 4, L^* is processed to reduce the lightness range of the image and in Chapter 5, C^* is processed during the three-dimensional gamut compression.

Chapter 3

Measuring the Gamut

In this chapter, the method used to measure the gamut of the printer is discussed. The particular device measured is a Hewlett-Packard Paintjet[1] color printer. The Paintjet is a four-color, 180 dpi (dots per inch horizontally and vertically), thermal ink jet printer. The dots are placed at regular fixed intervals in a square grid. At any point in the grid, one of eight colors is possible. Using only one color at a time, cyan, magenta, yellow, black, and white (no ink) can be printed. Using two primaries at once, red, green, and blue can be produced. Black can not be mixed with any of the colors. All printing is done on Hewlett-Packard Paintjet paper[2], which is a coated paper designed for use with the Paintjet printer.

As mentioned in Chapter 1, the color gamut of an output device can be represented as a solid in a three-dimensional color space. Colors contained within the solid are reproducible, and colors outside of the solid are not reproducible. In order to gamut-compress an image for display on a particular device, it is necessary to know exactly what the gamut of the device is.

One approach is to determining the gamut of a printer is to develop a model, and calculate the color gamut from the model. For example, the color gamut of a thermal

[1] Hewlett-Packard catalog number HP 3630A.
[2] Hewlett-Packard catalog number HP 51630P.

ink jet printer can be calculated from the Kubelka-Munk reflectance equation [10]. If one were planning to specify the gamut of many printers it might be useful to study methods for calculating the gamuts without measuring each of them. Only one or two printers were to be measured for this project, so it did not make sense to develop elaborate models, since the models would need to be tested against measurements anyway. The approach taken is to measure the gamut of the printer by measuring many color print samples.

Since the gamut is a continuous solid, it would be necessary to print an infinite number of color patches to measure every point in the gamut. Even if all colors could be measured, most of them would lie on the interior of the gamut solid, and thus would give no additional information about its shape and size. In order to minimize the number of print samples, only the colors that lie on the gamut surface should be printed. Fortunately, for the Paintjet printer, the colors which lie on the gamut surface are easy to predict.

Since the Paintjet printer can print only one of eight colors at any given point, the gamut, to a first order approximation, is linear. The human visual system integrates the small, relatively distinct dots. If nonlinearities from ink mixing and dot overlap were not present, the color gamut could be modeled perfectly by a polyhedral solid in a linear color space. Figure 3-1 illustrates the polyhedral gamut solid which, for the Paintjet, is a dodecahedron. The eight primaries are the vertices of the dodecahedron. Since the printing process is not completely additive, we do not expect the polyhedral model to be accurate, but we can see that the colors which form the gamut surface are produced by only twelve combinations of the primaries. Each of the twelve combinations, corresponds to one face of the dodecahedron in Figure 3-1.

The twelve combinations of three primaries which produce colors on the gamut surface are:

Y G W	Y G K
Y R W	Y R K
C B W	C B K
C G W	C G K

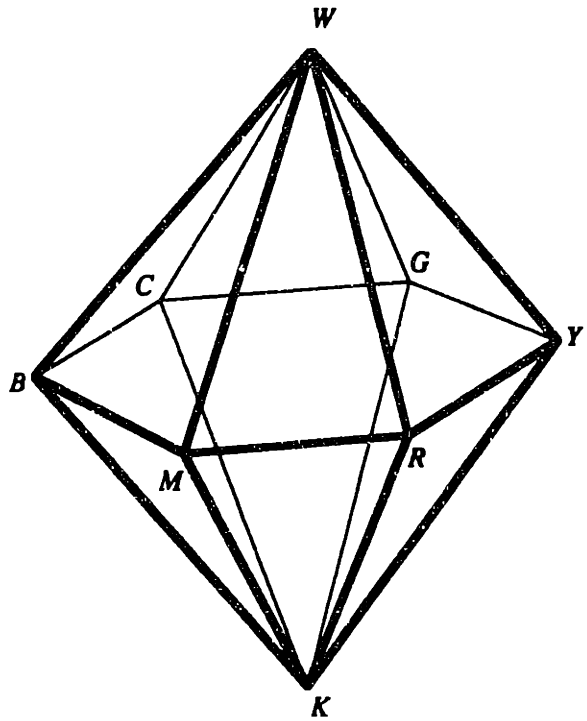


Figure 3-1: A polyhedral gamut.

M R W	M R K
M B W	M B K

The sum of the amounts of each of primary must be 100%. Note that any of the three quantities in the combination can be zero, giving the one and two-color combinations which lie on the gamut boundary. Each of the combination of primaries corresponds to one of the twelve faces of the polyhedral solid. One page of color print samples is devoted to each face of the dodecahedron. Figure 3-2, shows the arrangement of one page of test samples. The patches at the three vertices of the diagram are printed with 100% of a primary color. There are six steps between each pair of primaries. When all twelve pages of color test patches are printed, black and white are printed six times, other primaries four times, two-color combinations twice, and three-color combinations once. This gives 296 unique colors on the surface of the gamut.

In order to print a patch with a given percentage of each of the three primaries, it is

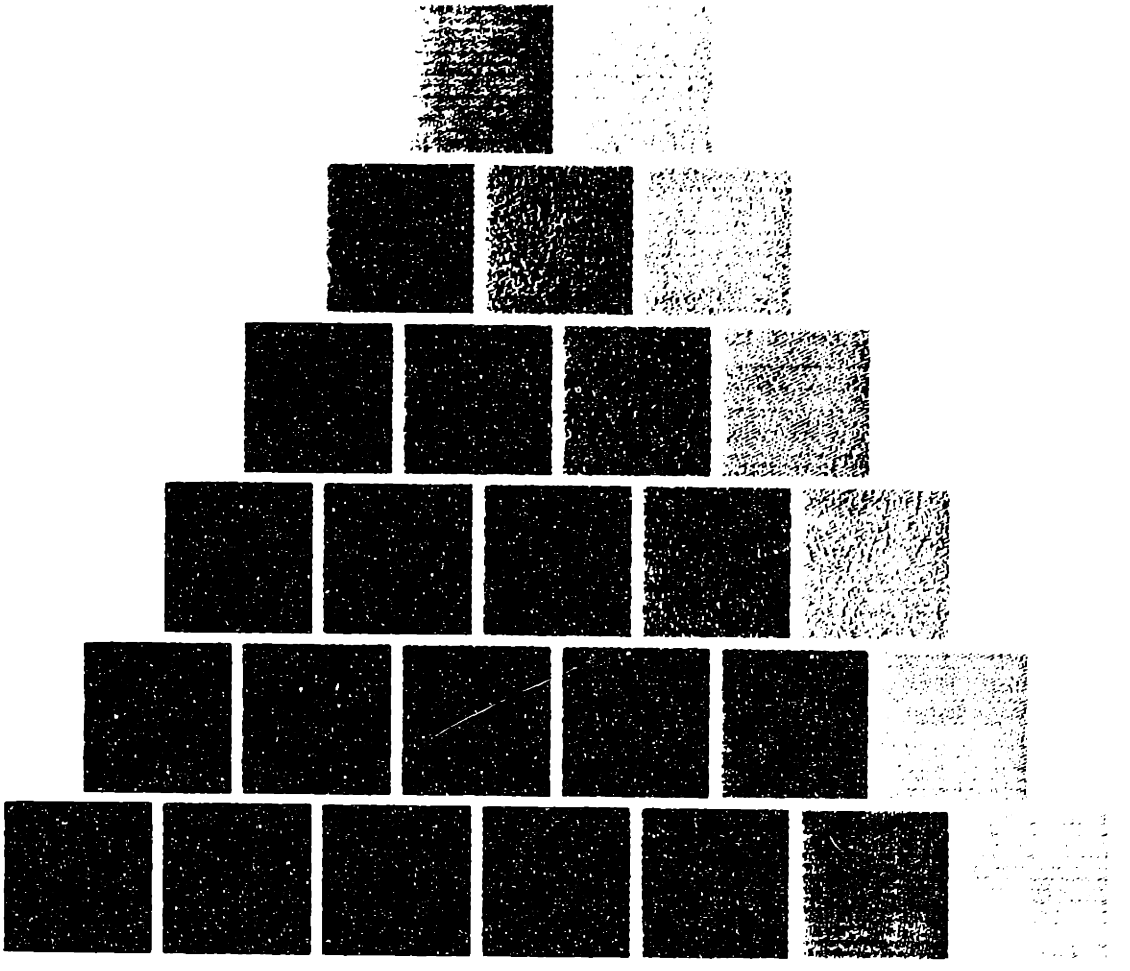


Figure 3-2: Arrangement of color test patches for color gamut measurement.

necessary select a method of arranging the dots in a square grid. Probably the simplest method in terms of implementation is to choose each color randomly. The probability of selecting each primary is fixed at the desired ratio. The color of one point in the grid is given by the outcome of each random trial. Although this method is simple, it does not work well because it produces "splotchy" test patches. The problem with this probabilistic method is that dots of one color are clustered together too often.

Another simple approach is to place the dots deterministically, such that colors are alternated as much as possible. This works much better than the simple probabilistic approach but has the disadvantage that regular patterns are produced for some ratios of the primaries, which causes different amounts of dot overlap. This is not a serious problem but it does cause uneven sampling of the gamut surface. One way to eliminate this problem is to make the dot selection algorithm slightly probabilistic in order to break up the patterns (for a more detailed look at dot patterning see [34] [32]).

After all twelve pages of test patches have been printed, each test patch is measured with a colorimeter[3]. Figures 3-3 and 3-4 show various views of the measured color samples. The mesh connects color test patches which are adjacent in Figure 3-2.

Since the measured values only describe the gamut at discrete points, it is necessary to interpolate in order to specify other values. Since the samples are spaced closely together, the values inside each triangle in the mesh of Figures 3-3 and 3-4 are interpolated from the plane which passes through the three sample points. Essentially, the 296 distinct color samples define a 588-sided polyhedron. Figure 3-5 shows three views of the interpolated gamut. To avoid having to search for the appropriate color points and interpolate them every time a gamut value is needed, a lookup table (LUT) was created. The gamut is stored in cylindrical coordinates in order to be immediately useful for the three-dimensional gamut compression of Chapter 5. The maximum printable chroma is stored as a function of lightness and hue angle in the LUT.

The fact that the white paper has a different chromaticity than 100% black ink causes

[3]The colorimeter used was a Minolta DP-100, which assumes a D_{65} illuminant.

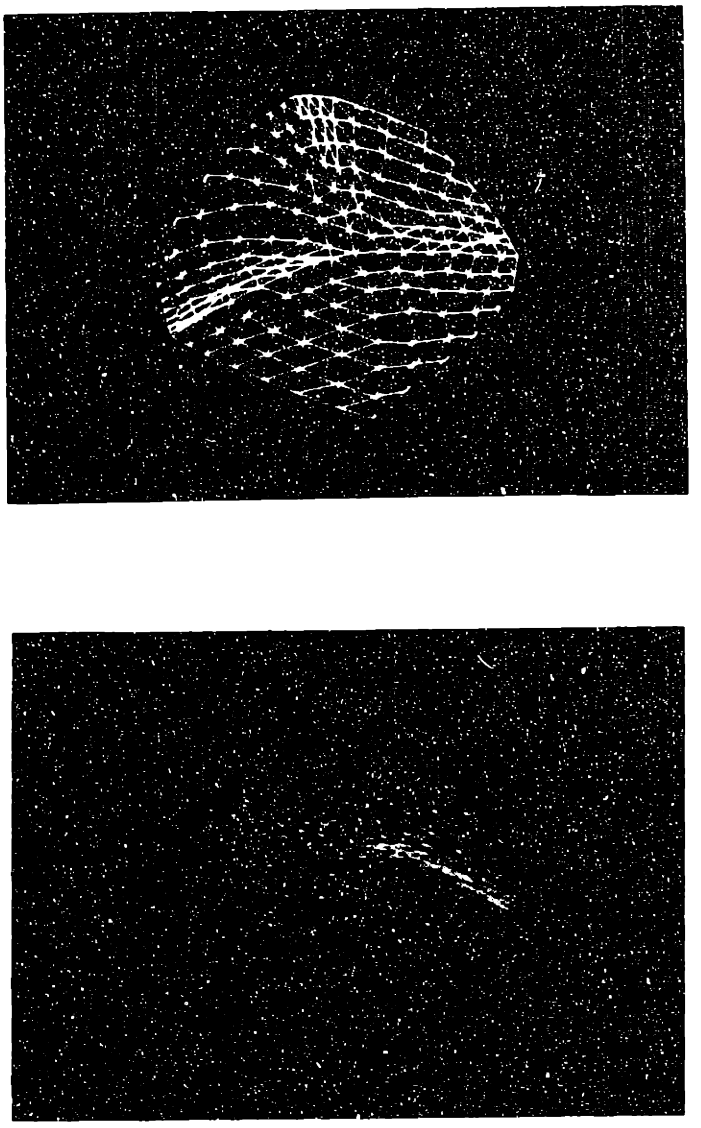


Figure 3-3: Projections of the top-half and bottom-half of the Paintjet color gamut.

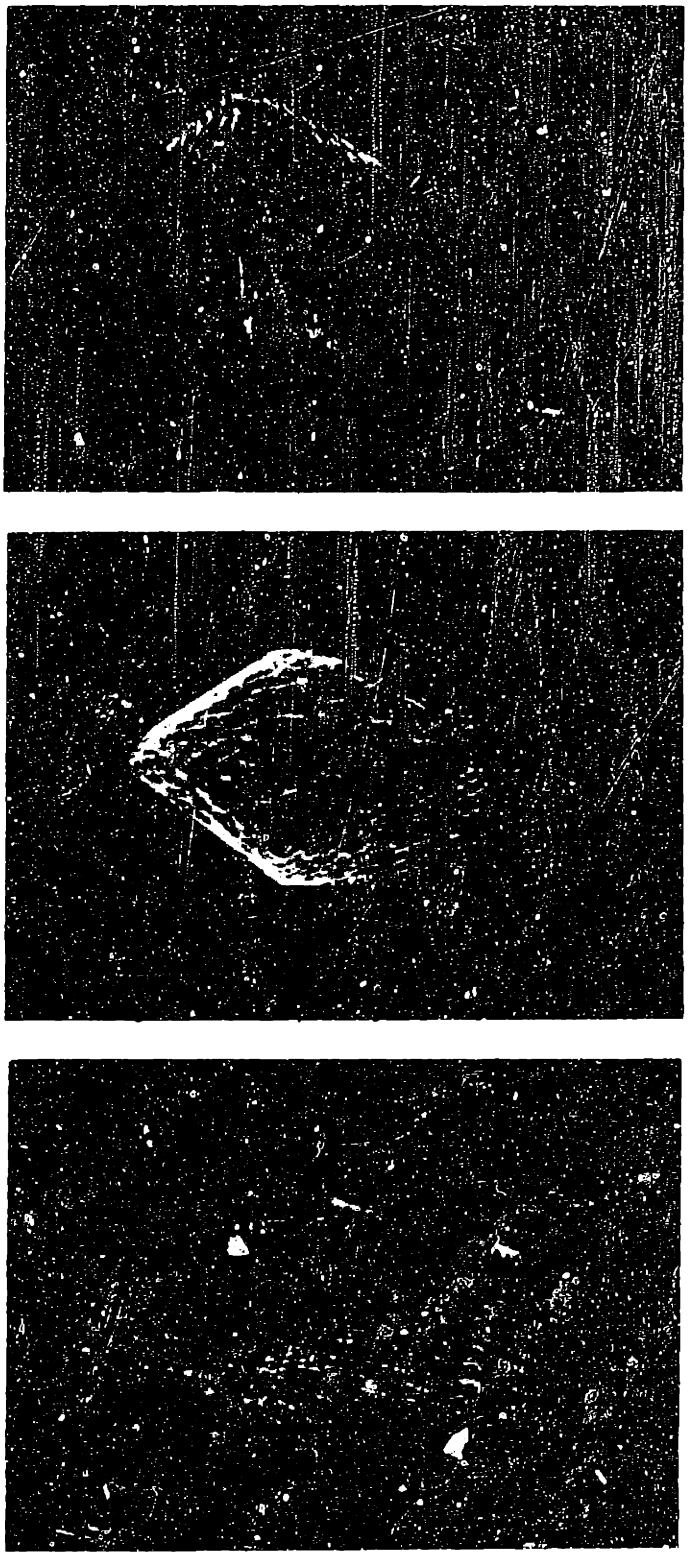


Figure 3-4: The top-half, whole, and bottom-half of the Paintjet color gamut.

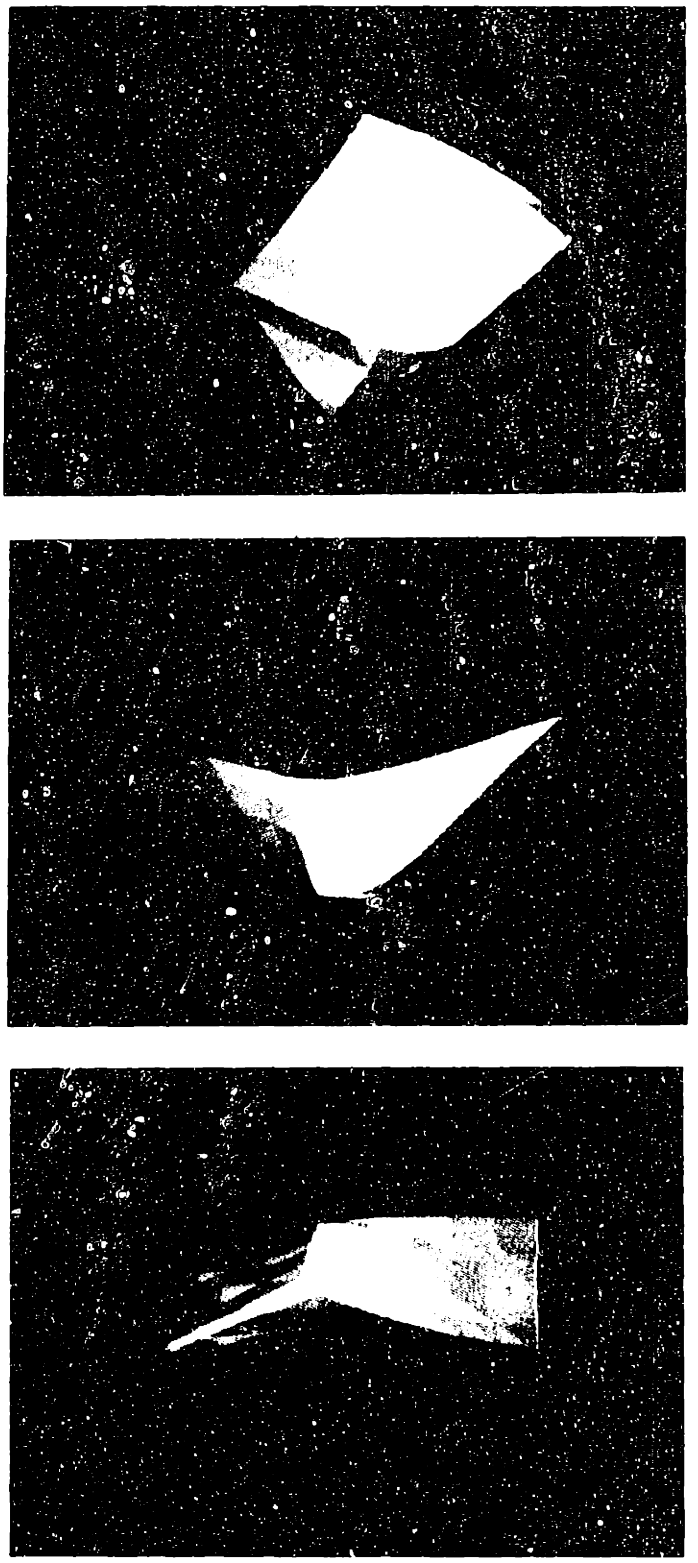


Figure 3-5: Three views of the interpolated Paintjet color gamut.

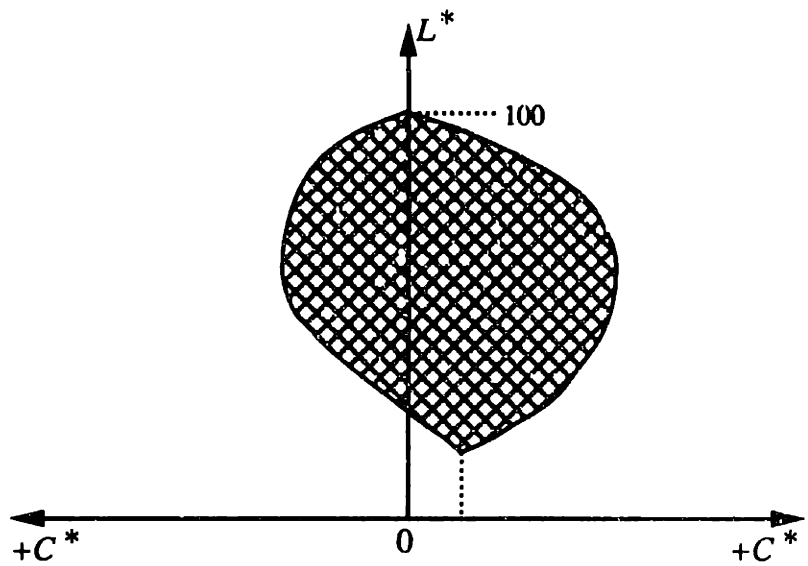


Figure 3-6: Color gamut when black and white have different chromaticities.

a small problem for creating the LUT. The problem is that both the white point and the black point of the gamut solid cannot lie on the L^*-axis ($C^* = 0$) if their chromaticities are different. Figure 3-6 illustrates the situation. The problem is that for low lightnesses, C^* has two values at some hues and is unspecified at others. One solution to this problem is to tilt the coordinate system slightly, so that both the white point and black point of the printer gamut lie on the same axis. Consequently, the color space is not truly $L^*a^*b^*$ or L^*hC^* anymore. Although this slightly modified version of L^*hC^* is what is actually used as the domain for gamut compression, the change is so small that the two are not distinguished. No inaccuracies or inconsistencies result, it just means that when we refer to compressing C^* in Chapter 5, the chroma is actually being moved towards the line which passes through the white point and black point of the printer gamut, not the L^*-axis. The white point of the printer gamut lies on the L^*-axis by definition.

Figure 3-7 shows the values of the final lookup table, and the shape and symmetry of the gamut. The vertical axis is lightness and the horizontal axis is hue. The maximum chroma of the printer gamut for a given lightness and hue is proportional to the lightness of the picture at those coordinates. The slight false contours which appear in Figure 3-7

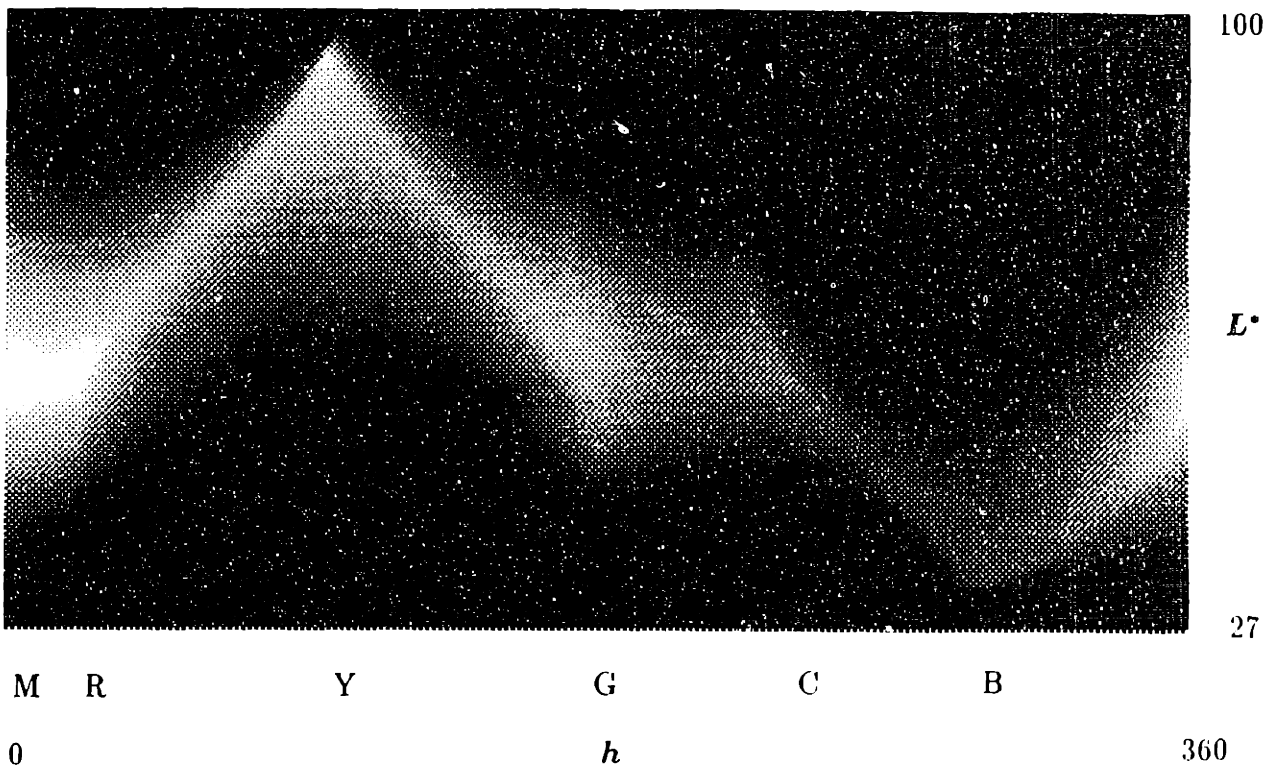


Figure 3-7: The interpolated color gamut lookup table.

are a result of the dithering used to print the figure, not the the LUT values.

Chapter 4

Lightness Compression

As discussed in Section 1.3, the lightness component of an image is given special attention by being processed independently as the first step of the color gamut compression. The lightness processing compresses the overall lightness range of the image to fit within that of the rendering device. The chromatic components, a^* and b^*, or C^* and h, are not considered in this chapter. In the next chapter, the second gamut compression step, three-dimensional gamut compression is discussed.

Given an image and an output device, there are three possible situations with respect to the lightness ranges. First, if the lightness range of the image exactly matches the lightness range of the printer, then there is no problem, and the image is likely to be reproduced well. In the second case, when the lightness values of the image do not use the entire lightness range of the printer, lightness compression is not necessary, and in fact it may be possible to enlarge the lightness range of the image. This second case will not be discussed because it relates more to image enhancement than color gamut compression. The third case, when the lightness range of the image exceeds the lightness range of the printer, is case of interest for this chapter.

Figure 4-1 illustrates the lightness compression problem. As discussed in Section 2.6, L^* is the CIE psychometric lightness scale and C^* is metric chroma. For printing, having L^* values too large is not a problem, because $L^* = 100$, which is the largest possible

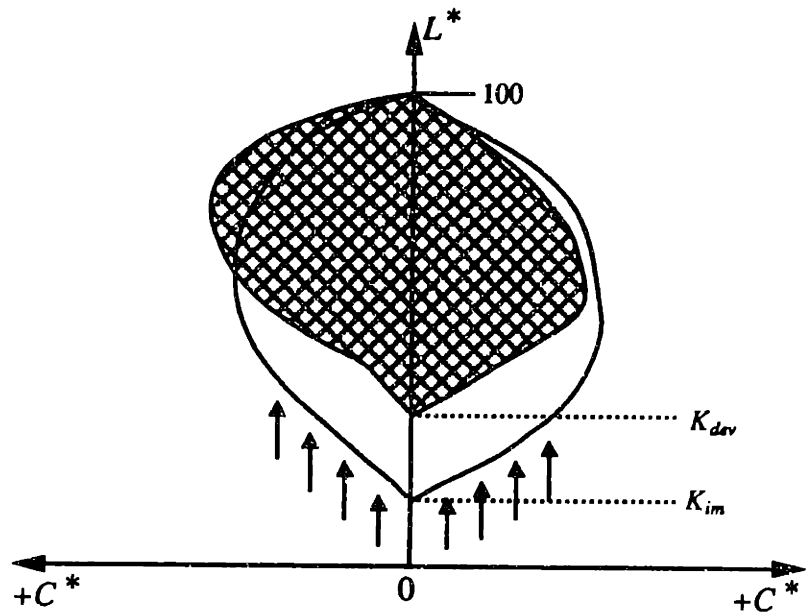


Figure 4-1: Lightness Compression of the color gamut.

value of L^*, is defined to be paper white. The lowest value of L^* depends on the dynamic range and may be very different for the image and the output device. In Figure 4-1, K_{dev} is the lightness value of the printer's 100% black and K_{im} is the lightness value of the darkest color in the image. If an image in which $K_{im} < K_{dev}$ is not compressed in lightness, then all image lightness values in the range $K_{dev} \geq L^* \geq K_{im}$ are printed as black, which results in a very dark image.

4.1 Compression Methods

A variety of methods can be used to compress the lightness of an image. In this section, two particular methods are discussed.

The simplest method of lightness compression is a straight line, or affine, transformation.

$$L^*_{OUT} = \gamma L^*_{IN} + L_o \tag{4.1}$$

The endpoints of the line are fixed by setting the white point of the image to the white

point of the printer and setting the black point of the image to the black point of the printer. Assuming the image contains lightness values in the range $K_{im} \leq L^* \leq 100$, then the parameters of the affine transformation are given by:

$$\gamma = \frac{K_{dev} - 100}{K_{im} - 100} \tag{4.2}$$

$$L_o = 100(1 - \gamma). \tag{4.3}$$

The affine transformation has the problem that it significantly reduces image contrast, which is proportional to the parameter γ. Since γ is less than one, contrast is decreased and the image tends to look "dull" and "flat". The goal of this chapter is to be able to reduce the lightness range of the image while minimizing degradations due to the reduction of image contrast.

Image contrast and sharpness are both very important to image quality. Most nonexperts do not distinguish between sharpness and contrast, and in general apparent contrast is directly related to apparent sharpness [24]. Contrast is related to differences in lightness, whereas sharpness is related to how quickly or slowly the changes in lightness take place. Contrast can also be thought of in the frequency domain. If the contrast of the high frequencies in an image is decreased, then sharpness will be decreased. Conversely, if the contrast of the high frequencies in an image is increased, then the sharpness will be increased.

Empirically, we know that high frequencies are important to image quality. One of the most common image enchancement techniques is to boost the high frequencies in the image's lightness component. We also know that if an image is blurred by decreasing the high frequencies, the image looks worse. The affine transformation of Equation 4.1 essentially decreases all the frequencies in the image by the factor γ. Since high frequencies are more important to image quality than the low frequencies, ideally, a different value of γ could be applied to each. Fortunately, most of the dynamic range of an image is contained in its lower frequencies, so it is possible to reduce the dynamic range of the entire image by reducing the contrast of the lower frequencies. Usually, very little of the dynamic range of the image is contained in the high frequencies, so decreasing the

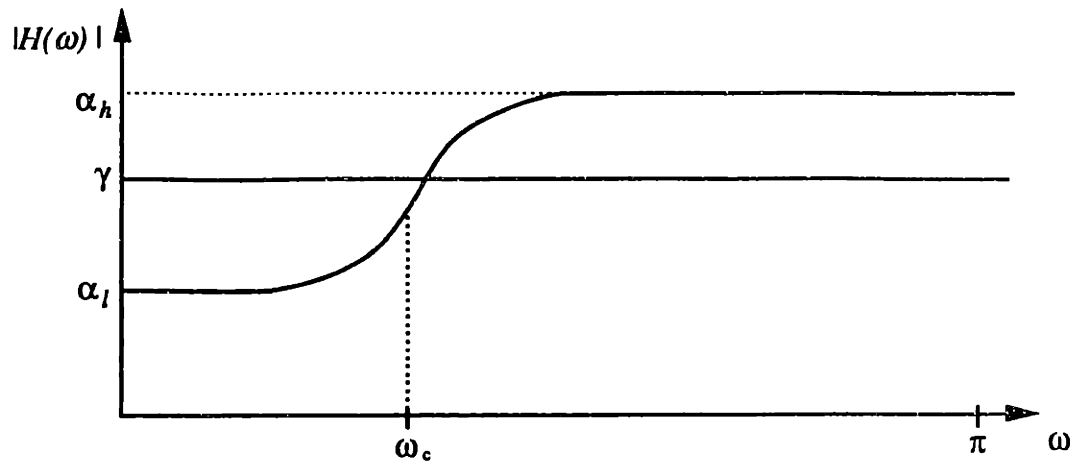


Figure 4-2: The affine transformation compared to LFLC.

contrast of the high frequencies would not decrease the dynamic range much anyway. This approach, *low frequency lightness compression* (LFLC) will be the subject of the remainder of this chapter.

Figure 4-2 compares the affine transformation to the low frequency lightness compression method in the frequency domain[1]. The affine transformation treats all frequencies essentially the same, whereas the LFLC method maintains the contrast of the high frequencies.

This LFLC method is similar to the techniques of unsharp masking [37] and homomorphic image processing [18] [4]. The main difference between these methods and LFLC is the domain in which the frequency modifications take place. Homomorphic image processing filters images in the log domain and unsharp masking operates in the density domain, which is the logarithm of reflectance or transmittance. LFLC by comparison is performed in the CIE psychometric lightness domain, which is explained in Chapter 2, and is a more uniform lightness scale with respect to human perception than the log scale.

[1]The affine transformation is not actually a linear system. The quantity $L^*_{OUT} - L_o$ is a linear system, and is what is graphed in Figure 4-2.

4.2 Separating the Image

4.2.1 Implementation Strategy

The frequency response curve in Figure 4-2 contains three parameters for specifying the LFLC, α_l, α_h, and ω_c. By Parseval's theorem, we know that the energy of the image in the spatial domain must equal the energy of the image in the frequency domain, so, as the parameters α_l and α_h are decreased, the energy of the image in the spatial domain decreases. We know that lower energy signals tend to have smaller amplitudes, but unfortunately it is a difficult problem to determine, a priori, how the dynamic range of the image will change as a function of α_l and α_h. In order to achieve the desired dynamic range, one approach is to iterate, repetitively filtering the image until the correct values of the parameters are found. This approach is impractical because it requires too much computation and time to filter an image. Consequently, a different approach must be found.

In order to reduce the amount of computation, the filter cutoff, ω_c, is held constant. Then the image is separated into a lowpass component and a highpass component, such that the sum of the components is the original image. Each of the image components is scaled before being recombined in order to generate families of filtered images. A block diagram for computing a series of filtered images is shown in Figure 4-3 and a corresponding family of frequency responses is illustrated in Figure 4-4.

The reason why this method saves computation is that the image only needs to be lowpass filtered once to produce the entire family of frequency responses. After the low frequency image is obtained, the high frequency image can be computed by subtracting the low frequency image from the original. Scaling the image components and adding them back together requires a relatively small amount of computation and time.

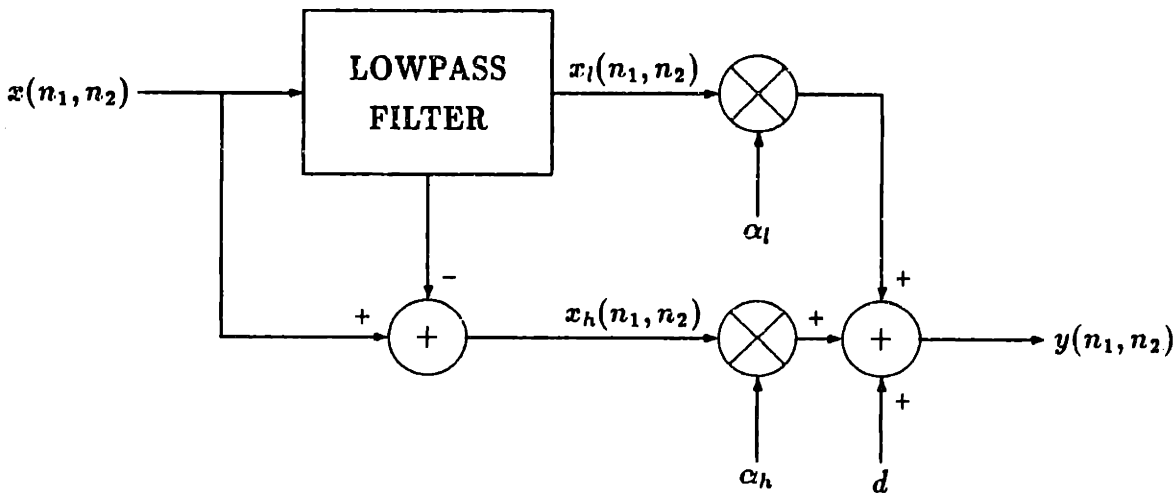


Figure 4-3: Block diagram of the Low Frequency Lightness Compression method.

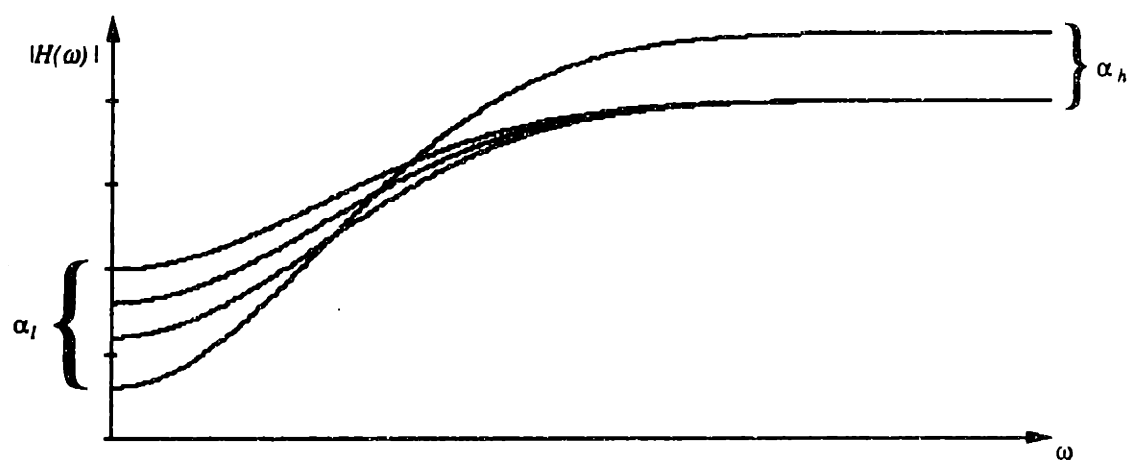


Figure 4-4: Examples of frequency responses from LFLC.

4.2.2 Lowpass Filtering

This section concentrates on the details of the block labeled "LOWPASS FILTER" in Figure 4-3.

Filtering can be thought of and computed in both the spatial domain and the frequency domain. The following two operations describe the filtering in both domains:

$$Y(k_1, k_2) = H(k_1, k_2) \cdot X(k_1, k_2) \tag{4.4}$$

$$y(n_1, n_2) = h(n_1, n_2) * x(n_1, n_2) \tag{4.5}$$

where $x(n_1, n_2)$, $y(n_1, n_2)$, and $h(n_1, n_2)$ are the original image, the filtered image, and the impulse response of the filter respectively. $X(w_1, w_2)$, $Y(w_1, w_2)$, $H(w_1, w_2)$ are the respective discrete Fourier transforms. The '$*$' in Equation 4.5 denotes convolution.

The two-dimensional discrete Fourier transform pair is defined as:

$$X(k_1, k_2) = \begin{cases} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) e^{-j\frac{2\pi}{N_1}k_1 n_1} \cdot e^{-j\frac{2\pi}{N_2}k_2 n_2} & \begin{matrix} 0 \le k_1 \le N_1 - 1 \\ 0 \le k_2 \le N_2 - 1 \end{matrix} \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

$$x(n_1, n_2) = \begin{cases} \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(k_1, k_2) e^{j\frac{2\pi}{N_1}k_1 n_1} \cdot e^{j\frac{2\pi}{N_2}k_2 n_2} & \begin{matrix} 0 \le n_1 \le N_1 - 1 \\ 0 \le n_2 \le N_2 - 1 \end{matrix} \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

Thus, a filter can be completely specified by either $H(k_1, k_2)$ or $h(n_1, n_2)$.

There are two basic approaches to digital image filter design and implementation. One can either design and implement a finite impulse response (FIR) filter or an infinite impulse response (IIR) filter (for details of filter design see [16]). Each method has its advantages and disadvantages. In direct implementations, an IIR filter usually requires much less computation than a comparable FIR filter, but if fast Fourier transforms (FFTs) are used to compute the discrete Fourier transform, then the methods have similar computational requirements. Two-dimensional IIR filters have the disadvantage that they are more difficult to design because it is difficult to test their stability. Two-dimensional FIR filters, in comparison, are relatively simple because stability is

guaranteed. FIR filters have another desirable property; it easy to design FIR filters that have zero phase. Zero-phase filters are desirable because they tend to preserve the shape of the signal in the passband. Zero-phase is more difficult to achieve with IIR filters and requires the cascade of at least two filters. Because of their relative ease of design and desirable properties, only FIR filters are considered in the remainder of this chapter.

Since we desire a zero-phase filter, the magnitude response completely specifies the response of the filter. Frequency sampling is one simple method for designing filters. To design the filter, the values of $H(k_1, k_2)$ are simply the samples of some continuous frequency function. In our case, the values of $H(k_1, k_2)$ might be samples of an ideal lowpass filter magnitude response. This approach has a problem because, in general, the inverse discrete-time Fourier transform will have an infinite impulse response. Consequently, the discrete Fourier transform is aliased in space and the convolution is also aliased. So, simply specifying the filter response in the frequency domain may have undesirable consequences in the spatial domain. Careful attention must be paid to the region of support of the filter, that is, the region in which it is nonzero in space. $H(k_1, k_2)$ must be specified in some way such that $h(n_1, n_2)$ has a small finite region of support.

Windowing is a standard method for limiting the non-zero range of a function [12]. Windows used for the purpose of creating FIR filters must be zero outside the region of support of the filter so that when the window is multiplied by the infinite length signal, the signal is forced to be finite length. Many different types of windows may be used. Some examples include the Hamming, Hanning, rectangular, Bartlett, and Kaiser windows. All of these windows are one-dimensional and must be extended to two-dimensions for image processing. The two most common methods to create a two-dimensional window from a one-dimensional window is to make the one-dimensional window circularly symmetric in two dimensions, or to make the window separable into the product of a window in the first dimension and a window in the second dimension. Windowing in space corresponds to convolving the frequency response of the filter with the frequency response of the window. Thus, windowing tends to add ripple and increase the transition bandwidth of

the frequency response of the original filter.

A second standard approach to designing a lowpass FIR filter is to window the impulse response of an ideal lowpass filter. The two-dimensional ideal lowpass filter is given by:

$$h(n_1, n_2) = \frac{w_c}{2\pi\sqrt{n_1^2 + n_2^2}} J_1(w_c\sqrt{n_1^2 + n_2^2}) \tag{4.8}$$

where $J_1(x)$ is the Bessel function of the first kind and first order (for a derivation see [16]). Figure 4-5 shows an example of an image filtered using the ideal lowpass filter of Equation 4.8 in conjunction with a circularly-symmetric rectangular window. This window has a magnitude of one inside the region $n_1^2 + n_2^2 \leq R^2$ and zero elsewhere, where R is the radius of the circular region of support. The original image is shown in Figure 4-5a, the low frequency component in 4-5b, and the high frequency component in 4-5c, The high frequency component is increased in amplitude and shifted towards gray for display purposes.

Examination of the images in Figure 4-5 shows strong banding around the head of the subject. Banding, which appears where there is a transition between light and dark regions in the image, is undesirable because it may also appear in the final image. The ripple in the filtered image is caused by ripple in the step response of the filter. One way to constrain the step response of the filter to be monotonic, and thus have no ripple, is to require that the impulse response of the filter be non-negative for all indices. This requirement can be seen by considering the convolution of the impulse response with a unit step. Mapping the nonnegative impulse response requirement into a useful frequency domain constraint is not a simple problem. We know that in general the step response will tend to have less ripple as the transition between the stopband and the passband of the filter's frequency response becomes smoother. Thus, we might expect windowing to reduce the ripple of the step response, but we can also see that a window will not eliminate the negative portions in the impulse response of the two-dimensional ideal lowpass filter. Windowing may improve the banding problem but will not eliminate it.

To improve the filtering shown in Figure 4-5, we would like to start with a lowpass filter design whose impulse response is nonnegative in order to eliminate the ripple.

One filter that has this property is the Gaussian filter. In continuous time, the Fourier transform of a Gaussian impulse response is a Gaussian lowpass frequency response. In discrete time, the same is approximately true. The two dimensional Gaussian impulse response can be expressed in the form:

$$h(n_1, n_2) = e^{\frac{-(n_1^2+n_2^2)}{\tau^2}} \tag{4.9}$$

which is clearly circularly symmetric. Written in the form,

$$h(n_1, n_2) = e^{\frac{-n_1^2}{\tau^2}} \cdot e^{\frac{-n_2^2}{\tau^2}} \tag{4.10}$$

it is clear that the two-dimensional Gaussian impulse response is also separable. Separability may be exploited to achieve greater efficiency in some implementations.

Like the ideal lowpass impulse response, the Gaussian impulse response is infinite. This is less of a problem than for most impulse responses because the Gaussian approaches zero rapidly as $n_1^2 + n_2^2$ increases. If a circularly-symmetric rectangular window is applied to the Gaussian impulse response at some $n_1^2 + n_2^2 \gg \tau^2$ then the filter will be largely unaffected. In the implementation used to produce the images in this thesis, the radius of the region of support was somewhat arbitrarily chosen to be equal to 2.6τ, setting the window radius at the point where the amplitude of the impulse response has decreased roughly three orders of magnitude.

To ensure that the filter will not change the overall scale of the image, the sum of the values of the windowed impulse response is set to one:

$$\sum_{n_1} \sum_{n_2} h(n_1, n_2) = 1 \tag{4.11}$$

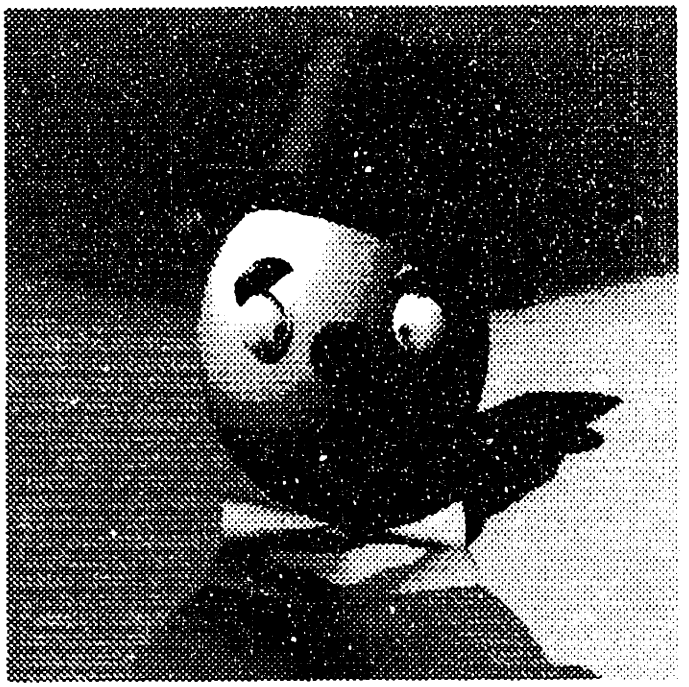
This requires each value of $h(n_1, n_2)$ be scaled by a constant. The normalization is not included in the definition of the Gaussian impulse response of Equation 4.9 because the scale factor depends on the exact characteristics of the window. To avoid having to sum all of the values of the impulse response, the constraint of Equation 4.11 can be met by scaling all frequency values of the filter by $1/X(0,0)$.

An example of an image separated by the Gaussian filter is given in Figure 4-6, which should be compared to the ideal lowpass filtered images of Figure 4-5. The important difference is that the banding has disappeared in the Gaussian filtered image. In the diagram, the original image at the top of the page is separated into the two components at the bottom of the page. As before, the highpass image has been increased in amplitude and shifted towards gray for display purposes.

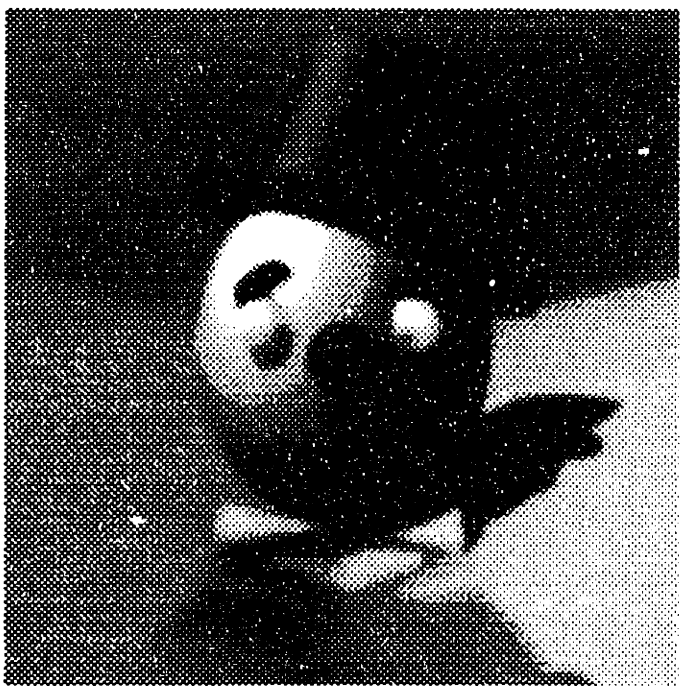
The parameter τ has not been discussed so far because it is held constant for a given family of filter responses. In the two-dimensional ideal lowpass filter, the cutoff frequency was specified by w_c. In the Gaussian filter, the cutoff frequency is inversely related to τ. Since the purpose of filtering is to preserve the high frequencies in the image as much as possible, a lower frequency cutoff is generally preferred to a higher frequency cutoff. The problem is that if the cutoff is too low, then the lightness range may not be able to be reduced sufficiently even if the low frequency image component is completely removed. Experience with selecting τ has shown that there is usually a point at which further decreases in the filter's cutoff frequency necessitates a proportionately much larger reduction in the amount of the low frequency image component that is retained. In the other direction, setting the cutoff too high reduces the advantages of the filtering, and in the limit as the cutoff frequency is increased, the LFLC method becomes equivalent to the affine transformation of Equation 4.1. Fortunately, the exact value of the filter cutoff is not critical. Relatively large changes in the value of the cutoff frequency have only a small effect on the final image. Using the definition of the impulse response given in Equation 4.9, values of τ between about 10 and 30 work well on most images. Examples of images that have been lowpass filtered using the two-dimensional Gaussian impulse response with various values of τ are shown in Figures 4-7, 4-8, 4-9, 4-10, and 4-11[2].

Examples of final images processed with different values of τ are given in Appendix B, and are discussed in Chapter 6.

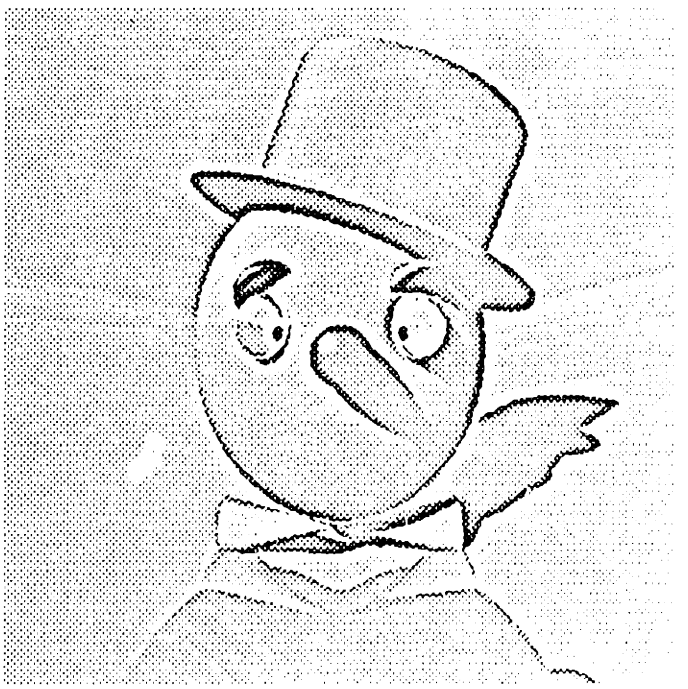
[2]To make the images photocopy better, spatial resolution and the number of gray levels is sacrificed by using a clustered dot dither. No false contours are actually present in the filtered images.



(a)



(b)



(c)

Figure 4-6: Image separated using a windowed Gaussian filter.

Figure 4-7: Original lightness image before Gaussian lowpass filtering.

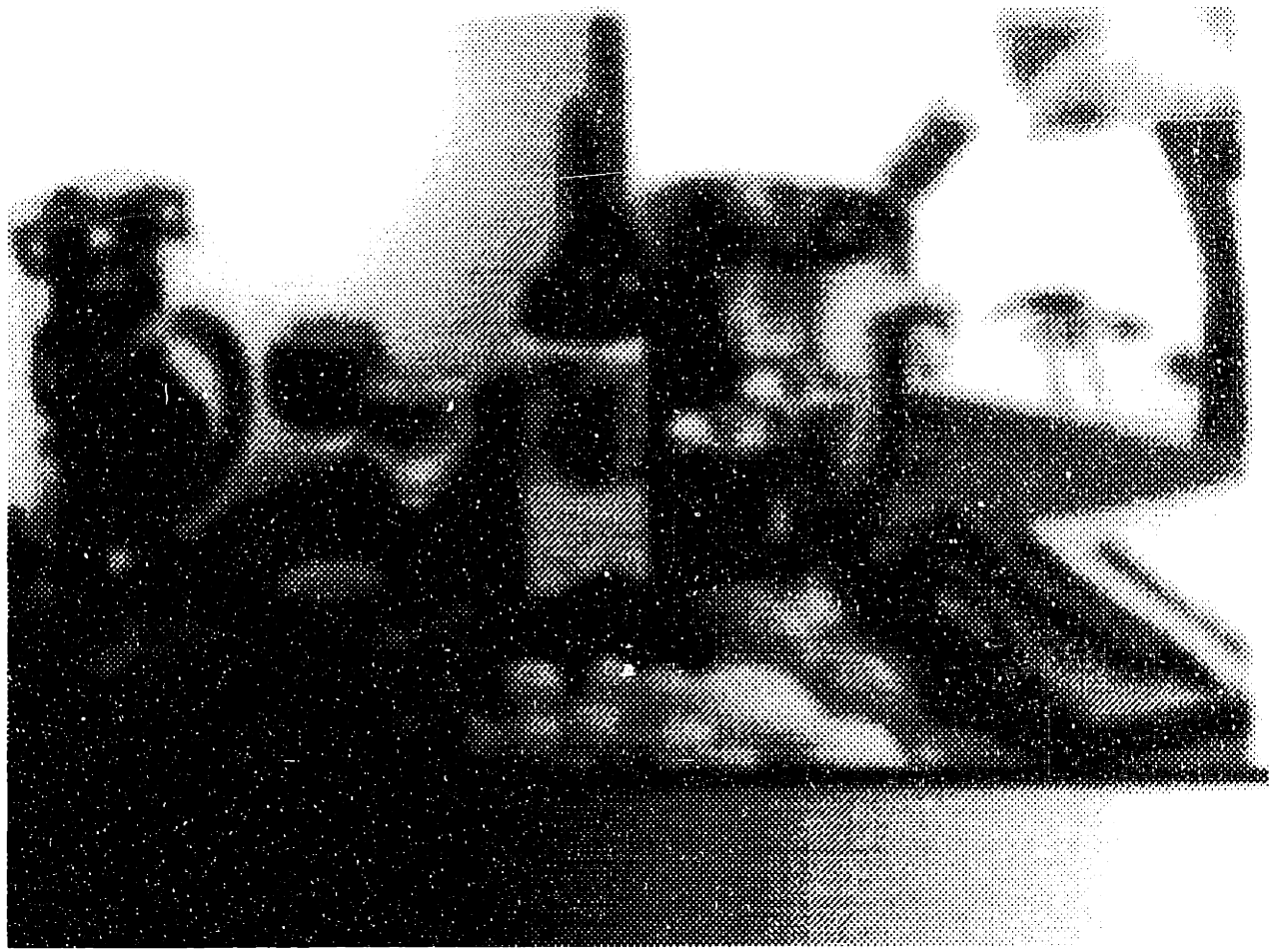


Figure 4-8: Gaussian lowpass filtered image, $\tau = 5$.

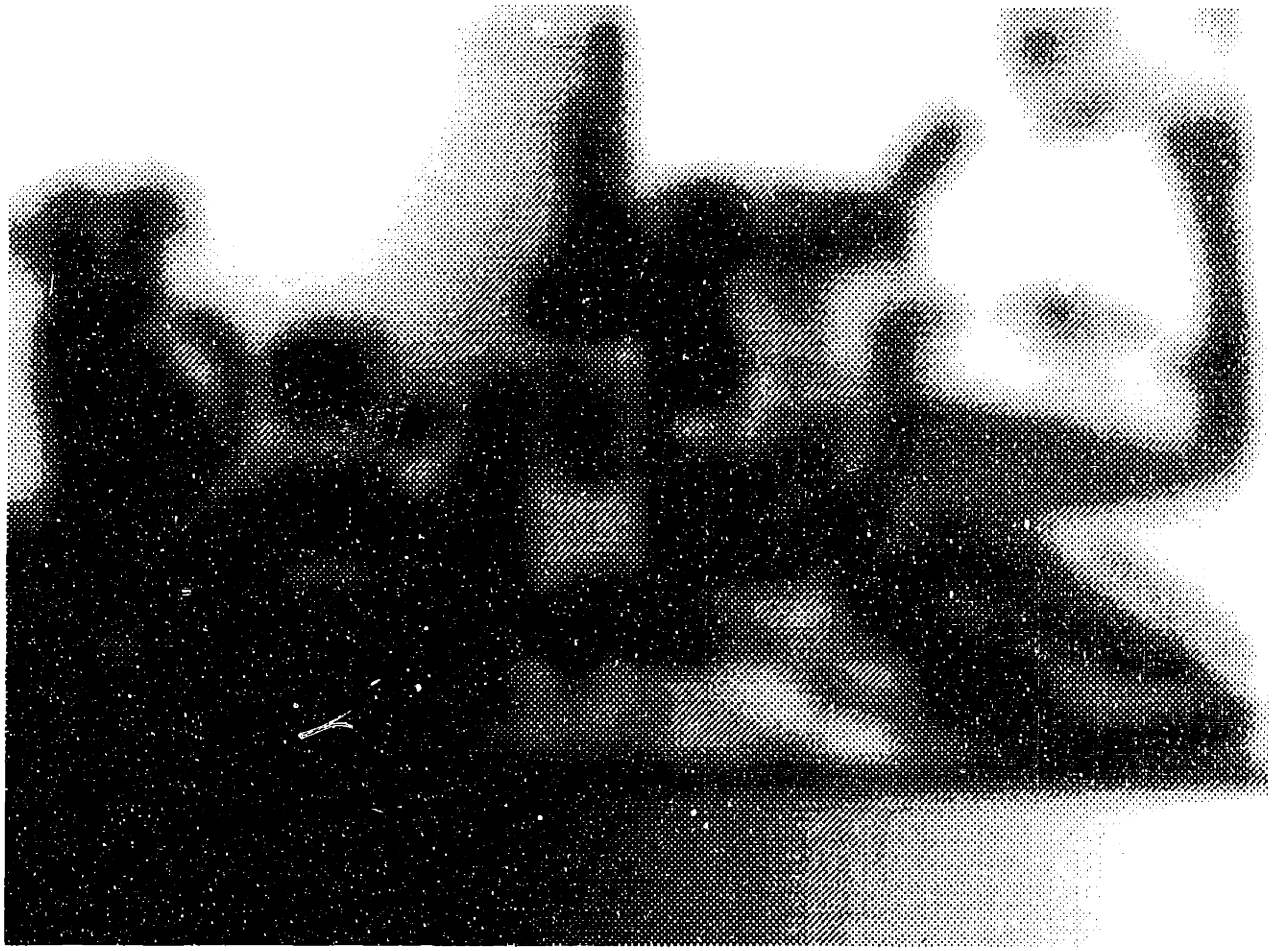


Figure 4-9: Gaussian lowpass filtered image, $\tau = 10$.

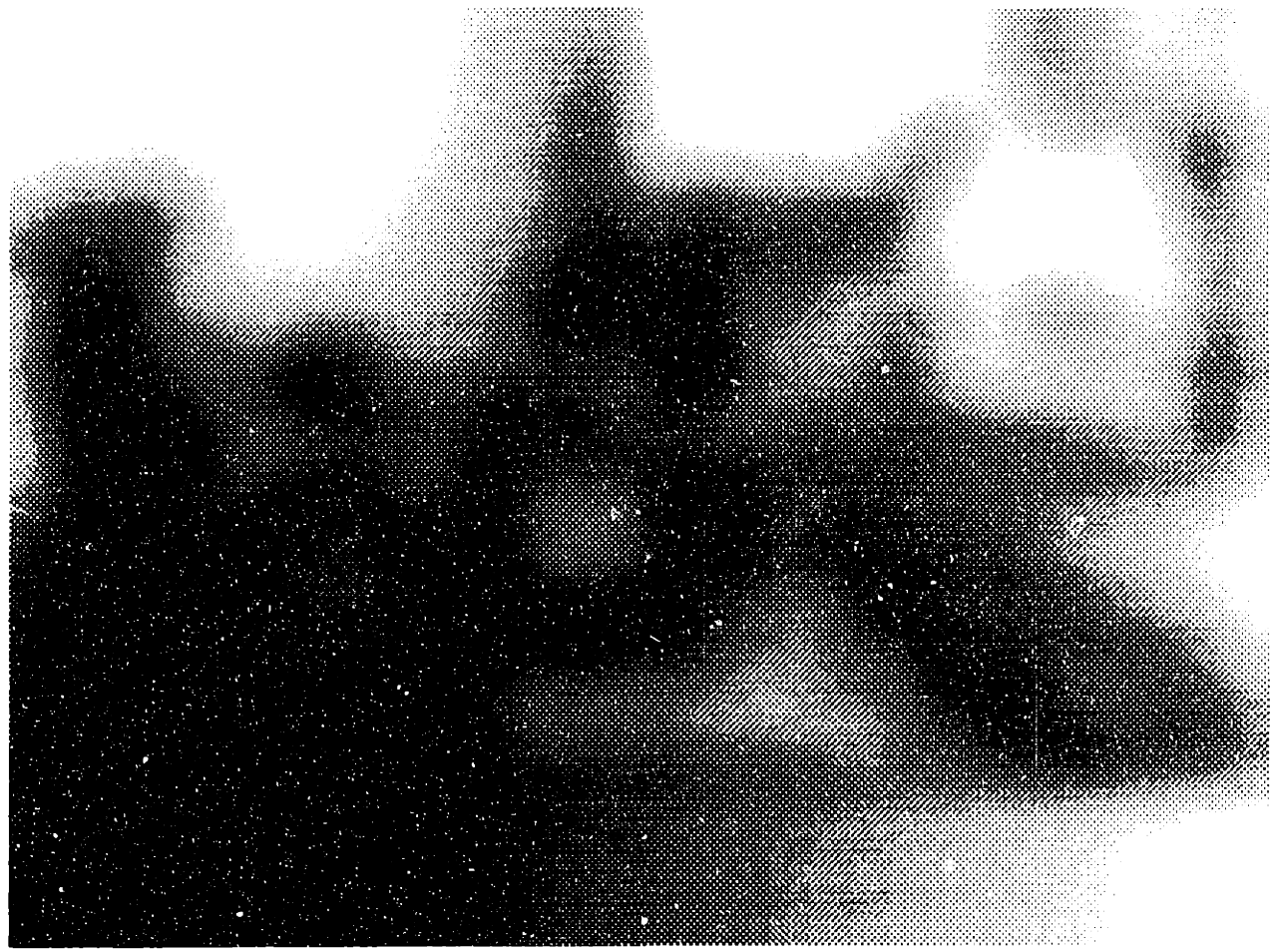


Figure 4-10: Gaussian lowpass filtered image, $\tau = 20$.

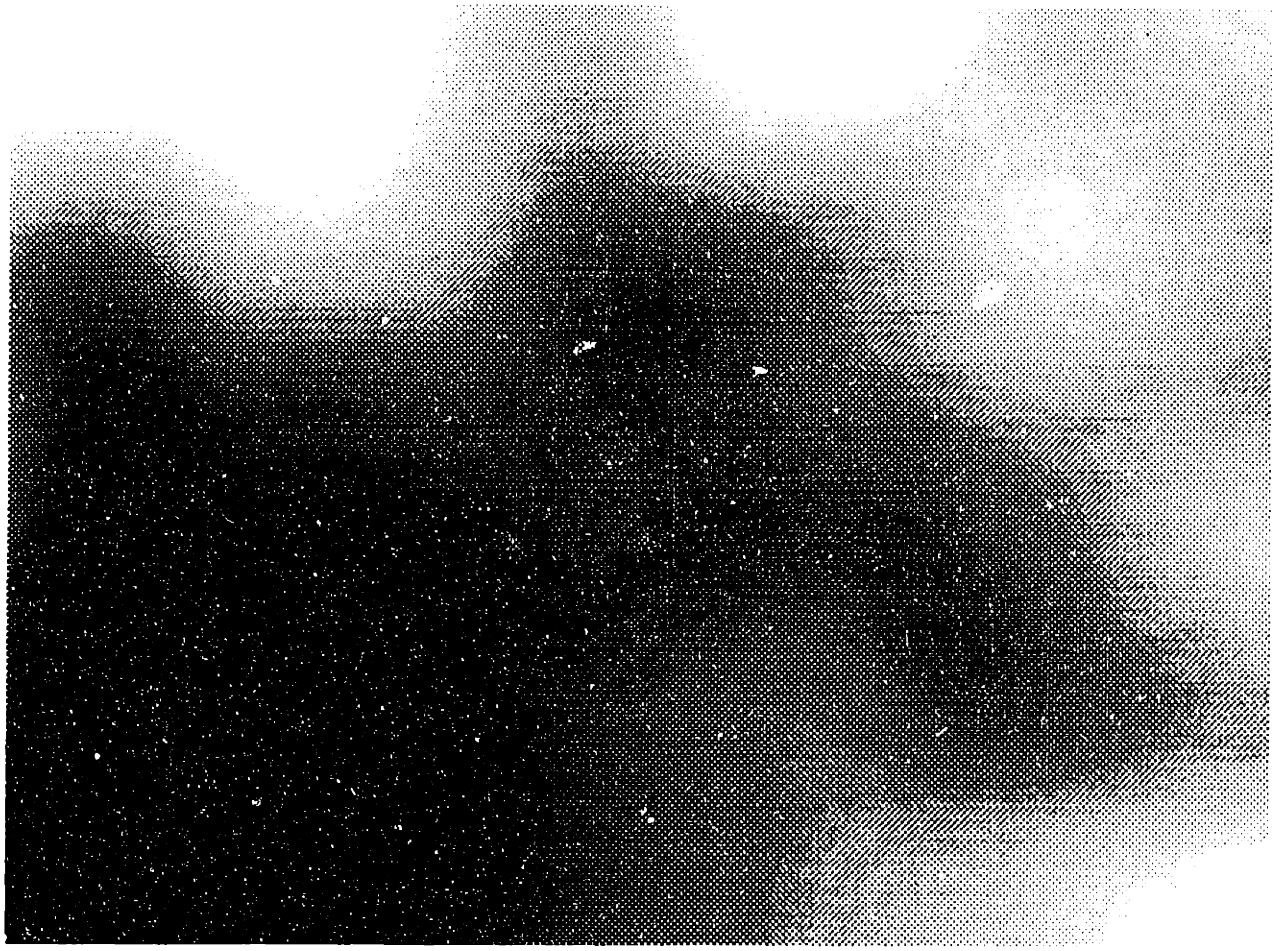


Figure 4-11: Gaussian lowpass filtered image, $\tau = 50$.

4.2.3 Implementation Details

One practical problem that has been ignored so far is the question of what to do about filtering near the image borders. The issue can be seen by looking at the convolution expression that specifies the filtering operation. The convolution of the signal $x(n_1, n_2)$ with the filter impulse response $h(n_1, n_2)$ is given by

$$y(n_1, n_2) = \sum_{m_1=-M_1}^{M_1} \sum_{m_2=-M_2}^{M_2} h(m_1, m_2) x(n_1 - m_1, n_2 - m_2) \tag{4.12}$$

where $h(n_1, n_2)$ has the rectangular region of support $-M_1 \leq n_1 \leq M_1, -M_2 \leq n_2 \leq M_2$. If the original image $x(n_1, n_2)$ has dimensions $N_1 \times N_2$, then to obtain an output image $y(n_1, n_2)$ of the same dimensions, $y(n_1, n_2)$ will depend on $x(n_1, n_2)$ over a range of $(2M_1 + N_1) \times (2M_2 + N_2)$, which is not completely specified.

One trivial solution to the problem is to define $x(n_1, n_2) = 0$ outside the region $0 \leq n_1 \leq N_1 - 1, 0 \leq n_2 \leq N_2 - 1$, and accept the degradation near the image borders. This may be acceptable for some applications and becomes less of a problem when the region of support of the filter's impulse response is small. For the Gaussian filter, a typical cutoff would be given by $\tau = 10$. This would yield an impulse response with a region of support of dimensions 53×53 pixels, although many of the values of the impulse response away from the origin are small. Clearly, the border effects cannot be ignored.

Another solution, often used when the FFT is used to filter the image, is to let the circular convolution of the FFT "wrap around" image values from one border to the opposite border. Conceptually, this approach wraps the filter around to the other side of the image whenever the filter falls over the border. The borders are defined by the size of the FFT used to compute the image. FFTs are usually constrained to be a power of two in length, so the original image is often different in size than the FFT. This requires the rest of the values in the FFT array to be filled in by some arbitrary value. This padding is treated as part of the image with respect to the wrap around. In cases where the image near the border is much different from the padding values, or different from the image on the opposite border, this method may produce undesirable results. These

problems become more significant as the size of the region of support of the impulse response increases, which occurs as the filter cutoff frequency decreases.

Conceptually, the solution used in this thesis is that whenever $h(n_1, n_2)$ is shifted beyond the border of the original image, the values of $h(n_1, n_2)$ that fall off the original image are "folded" back onto the image. This approach ensures that the filtered image depends only on the original image in the same region, not on some arbitrary padding value or on the image content near the opposite border.

Since a direct implementation of folding the impulse response would require different filters near the border, the computation is done in a different way. First, the size of the original image is increased by the size of the region of support of the filter by flipping and appending copies of the image to itself at the borders. To handle the corners, flipped copies of the image are first appended to the top and bottom to increase the height, then the left and right borders are extended. Second, the enlarged image is filtered, taking no explicit care near the borders. Third, the enlarged border region is discarded to obtain an output image of exactly the same dimensions as the input image. Since the size of the original image is increased and the borders of this enlarged image are thrown away anyway, it does not matter how the borders are treated in the computation. Figures 4-12 and 4-13 provide an example of an image enlarged by flipping its borders. For example, the value of $x(-1, -4)$ is given by $x(1, 4)$.

For increased speed, the filtering was computed by taking the discrete Fourier transforms of the filter's impulse response and the original image, multiplying the transforms in the frequency domain, and inverse transforming. The discrete Fourier transforms were computed using a modified version of an N–dimensional FFT program [20]. Large images present a problem for computing two-dimensional FFTs, because they require large amounts of memory, which may or may not be available. On the workstation used to run the filtering program, memory allocation was rarely a problem, but as FFT sizes increased, the time required per pixel also increased. The decrease in speed is caused by the decreased efficiency of the memory system for large amounts of data. A large mem-

Figure 4-12: Image before enlarging borders.

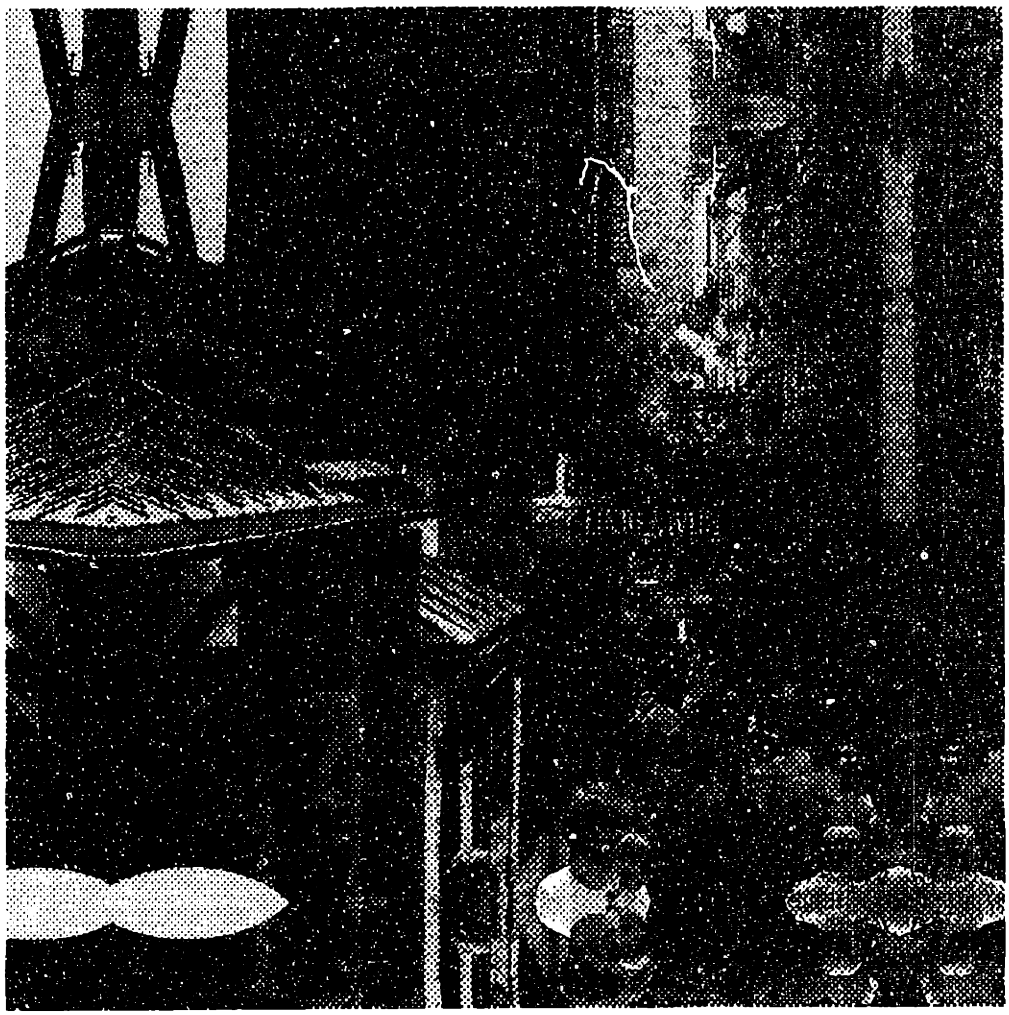


Figure 4-13: Image with enlarged borders.

ory allocation decreases the cache hit rate and causes extreme slowdown if the virtual memory system is required to swap pages. Therefore, it is advantageous, and on smaller computers necessary, to filter the image in sections in order to reduce the size of the FFTs.

There are two common methods for sectioning convolutions, the overlap-add and the overlap-save methods [21]. There is no compelling reason to prefer either one over the other, so the overlap-save method was chosen. To use the overlap-save method, the output image is sectioned into pieces which are slightly smaller than the size of the FFT and each piece is computed separately.

For any given image there are usually several possible ways to break it into pieces. Since the time required to filter the image can change dramatically depending on the sectioning, the program which lowpass filters the image selects the optimal FFT size before beginning the computation. The optimal FFT size depends on the size of the image to be processed, the size of the region of support of the impulse response, and the amount of time that it takes the given workstation to compute the all of the possible sizes of two-dimensional FFTs, which must be powers of two for the particular algorithm. In many cases, this approach gave significant performance improvements.

In summary, in order to compress the lightness range, images are filtered to obtain a low frequency component image and a high frequency component image. Zero-phase FIR filters with a monotonic step response work well. Once the image is separated into two components, the components can be iteratively scaled and recombined to achieve the output image with the desired lightness range and appearance. How the image components are recombined is the topic of the next section.

4.3 Recombining the Image Components

In the previous section, methods were discussed for separating an achromatic image into a low frequency image and a high frequency image such that the sum of the two images

is equal to the original image. In this section, the component images are scaled by constants, recombined, and shifted by a constant lightness to produce an output image of the correct lightness range.

The recombining procedure is described by:

$$y(n_1, n_2) = \alpha_l x_l(n_1, n_2) + \alpha_h x_h(n_1, n_2) + d \tag{4.13}$$

where $x_l(n_1, n_2)$ is the low frequency image computed in the previous section, $x_h(n_1, n_2)$ is the high frequency image obtained by subtracting the low frequency image from the original, and $y(n_1, n_2)$ is the output image. The parameters, α_l and α_h are constant scale factors less than or equal to one, and d shifts resulting image to the appropriate lightness level. These recombination parameters are included in Figure 4-3.

In order to quickly experiment with many values of α_l, α_h, and d, an interactive computer program was created to display the processed image on a monitor. The program allows a variety of parameters to be specified and also computes a histogram of the resulting image. In order to change the value of τ quickly, several different lowpass images are pre-computed to avoid the substantial delay of lowpass filtering. Different low frequency images are read in on demand and output images can be saved to files.

The lightness values of the original image are stored as unsigned characters using one byte per pixel. The low frequency image is stored as floating point numbers using four bytes per pixel. Four bytes per pixel is probably more than necessary, but one byte per pixel for the low frequency image is not sufficient. When only 256 levels are used to represent the low frequency image, false contours sometimes appear in the output image. This quantization error also appears as a series of spikes in the histogram of the output image. When floating point numbers are used for storing the low frequency image, the quantization artifacts disappear and the histogram of the output image looks much smoother.

As mentioned in Section 4.1, decreasing α_h normally does not reduce the dynamic range of the image significantly but does degrade the image significantly. Thus, setting $\alpha_h < 1$ is not a good idea. Although increasing α_h generally improves the appearance of

an image, it is not allowed because it is more related to image enhancement than gamut compression. Consequently, the value of α_h is set to be one.

The parameter α_l is used to decrease the lightness range of the image, and in general must be less than one. The exact value of α_l can be quickly determined using the interactive display program.

One method that often worked well for finding an initial estimate of α_l is based on the approximation that the decrease in the lightness range of the low frequency image component is approximately equal to the decrease in the lightness range of the final output image. Using this assumption:

$$\alpha_l \approx 1 - \frac{K_{dev} - K_{im}}{\max[x_l(n_1, n_2)] - \min[x_l(n_1, n_2)]} \tag{4.14}$$

where K_{im} and K_{dev} are the black points of the image and the device (see Figure 4-1). The denominator terms, $\max[x_l(n_1, n_2)]$ and $\min[x_l(n_1, n_2)]$, are the maximum and minimum lightness values in the low frequency image. Equation 4.14 assumes $K_{im} < K_{dev}$. This approximation works well for $\max[x_l(n_1, n_2)] - \min[x_l(n_1, n_2)] \gg K_{dev} - K_{im}$, that is, as long as the low frequency image component is the dominant contributor to the output lightness range. For low frequency images which have very small lightness ranges, the approximation does not work well. This may occur when the filter cutoff is to low (τ too large for the Gaussian filter). In these cases the lightness range is more dependent on the high frequency image, and it may be impossible to reduce the lightness range of the output image sufficiently without increasing the filter cutoff.

The parameter d, is the last parameter that needs to be specified. The value of d is selected so that the reproduced image will have the same apparent lightness as the original image and the reproduced image is within the lightness range of the printer. Adding a positive value of d is usually necessary because the image needs to be compressed towards the white point rather than zero, which is what $y(n_1, n_2)$ tends to as α_l and α_h become small. As with α_l, d can quickly be determined iteratively using the interactive display program.

Conceptually, to determine an initial estimate of d, the value of d that holds the

peak value of the low frequency image constant after scaling is chosen. Another way of thinking about this process is to first shift the lightnesses of the low frequency image so that the maximum value is moved to zero. Then scale the low frequency image by α_l. Lastly, shift the lightnesses back by exactly the same amount that they were first shifted. This holds the maximum value of the low frequency image constant, and compresses the other values towards this maximum point.

For implementation purposes, it is unnecessary to shift, scale, and shift back. The two shifts can be combined to give:

$$d \approx \max[x_l(n_1, n_2)](1 - \alpha_l). \tag{4.15}$$

When the two image components are recombined using Equation 4.13, the peak-to-peak lightness range is usually greater than the original image. The reason for this is that most images have some small light points in larger dark regions and some small dark points in larger light regions. In each case, the amplitude of the points is not decreased because it is part of the high frequency image which is not attenuated. When the light point occurs in a dark region, the value of the light point is increased along with the dark values. When a dark point occurs in a light region, its value remains nearly unchanged in the output image. These few extraneous lightness values can cause problems is if the lightness range is measured automatically from its endpoints. For example, if the endpoints of the histogram of the output image are used to automatically set the lightness range of the image, a very poor picture will result. The solution to the problem of the extraneous values is to clip them to the minimum and maximum lightness values of the printer. Clipping lightness values is simple and works well.

The LFLC method is not a one-to-one, input-to-output mapping. Pixels of the same lightness in the input image may become different lightnesses in the output image, and pixels of different lightness in the input image may mapped to the same lightness in the output image. Figure 4-14 shows the input-output relationship of the LFLC for the wine image of Figures B-4 and B-5. Likewise, Figure 4-15 shows the input-output relationship of the LFLC for the rose image of Figures B-7 and B-8. In Figures 4-14 and 4-15, the

intensity of the distribution is proportional to the number of pixels having that particular input output relationship.

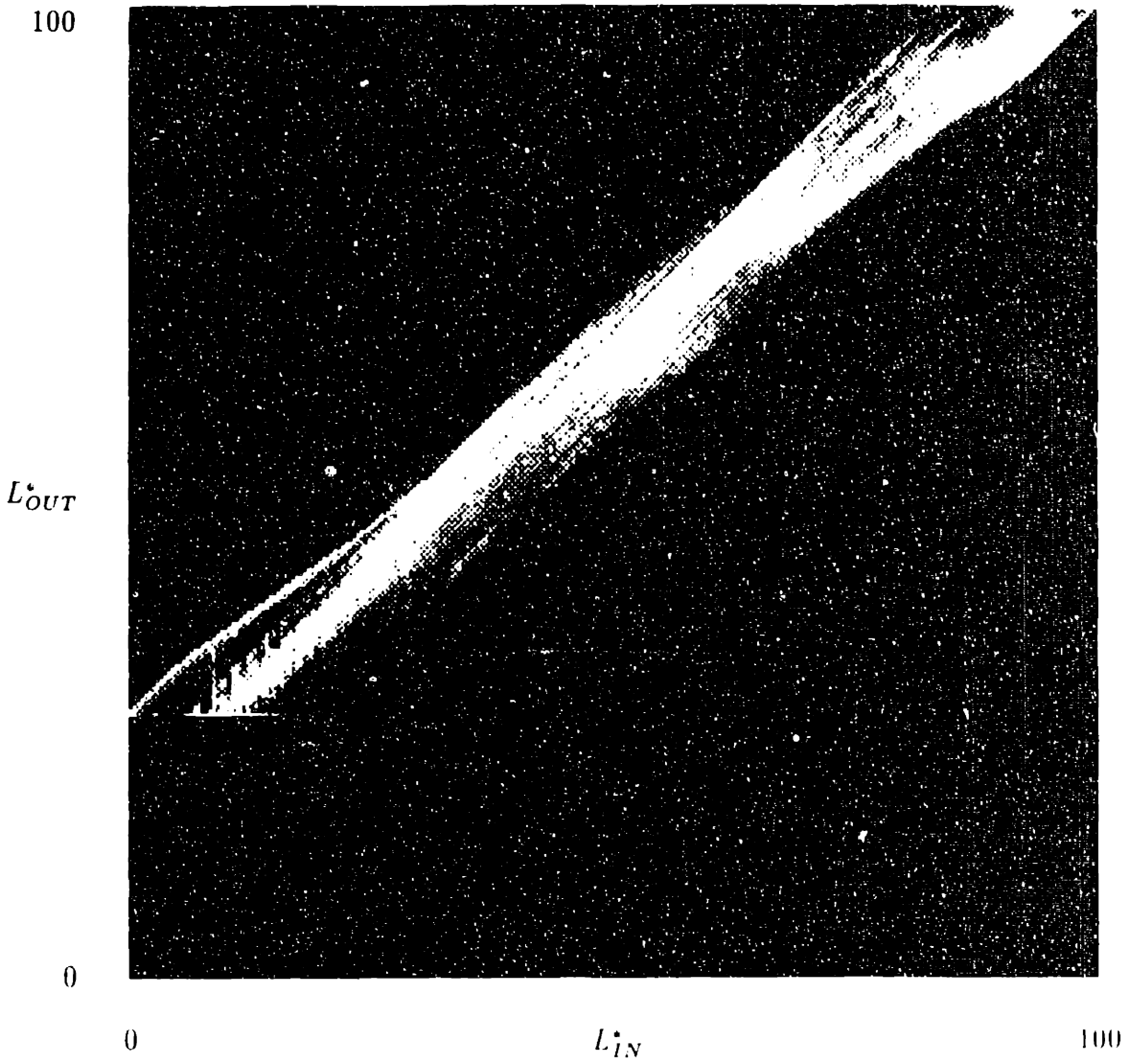


Figure 4-14: Input-output relationship of the wine image for LFLC.

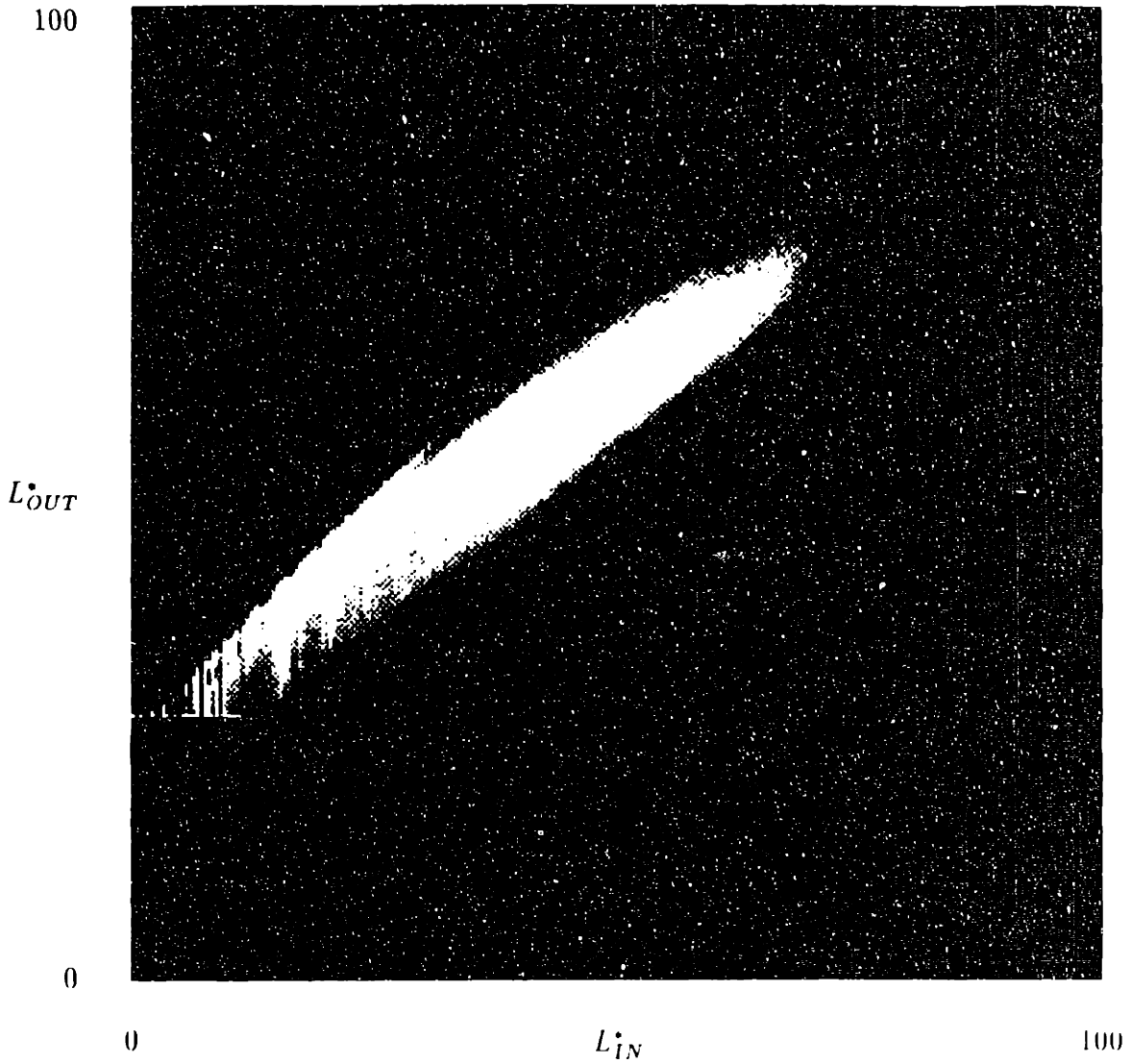


Figure 4-15: Input-output relationship of the rose image for LFLC.

Chapter 5

Three-dimensional Gamut Compression

5.1 Choosing the Compression Variable

Color gamut compression is inherently a three-dimensional problem because human color perception is three-dimensional. There is an infinite number of three-dimensional spaces that could be used to represent color and provide the domain for performing color gamut compression. As discussed in Chapter 1, the $L^*a^*b^*$ color space, which is the L^*hC^* color space in cylindrical coordinates, was chosen. The L^*hC^* representation is particularly useful in this chapter because L^*, h, and C^* are somewhat orthogonal with respect to human perception, that is, they relate to different perceptual attributes of color.

During the gamut compression process, out-of-gamut colors must be changed. The question is, which of the three numbers which represent the color should be changed. Sometimes, changing any one of the three values is sufficient. Since the lightness compression of Chapter 4 guarantees that all lightness values will be within the achromatic range of the device, changing L^* is never necessary. Likewise, since changing h only rotates the hue value around the gamut, changes in h are never necessary. Only if changes

in C^* are allowed can it be guaranteed that a pixel value can be changed to be in-gamut[1].

Although it is never necessary to change L^* or h, it may be advantageous. Since the color gamut is not circularly-symmetric, changing h can sometimes allow larger amounts of chroma. Likewise, for dark or light colors, changing L^* may also allow larger amounts of chroma. Caution must be exercised when changing the hue or lightness of a pixel. In general, for a constant ΔE, changes in hue are riskier than changes in lightness, and changes in lightness are riskier than changes in chroma. Because L^*, h, and C^* correspond to different perceptual attributes, the relative acceptability of various color changes may be different even though the perceptibility of the changes is equal. For example, if the color of a red apple is compressed by shifting the red towards purple or orange, the new color may be unacceptable. Changes in lightness or chroma that are perceptually equidistant may be preferred.

Later in this chapter, an adaptive method for gamut compression is developed. Making the gamut compression adaptive over the entire space adds quite a bit of complexity to the problem. In order to keep the problem manageable, only C^* will be allowed to change during the three-dimensional gamut compression. C^* is the only one of the three variables that can guarantee that all colors can be moved in-gamut, and fortunately, it is the least likely of the three variables to cause an objectionable error. Figure 5-1 illustrates the approach and should be compared to Figure 1-1.

5.2 Changing C*

There are many possible ways to reduce the chroma of image pixel values in order to gamut compress an image. In this section, two extreme approaches and the form of the compromise will be discussed.

[1]If the white point or the black point of the device gamut does not lie on the achromatic axis, then reducing C^* of a very light or very dark color may not be sufficient to bring that color in-gamut. This case is not discussed because it can be avoided by temporarily tilting the achromatic axis to pass through the white point and the black point.

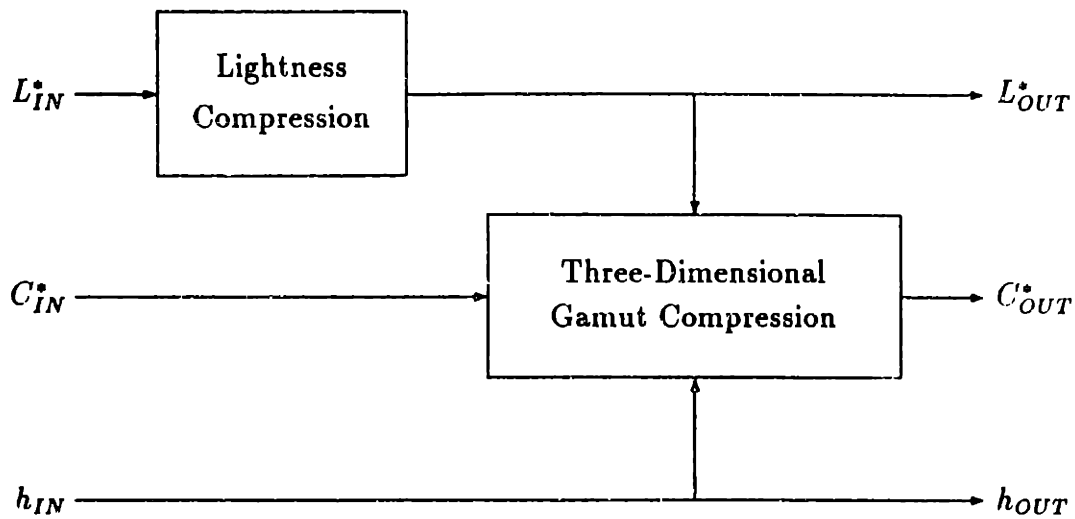


Figure 5-1: Simplified method for color gamut compression.

The first extreme method, which is conceptually similar to the MEGC method of Section 1.2.4, is to "chop off" any chroma values that are out-of-gamut. This approach, *chroma clipping*, leaves all colors that are originally in-gamut unchanged. To move colors which are out-of-gamut into the device gamut, the chroma value of each out-of-gamut pixel is reduced by the minimum amount needed to bring the pixel's color in-gamut:

$$C^*_{OUT} = C^*_{IN} \qquad \text{for} \quad C^*_{IN} \leq C^*_{device}(L^*, h) \tag{5.1}$$

$$C^*_{OUT} = C^*_{device}(L^*, h) \qquad \text{for} \quad C^*_{IN} > C^*_{device}(L^*, h) \tag{5.2}$$

C^*_{IN} is the chroma value of a pixel in the input image, C^*_{OUT} is the same pixel's chroma value after the gamut compression, and $C^*_{device}(L^*, h)$ is the maximum reproducible chroma of the particular output device for a given lightness and hue.

The second extreme approach, Buckley's method, discussed in Section 1.2.3 is to globally reduce all chroma values until the entire image is in-gamut. Thus, if there are any out-of-gamut colors, then all colors in the image will have their chroma value decreased by the same scale factor required to bring the most extreme color into the device gamut.

$$C^*_{OUT} = \beta \cdot C^*_{IN} \qquad \text{all} \quad C^*_{IN} \tag{5.3}$$

All image chroma values are scaled by the positive real constant $\beta \leq 1$. β is the largest scale factor which moves all image colors inside the device gamut.

As mentioned in Section 1.3, these two extreme approaches, chroma clipping and global chroma scaling, each have their own associated advantages and disadvantages. The chroma clipping method has the advantage that it results in the largest average chroma in the output image, which makes the image look colorful. The chroma clipping method has the disadvantage that it is a many-to-one mapping which transforms colors which are originally different to the same printable color. This many-to-one mapping tends to reduce the visibility of image detail. If an image contains a large object that has most of its colors in one region which is outside of the device gamut, then the object may suffer considerable loss in detail as a result of the many-to-one color mapping. An example of this can be found in [15].

The global chroma scaling has the opposite properties. It has the severe disadvantage that the average chroma of the output image is often reduced significantly. Images processed by the global chroma scaling technique usually look "washed out" because of their low average chroma. Chroma scaling has the advantage that it keeps separate colors separate throughout the mapping process. Thus, more image detail is preserved using the global chroma scaling method.

Given the two extremes, with diametrically opposing advantages and disadvantages, the obvious goal is to find the optimal balance point between the two, which maximizes the benefit and minimizes the degradation from each. This optimal method must keep the average chroma as high as possible while trying to avoid bunching a lot of colors together. Given this goal, it is not clear what method can be used to accomplish it. What must be found is a set of equations which provide a compromise between the chroma clipping of Equations 5.1 and 5.2, and the chroma scaling of Equation 5.3.

The two extreme approaches have an important difference. Chroma clipping gamut-compresses each pixel color independently from all others, whereas global chroma scaling operates on all pixel colors together. Since we are looking for a compromise between

these two methods, what is desired is compression that is *adaptive* over different regions of the gamut. This way similar colors are essentially processed together to avoid the many-to-one mapping problem, yet colors that are very different are essentially treated independently.

The reason why the global chroma scaling method almost never works well is that the solid that represents the image data in three-dimensional color space is usually shaped differently than the solid representing the device's color gamut. Global scaling shrinks entire image gamut with respect to the worst case, overcompressing all other colors and wasting a large portion of the output device's color gamut capabilities. Scaling by a constant factor will never make the shapes fit together better. What is needed is to "warp" the shape of the image gamut to fit the device gamut better. This way, the image is not compressed for regions of color that are in-gamut, but regions of the image that are out-of-gamut will be compressed to fit within the device gamut. This warping is precisely what adaptive processing can accomplish.

The most general adaptive compression function is a function of L^*, h, and C^*:

$$C^*_{OUT} = f(L^*, h, C^*_{IN}) \tag{5.4}$$

where C^*_{IN} is the chroma value of an arbitrary input color and C^*_{OUT} is the output chroma value that is guaranteed to be in-gamut. This chroma compression is adaptive over all three dimensions of the color space.

Figure 5-2 helps illustrate the sort of problem that often occurs with the two extreme chroma compression methods, and provides the motivation for adaptive processing. The points A, B, C, and D represent four pixel colors in an unprocessed image. The color A is the only color that is outside of the color gamut of the device. The chroma clipping method would change color A to be the same as color B, and leave the other colors unchanged. The global chroma scaling method would move color A to the original position of color B, but the chroma of all of the colors are also be reduced by the factor $\beta = \frac{C^*_B}{C^*_A}$. In this example, the color A has a high lightness value, which is near the white point of the device gamut. Color gamuts have a very small chroma range at high lightnesses

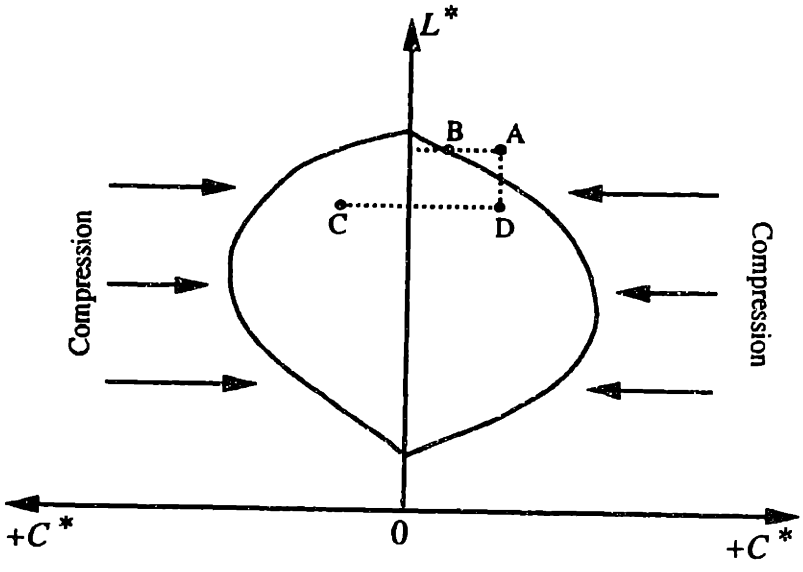


Figure 5-2: Chroma compression of image pixel colors.

(and at low lightnesses), so the scale factor β is very small. Unless the compression is allowed to vary as a function of lightness, color D, which is in-gamut, will be compressed severely. Likewise, unless the compression is allowed to vary as a function of hue, color C, which is also in-gamut will also be compressed severely. Consequently, it is important that the chroma scaling be adaptive for different lightnesses and hues.

To avoid the problems discussed above, the chroma compression is adaptive over all three dimensions of the color space. Being adaptive over different hues allows, for example, the greens in an image to be compressed more or less than the blues. Being adaptive over different lightnesses allows light blues, for example, to be compressed more or less than dark blues. The compression may even be a nonlinear function dependent on C^*_{IN}. This would provide the ability to compress colors that are far out-of-gamut proportionately more than colors that that are well within-gamut, thus keeping the average chroma of the image as high as possible.

Although the gamut compression is allowed to be adaptive over the entire color space, it cannot be totally unconstrained. Three intuitive constraints are assumed. The first assumption is that the gamut compression function should vary smoothly across the

color space in all directions. If neighboring regions of the color gamut were allowed to be processed radically differently, we might expect, as Evans would describe it, "internal inconsistencies" in the image. The second assumption is that the average chroma of an image should be maximized in order to make the image look as colorful as possible. The third assumption is that it is desirable to keep distinguishable colors separate after processing. Achieving the goals of the second and third assumptions requires a compromise, since separating out-of-gamut colors requires moving them away from the borders, towards the middle of the gamut.

These few assumptions really do not constrain $f(\cdot)$ of Equation 5.4 much at all. The chroma compression function can still assume a wide variety of mathematical forms. What is needed is to select a particular form for $f(\cdot)$ that can provide the basis for the adaptive chroma scaling. One reasonable approach is to define a family of transfer functions having exogenous parameters to control the smoothness of the gamut mapping and the relative tradeoff between clipping and scaling. Using this approach, the processing can be controlled by only two or three parameters. Reducing the number of control parameters makes experimentation easier, and is useful for reducing the complexity of gamut compression systems. If the gamut compression system is manually controlled, having fewer parameters may make the adjustments easier. If the gamut compression is automatic, then fewer control parameters need to be estimated from the image in order to process the image.

5.3 A Particular C* Transfer Function

As we have seen from the previous section, there is no clear choice of what function should be used for $f(\cdot)$. In this section the particular choice of $f(\cdot)$ that was implemented will be explained.

Since the transfer function described here must actually be implemented, the implementation issues are intertwined with the selection of the transfer function. All else being

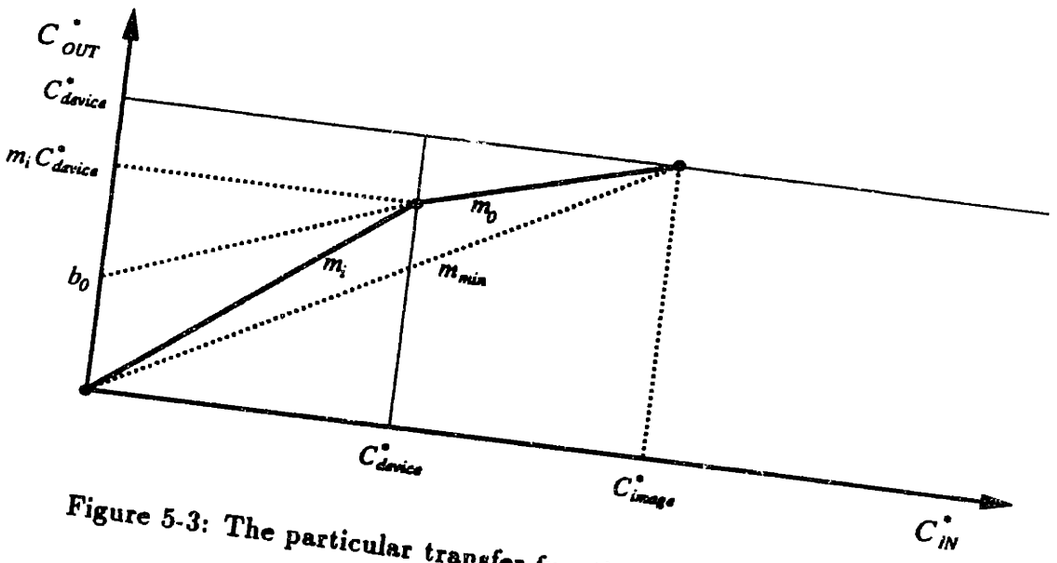


Figure 5-3: The particular transfer function used to compress chroma.

equal, transfer functions which require less computation are preferred. The function chosen is piecewise-linear:

$$C^*_{OUT} = m_i \cdot C^*_{IN} \quad \text{for} \quad C^*_{IN} \leq C^*_{device}(L^*, h) \tag{5.5}$$

$$C^*_{OUT} = m_o \cdot C^*_{IN} + b_o \quad \text{for} \quad C^*_{IN} > C^*_{device}(L^*, h) \tag{5.6}$$

The transfer function is comprised of a linear transformation for colors inside of the device gamut and an affine transformation for colors outside of the device gamut. The transfer function is shown in Figure 5-3.

Figure 5-3 refers to a particular lightness-hue region of the gamut. The value of the input chroma is plotted along the abscissa and the corresponding output chroma is plotted along the ordinate. The parameters m_i, m_o, and b_o, along with the maximum measured chroma of the output device, C^*_{device}, specify the exact form of the equation. Note that m_i, m_o, b_o, and C^*_{device} are all functions of L^* and h, and thus vary the gamut mapping over different lightness-hue regions of the gamut. In order to have the overall gamut mapping be smooth, the variables m_i, m_o, and b_o must vary smoothly as functions of L^* and h.

The two line-segments that make up the transfer function are specified by three points. One of the three points is fixed at the origin so that input chroma values of zero are

mapped to output chroma values of zero. In other words, adding a constant to in-gamut chroma values is not allowed. The other endpoint is fixed such that the maximum image chroma value in a particular L^*, h region of the gamut, C^*_{image}, is mapped to the gamut boundary of the device. The middle of the three points lies on the line $C^*_{IN} = C^*_{device}$ and has a height $m_i C^*_{device}$, where $0 < m_i C^*_{device} \leq C^*_{device}$. The value of $m_i C^*_{device}$ controls whether clipping or scaling is emphasized. When $m_i = 1$, then the transfer function is exactly like the chroma clipping of Equations 5.1 and 5.2. When $m_i = m_{min}$,

$$m_{min} = \frac{C^*_{device}}{C^*_{image}} \tag{5.7}$$

such that the two line segments of the transfer function form a straight line, the transfer function is identical to the linear scaling of Equation 5.3 except that the transfer function is adaptive and Equation 5.3 is global. When $m_i < m_{min}$ then chroma values are reduced and separated even more than the chroma scaling technique. Since chroma scaling already has the problem that it tends to reduce and separate chroma too much, $m_i < m_{min}$ is an unlikely candidate to be a good parameter choice and will not be allowed.

Thus, as m_i varies between 1 and m_{min}, the transfer function behavior varies between clipping and scaling. One of the endpoints of the transfer function is fixed at the origin and the other is fixed by the gamut of the output device and the gamut of the image. This leaves only one free variable, the position of middle point of the chroma transfer function. Consequently, it is not necessary to store m_i, m_o, and y_o to apply Equations 5.5 and 5.6. Only two parameters are needed for each lightness-hue region of the gamut. In the implementation, m_i, m_o, and y_o are each stored to avoid repetitive calculation of the redundant variable.

Since the gamut warping function depends on the particular colors in the image, the first step in calculating the parameters m_i, m_o, and b_o is to tabulate the multidimensional histogram of image colors. This histogram is a function of the form $N(L^*, h, C^*)$, where $N(\cdot)$ is the number of pixels of color (L^*, h, C^*). Histogram bins are defined over a certain volume which determines the number of bins needed. The larger the volume of each individual bin, the fewer the bins that are necessary. Since we eventually want

to average many adjacent histogram values to produce a smoothly changing transfer function, a large number of histogram bins is desirable. Since the number of bins can quickly become very large due to the histogram's cubic form, a simplified version of histogram is used in the implementation. The three-dimensional histogram is reduced to three, two-dimensional histograms which are functions of lightness and hue:

$$N_{in}(L^*, h)$$

$$N_{out}(L^*, h)$$

$$C^*_{max}(L^*, h).$$

The three variables are non-negative integer quantities, representing the number of in-gamut pixels, the number of out-of-gamut pixels, and the maximum chroma value for each small lightness-hue bin of the color gamut. In the implementation, 187 lightness bins and 360 hue bins were used. This total of 67,320 bins is probably much more than necessary, but was chosen to avoid problems in achieving a smooth and accurate warping function due to sampling the color space too coarsely.

The next step, having tabulated the characteristics of the distribution of color within the image, is to produce a smoothed version of each of the three, two-dimensional histogram tables. The smoothed version of the histograms will be used to calculate m_i, m_o, and b_o. It is important to note that the smoothing is performed on the raw histogram data rather than the final parameters m_i, m_o, and b_o. If the smoothing were done on m_i, m_o, and b_o then all information about the relative importance of each L^*, h bin is lost. The sort of problem that could occur if m_i, m_o, and b_o are smoothed is that one m_i, m_o, b_o triplet derived from only a few pixels could be averaged (weighted equally) with an adjacent m_i, m_o, b_o triplet which was derived from thousands of pixels. By averaging the raw data, the weight or importance given to each bin is proportional to the number of pixels that it represents.

The next issue related to smoothing is how exactly to average nearby histogram bins. Should all the bins in the average be given equal weight? Over how large a lightness-hue range should the histogram bin averaging cover? Since it is practically impossible

to predict how large the area of averaging should be, this is left as an input to the computer program that performs the chroma compression. At first glance it might seem reasonable to give all of the histogram bins in the averaging region equal weight, but this can cause a problem. The problem is that discontinuities are more likely when using equal weighting because each histogram bin must either be included or completely ignored. Equal weighting is the same as using a rectangular window function. Whenever the rectangular window crosses over a histogram bin that has a very different nature compared to its neighbors, the full impact of that bin is added immediately. Histogram bins should be able to be included in the computation gradually. Instead, if a triangular (Bartlett) window function is used, then discontinuities in the histogram will become ramp changes in the smoothed histogram which would have appeared as step changes when the rectangular window was used.

A triangular window is a one-dimensional function. A two-dimensional window is needed because the histograms are functions of two variables, L^* and h. A separable two-dimensional window can be formed as the product of two, one-dimensional windows. Figure 5-4 shows the two one-dimensional windows.

The variables L_{radius} and h_{radius} set the width of the windows. There are $2L_{radius}+1$ and $2h_{radius}+1$ nonzero points in each one-dimensional window. The maximum value of both windows is set to be 1. The two-dimensional window is given by:

$$\begin{aligned} w(L^*,h) &= w_L(L^*)\cdot w_h(h) \\ &= \begin{cases} \dfrac{(L_{radius}+1-|L^*|)(h_{radius}+1-|h|)}{(L_{radius}+1)(h_{radius}+1)} & |L^*|\le L_{radius},\ |h|\le h_{radius} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{5.8}$$

Applying the window to each of the three, two-dimensional histograms gives:

$$S_{in}(L^*,h) = \sum_{i=-L_{radius}}^{L_{radius}} \sum_{j=-h_{radius}}^{h_{radius}} N_{in}(L^*+i,h+j)w(i,j) \tag{5.9}$$

$$S_{out}(L^*,h) = \sum_{i=-L_{radius}}^{L_{radius}} \sum_{j=-h_{radius}}^{h_{radius}} N_{out}(L^*+i,h+j)w(i,j) \tag{5.10}$$

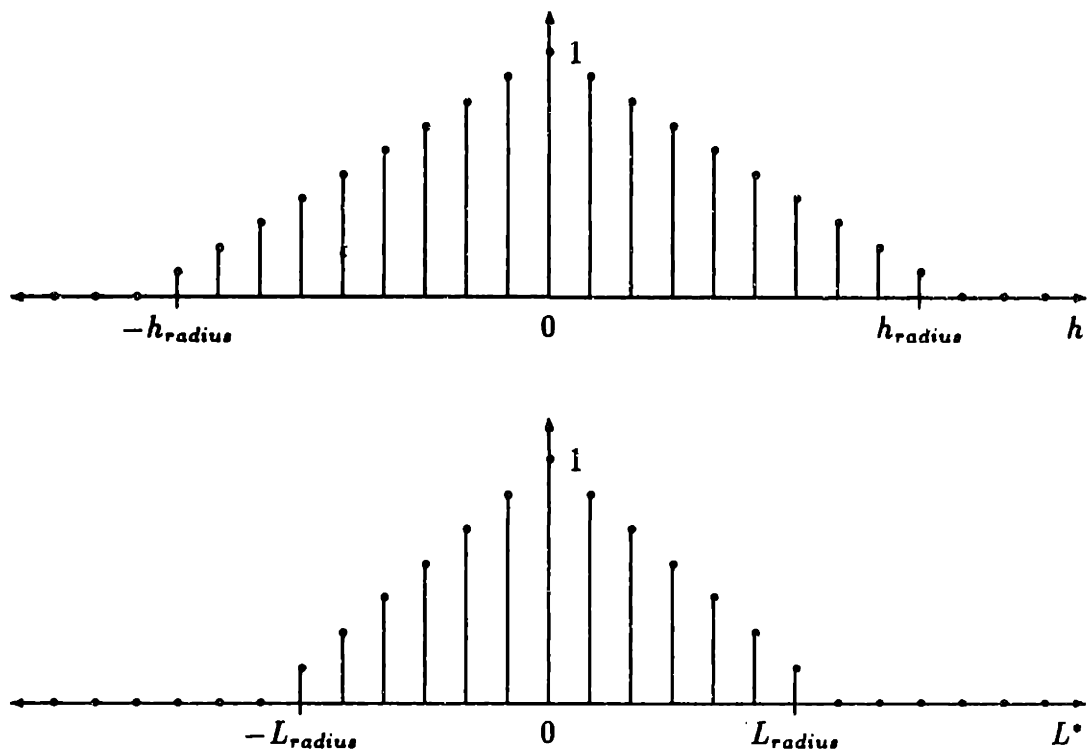


Figure 5-4: Windows used to smooth the histograms.

$$C^*_{image}(L^*,h) = \sum_{i=-L_{radius}}^{L_{radius}} \sum_{j=-h_{radius}}^{h_{radius}} C^*_{max}(L^*+i,h+j)w(i,j) \qquad (5.11)$$

where S_{in}, S_{out}, and C^*_{image} are the smoothed versions of N_{in}, N_{out}, and C^*_{max} respectively. When the smoothing is actually computed, the window will cross over the borders of the histogram array. This is not really a problem if the histogram entries are defined to be zero outside the borders of the histogram table. When the window falls beyond the lightness boundary, the zero values of these non-existent histogram bins will not effect the smoothing. Since hue is an angle value, whenever the window passes beyond the hue border of the table, the window should be wrapped around to the opposite end of the table. So, for example, if histogram hue bins are spaced one degree apart, $h_{radius} = 2°$, and the histogram is windowed about $h = 0$, then hue values of -358, -359, 0, 1, 2 degrees would be used.

Because the window is an even function, Equations 5.9, 5.10, and 5.11 are equivalent to convolution in space:

$$S_{in}(L^*,h) = N_{in}(L^*,h) * w(L^*,h) \qquad (5.12)$$

$$S_{out}(L^*,h) = N_{out}(L^*,h) * w(L^*,h) \qquad (5.13)$$

$$C^*_{image}(L^*,h) = C^*_{max}(L^*,h) * w(L^*,h) \qquad (5.14)$$

where $*$ denotes convolution. Note that the smoothing operation could be performed in the frequency domain by multiplying the discrete Fourier transform of each of the three histogram tables by the discrete Fourier transform of the window. If care is taken, the wrap around requirement of windowing the hue can be handled implicitly using the circular convolution performed by the DFT multiplication.

From the smoothed histogram data, the parameters m_i, m_o, and b_o are calculated by:

$$\left.\begin{aligned} m_i &= \frac{S_{in}}{S_{in}+S_{out}} \\ m_o &= \frac{C^*_{device}(1-m_i)}{C^*_{image}-C^*_{device}} \\ b_o &= C^*_{device}(m_i-m_o) \end{aligned}\right\} \quad \text{for} \quad m_i \geq m_{min} \qquad (5.15)$$

$$\left.\begin{aligned} m_i &= m_{min} \\ m_o &= m_{min} \\ b_o &= 0 \end{aligned}\right\} \quad \text{for} \quad \frac{S_{in}}{S_{in} + S_{out}} < m_{min} \tag{5.16}$$

Since the gamut compression is adaptive and S_{in}, S_{out}, and C^*_{image} are functions of (L^*, h); m_i, m_o, and b_o are also functions of (L^*, h). For implementation purposes, there is no reason to store S_{in}, S_{out}, and C^*_{image} in an intermediate table. Once these values are computed for a particular lightness-hue bin, the corresponding m_i, m_o, and b_o can be computed immediately and stored in a lightness-hue table.

The final step is to apply the compression function to the image chroma values. This involves the application of Equations 5.5 and 5.6. Each pixel is tested whether it is in-gamut or out-of-gamut in order to determine whether to apply Equation 5.5 or Equation 5.6.

It is important to note that Equation 5.15 is where the tradeoff between clipping and scaling is made. If all pixels are in-gamut, $S_{out} = 0$, which gives $m_i = 1$, setting the output chroma equal to the input chroma. As the number of out-of-gamut pixels is increased, the slope of m_i is is decreased to make space for the out-of-gamut colors to be mapped into the gamut. If all the pixels are out-of-gamut, then the slope of m_i and m_o are limited by m_{min}.

Equations 5.15 and 5.16 are somewhat arbitrary. There are many other, possibly better, ways of specifying and implementing a chroma transfer function. Equations which emphasize clipping would probably be an improvement, and are a topic for future research.

Chapter 6

Results, Conclusions, and Suggestions

6.1 Experimental Results

Appendix B contains eight pages of color images. The following evaluations are based on my own subjective judgements with input from both people experienced and inexperienced in graphic arts and image processing. The evaluation of the images is described in some detail because copies of this document will not have the original pictures included. Also, a problem to be wary of is that these color pictures fade over time. The fading can be noticeable after only a few months, depending on the amount of light that the picture has been exposed to. As the pictures fade, subtle differences in processing become less obvious. In an attempt to make the color images for this thesis more permanent, color photocopies were produced. Because the color gamut of the copier was sufficiently different from that of the Paintjet printer, the results were obscured. Table 6.1 summarizes the processing applied to each of the eight images in Appendix B. Table 6.2 lists the exact processing parameters, discussed in Chapter 4, used for the low frequency lightness compression of both images. Table 6.3 shows how the color space is compressed during the adaptive chroma scaling of Chapter 5. The percentages in Table 6.3 indicate the

Table 6.1: Processing Summary of Images in Appendix B

Figure	Lightness Processing	Chroma Processing
1a	affine	none
1b	affine	clipping
2	none	clipping
3	affine	clipping
4	LFLC	clipping
5	LFLC	ACS
6	affine	none
7	LFLC	clipping
8	LFLC	ACS

Table 6.2: Parameters of Low Frequency Lightness Compression (LFLC)

Parameter	Wine Image	Rose Image
τ	30	10
α_h	1	1
estimated α_l	.70	.69
actual α_l	.70	.69
estimated d	29	21
actual d	29	29
$L^*_{OUT} < K_{dev}$	1.6%	2.1%
$L^*_{OUT} > 100$	2.2%	0.002%

Table 6.3: Characteristics of Adaptive Chroma Scaling (ACS) over the Color Gamut

$L_{rad} = h_{rad} = 6$			
Case	Wine	Rose	comments
$m_i = 1$	52%	35%	all nearby colors are in-gamut
$1 > m_i > m_{min}$	8%	14%	piecewise linear compression
$m_i = m_{min}$	25%	29%	linear scaling; many out-of-gamut pixels
unspecified	15%	22%	no pixels in this color range

amount of the color space (the number of histogram bins) associated with each case of the processing.

The first color image, Figure B-1, shows the importance of eliminating out-of-gamut colors before halftoning and printing. In Figure B-1a, no form of chroma compression was applied. The out-of-gamut colors cause quite undesirable results in the reproduction. The blue hat has been printed as magenta, cyan blobs have appeared in the eyes, and the bow tie is almost purely the red primary. In Figure B-1b, chroma clipping was performed on the image, eliminating the gross errors due to out-of-gamut colors.

The images in Figures B-2, B-3, B-4, and B-5 have no out-of-gamut colors. To more carefully observe the changes in lightness, a gray scale has been appended to the bottom of the original image. The gray scale varies linearly in lightness from the value of the blackest color in the image to the value of the whitest color in the image. Figures B-2, B-3, and B-4 illustrate the importance of properly adjusting the lightness and Figure B-5 is for comparing adaptive chroma scaling to chroma clipping.

The image in Figure B-2 clearly shows the need for matching the lightness range of the image to the lightness range of the output device. The lightness range of the image is greater than the lightness range of the printer, so many colors are clipped to black. The left side of the gray scale shows the wide range of lightnesses that have been mapped to black.

The lightness range of the image in Figure B-3 has been reduced using the affine transformation to map the image's black to the printer's black and the image's white to

paper white. Note that the left edge of the gray scale is the only part printed as pure black and the right edge of the gray scale is the only part that is pure white. Although the affine-compressed image looks much better than the unprocessed image, it also has low contrast.

The image in Figure B-4 illustrates the low frequency lightness compression method. It clearly is an improvement over the affine lightness compression in Figure B-3, which looks dull and flat by comparison. The LFLC image has much higher apparent contrast and sharpness. The blacks look blacker and the image appears to have much more depth. Note the slight taper in the upper right corner of the gray scale. This is caused by the filtering.

The last image in this group, Figure B-5, shows the effect of adaptive chroma scaling. The differences between this and the previous image are subtle, and the improved detail is partially obscured by the low resolution of the printer. Some improved separation of colors can be seen. For example, the red grapes appear to have more shape than before.

The last three images, Figures B-6, B-7, and B-8, are cases in which judgments of the images varied the most. The image in Figure B-6 is compressed in lightness by the affine transformation and has not been chromaclipped. In this image, 38% of the pixel values are out-of-gamut. There is considerable loss of detail because many pixels are reproduced as the red primary of the printer. This also results in a maximally saturated image. Some people preferred this image because it was extremely saturated even though the detail was lacking. This image was certainly not the most accurate, but some people found it pleasing.

The image in Figure B-7 was processed by the LFLC technique and had chroma clipped. The LFLC improved the image contrast and the chroma clipping stopped the halftoning algorithm from selecting the red primary for most of the rose. Some people preferred this image over the previous because of its improvement in detail.

The last image, presented in Figure B-8, was processed by the LFLC method and adaptive chroma scaling. This image has the best detail of the three rose images but

also has the lowest overall chroma. Most people found this image to be too desaturated and did not prefer it. It is interesting to note that when the color copies of the rose image were judged, Figure B-8 was preferred. The reason behind this is that the color gamut of the copier is much more restricted in the reds than the Paintjet. The copies of Figures B-6 and B-7 lost their advantage of high chroma but still lacked in detail. The detail of the image processed by adaptive chroma scaling was still clear in the color copy.

6.2 Conclusions and Suggestions

For all images tested, the low frequency lightness compression method presented in Chapter 4 was clearly superior to the affine transformation. Low frequency lightness compression always resulted in an image with equivalent or better appearance than the affine method. The methods for estimating the parameters of the low frequency lightness compression generally worked well. The only parameter which was not explicitly calculated, τ, turned out not to be particularly crucial. Often, a 50% change in the value of τ had an almost unnoticeable affect on the final image.

One possible extension of the LFLC method would be to allow image enhancement by removing the constraint $\alpha_h = 1$. If $\alpha_h > 1$ then the image is sharpened. This capability would require no extra computation and could open up many image enhancement possibilities.

There are methods which are simpler than LFLC and better than the affine transformation, such as various forms of static nonlinear transformations[1] that might be able to produce good results also [31] [19]. The low frequency lightness compression method has the advantage that it is easier to control than these fixed input-output operations. In systems designed for novice users or for automatic systems, having fewer parameters which are easy to select is an advantage. In their present form, Equations 4.14 and 4.15, which estimate values of α_l and d, are not accurate enough to control a completely au-

[1] Also called tone reproduction curves or Jones curves in the graphic arts.

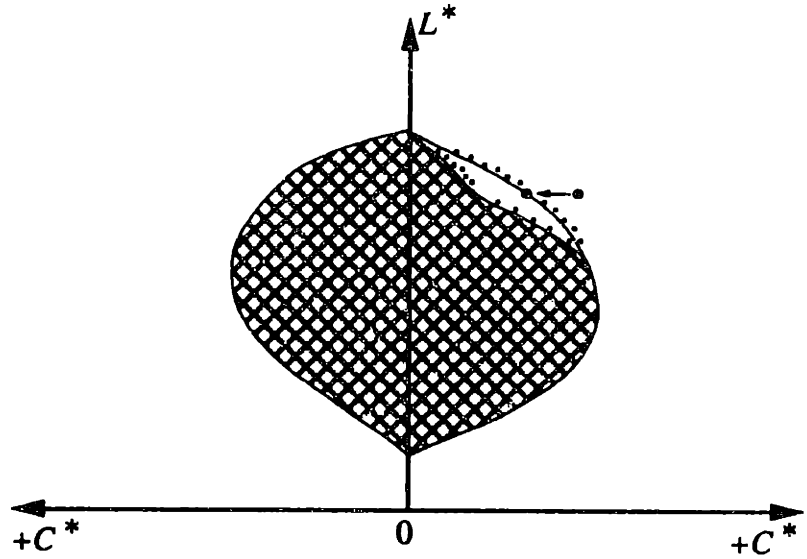


Figure 6-1: Severe reduction in chroma due to one far out-of-gamut pixel.

tomatic system. One possibility for future work would be to improve the methods for estimating the control parameters.

The adaptive chroma scaling method presented in Chapter 5 worked well on some images and did not work as well on other images. In all cases it improved image detail as predicted, but it sometimes over-compressed the color gamut. The lesson learned is that small amounts of adaptive chroma compression improve the image, but moderate or large amounts of chroma compression degrade the image because the average chroma becomes too low. One possible improvement of this particular implementation would be to change the form of Equations 5.5 and 5.6, which define the particular C^* transformation used, to emphasize clipping more and the distinction between colors less. The implementation could also probably be improved by allowing single extraneous colors to be ignored during the calculation of the transfer function. The problem that could occur is that if a region contains many colors which are slightly out of gamut, then the transfer function is only based on the peak image chroma in that region of color. One outlying color point could cause the entire region of the gamut to be desaturated. Figure 6-1 illustrates this potential problem.

Another problem occurred when converting from rectangular to polar coordinates to calculate h and C^* from a^* and b^*. Since a^* and b^* have only 256 possible values each, only a discrete number of hue angles is possible. The fact that many (a^*, b^*) values are converted to a few hue angles, such as 45°, and other hue angles are rare, such as 44°, causes the color histogram to be very uneven. The effects of these spikes can be reduced by increasing L_{radius} and h_{radius}, the dimensions of the window used to smooth the histogram (Equation 5.8). As a result, in the current implementation, L_{radius} and h_{radius} cannot be reduced arbitrarily without noticeable discontinuities in the gamut mapping.

One last suggestion for future research is to allow L^* and h to change during the three-dimensional gamut mapping. This is the general approach discussed in Chapter 1 and illustrated in Figure 1-1. L^* and h were held fixed in Chapter 5 to make the adaptive compression more manageable.

In summary, the low frequency lightness compression method worked very well. The three-dimensional adaptive chroma scaling method demonstrated the ability to improve images but the particular form of the transfer function should be improved so that a wider range of images can be processed without overcompression.

Appendix A

Printing L*a*b* Images

The color images processed in this thesis are stored as an array of $L^*a^*b^*$ values. L^*, a^*, and b^* are each represented by one byte, giving a total of $16,777,216$ different colors. The printer used is capable of producing only eight different colors of dots. To print colors different from the eight primaries of the printer, it is necessary to halftone, that is, use combinations of the eight possible color dots placed side-by-side to produce the desired color. There are two steps in the printing process. First, a correction is made for nonlinearities in lightness. Second, error-diffusion is used to halftone the $L^*a^*b^*$ image so that is suitable for printing.

The first step, correcting for nonlinearities in lightness, is necessary because of the characteristics of the printer. The printer's dots are round rather than square, and overlap when placed next to each other. Consequently, the area covered by dots increases rapidly with the number of dots added when there are few dots, and the area covered does not increase much when there are many dots. Figure A-1 shows the resulting lightness as a function of the percentage of black dots.

To measure the points graphed in Figure A-1, various percentages of black dots are printed. Although it is not always the case, there is a one-to-one mapping of lightness to dot percentage. If the function is inverted, we can determine the percentage of dots needed to print a desired lightness. In general, what is desired is to apply the inverse of

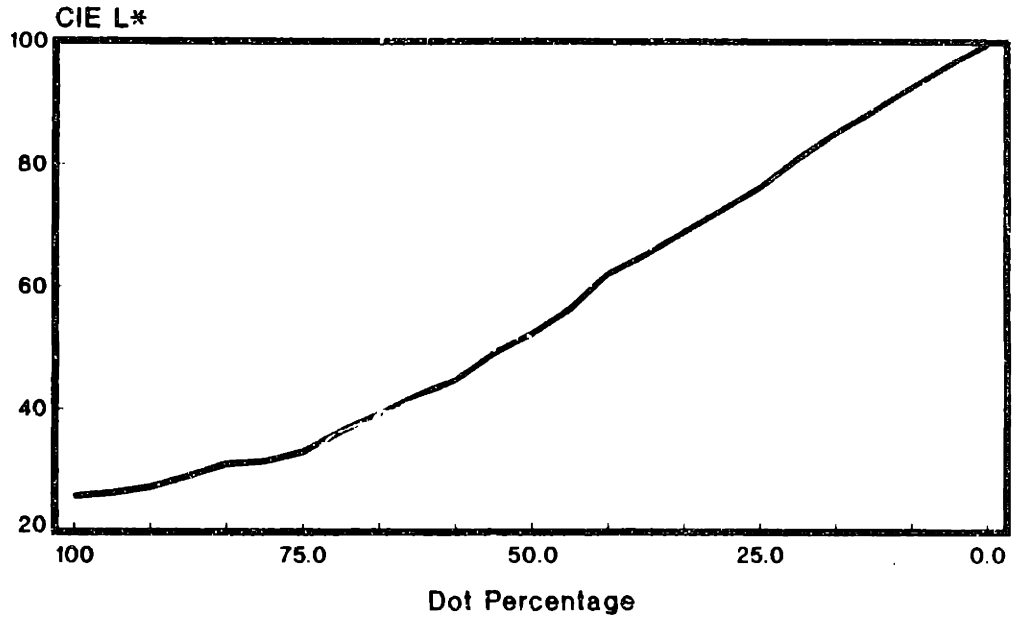


Figure A-1: Lightness printed as a function of dot percentage.

the printing function to the data. The printing system has inputs, and produces print samples which can be measured as $L^*a^*b^*$ values. To print a particular $L^*a^*b^*$ value, the correct inputs to the printing system can be found from the inverse of the printing function. Very sophisticated and accurate methods, such as fitting polynomials, are possible for calculating the proper inputs to produce a specified color[33]. By applying the inverse of the function graphed in Figure A-1, the image is corrected for the printer's nonlinearities in lightness before the halftoning.

The second step in the printing process is to halftone the image. The standard Floyd and Steinberg error diffusion technique [32] is used, extended to three-dimensions for color printing. The Floyd and Steinberg method method was originally intended for printing black and white images. The value of each input pixel is a number between zero and one. The two possible output values are zero for white and one for black. Each pixel is processed in turn, left to right and top to bottom. To determine the output value of the current pixel, it is compared to the threshold, which is $\frac{1}{2}$. If the pixel is greater than the threshold, then black is printed, and if the pixel value is less than the threshold, then

white is printed. The error is defined as the difference between the input value and the printed value. The error is divided up and distributed to neighboring pixels which have not been processed. The exact distribution and weights are shown in Figure A-2. The

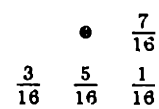


Figure A-2: The Floyd and Steinberg error diffusion weights.

solid dot in the figure represents the current pixel, and the fractions show the amount of the error distributed to each neighbor. The error weights sum to one so that the error is not increased or decreased as it propagates. This procedure is repeated until all pixels have been processed.

Conceptually, the Floyd and Steinberg error diffusion method extended to color printing is the same as the gray scale version. Instead of error-diffusing scalars which represent gray levels, three-dimensional vectors are processed to represent colors. Instead of two choices for each output pixel, there are eight, one corresponding to each primary dot color. Instead of a threshold, the ΔE_{ab} metric of Equation 2.19 is used to select the closest of the eight possible output colors. The error to be distributed to the neighbors is a vector rather than a scalar.

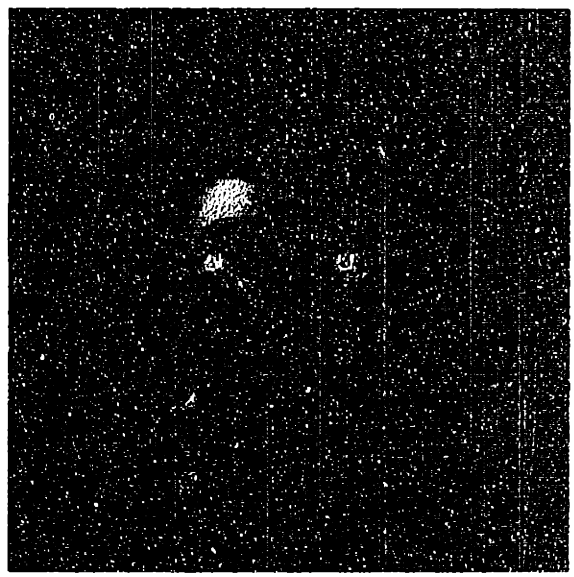
The extended, three-dimensional form of the error diffusion is more prone to problems associated with pixel values beyond the range of the primaries, compared to the one-dimensional version. This is why color gamut compression is important for printing color images. If the pixels are out-of-gamut, the error grows without bound and inappropriate output colors are selected. The image in Figure B-1 shows how the out-of-gamut colors cause gross printing errors. When actually implementing these algorithms, LUTs are used to minimize the number of calculations.

There are many possible improvements for the above printing process. For example, adding a serpentine raster or randomizing the weights in some fashion could reduce the patterning produced by the Floyd and Steinberg method. Another improvement would

be to use a more sophisticated pre-correction for nonlinearities. Fortunately, for the Paintjet printer, the error produced by this printing method is relatively small.

Appendix B

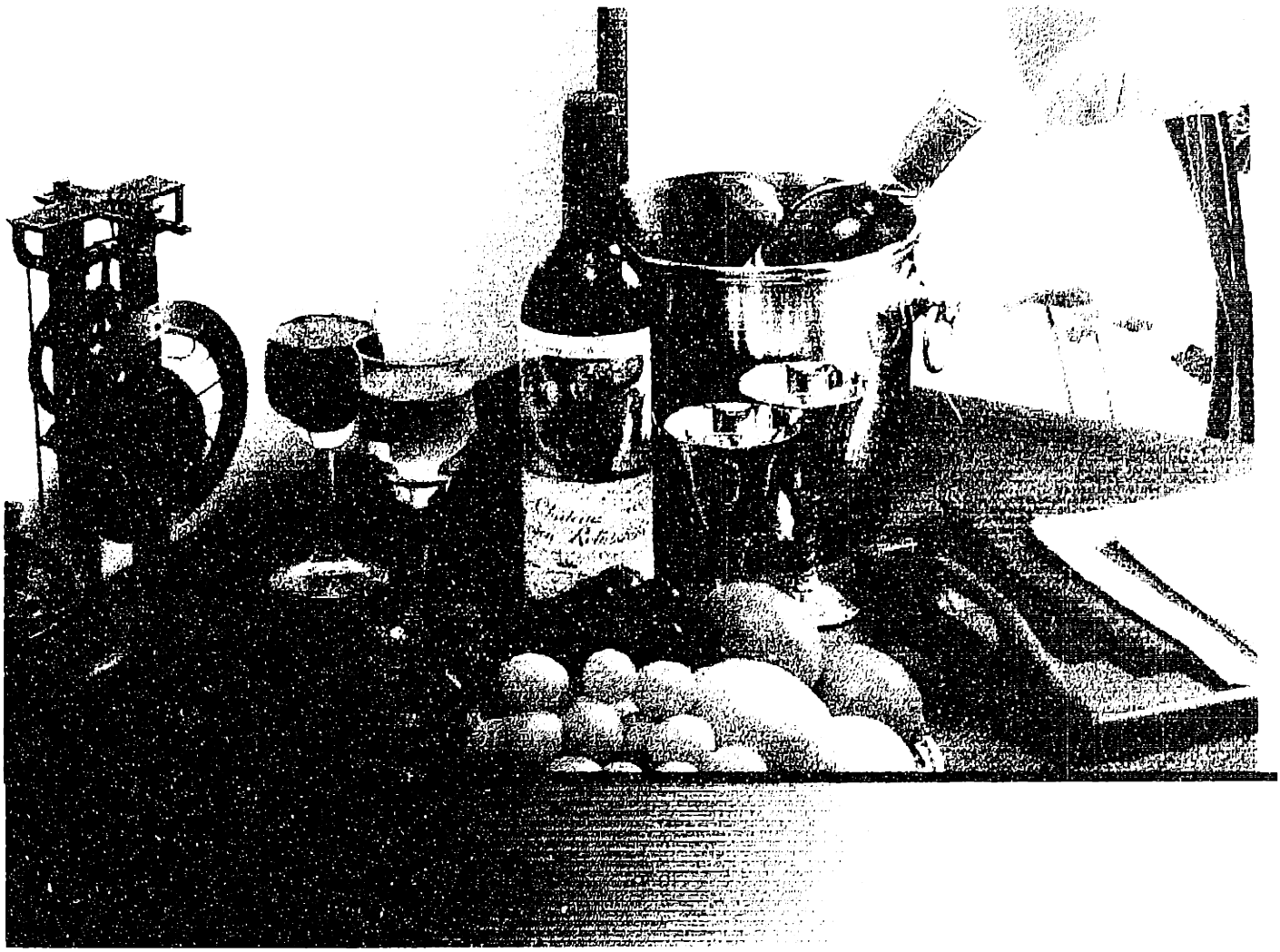
Color Images

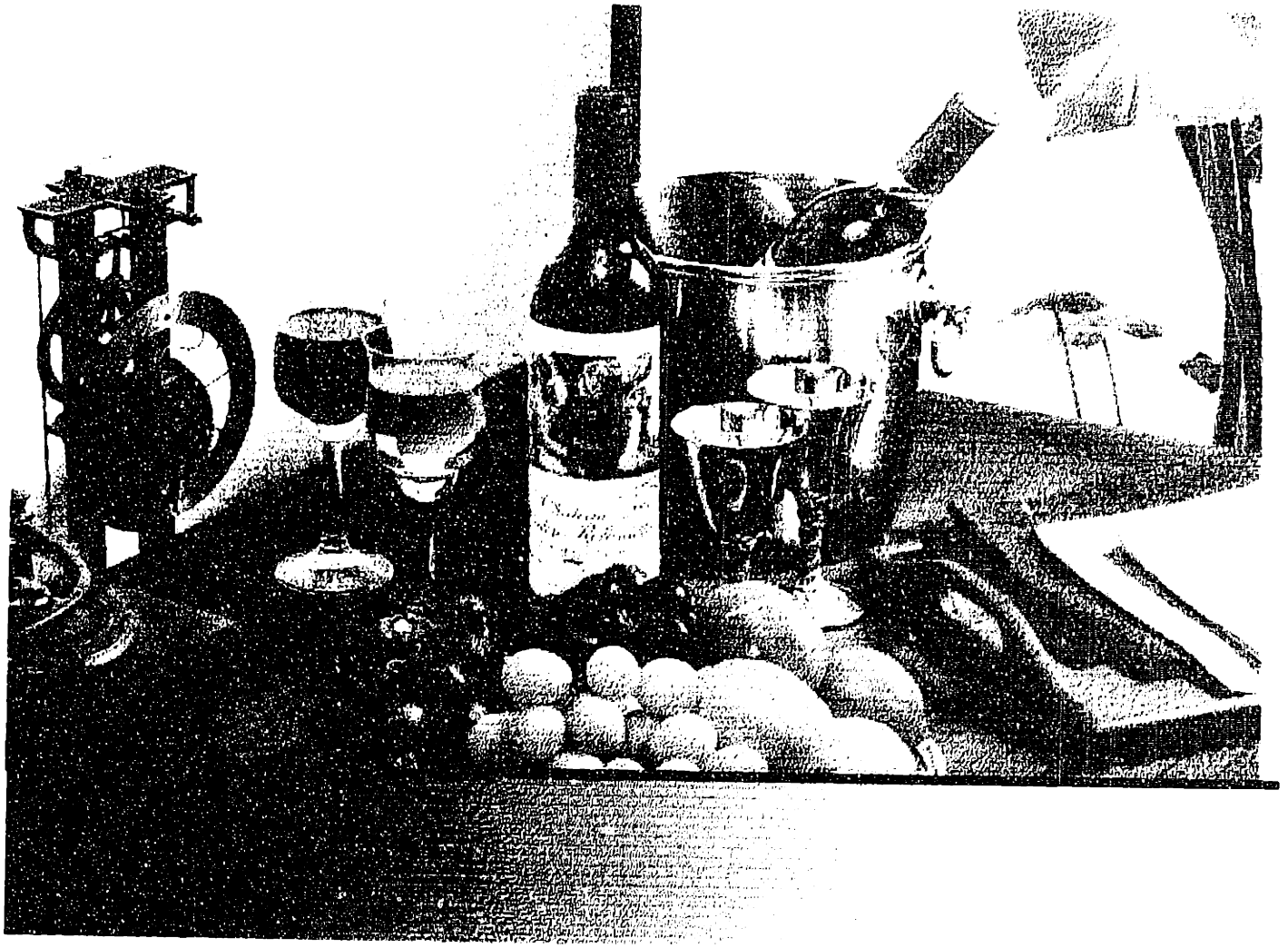


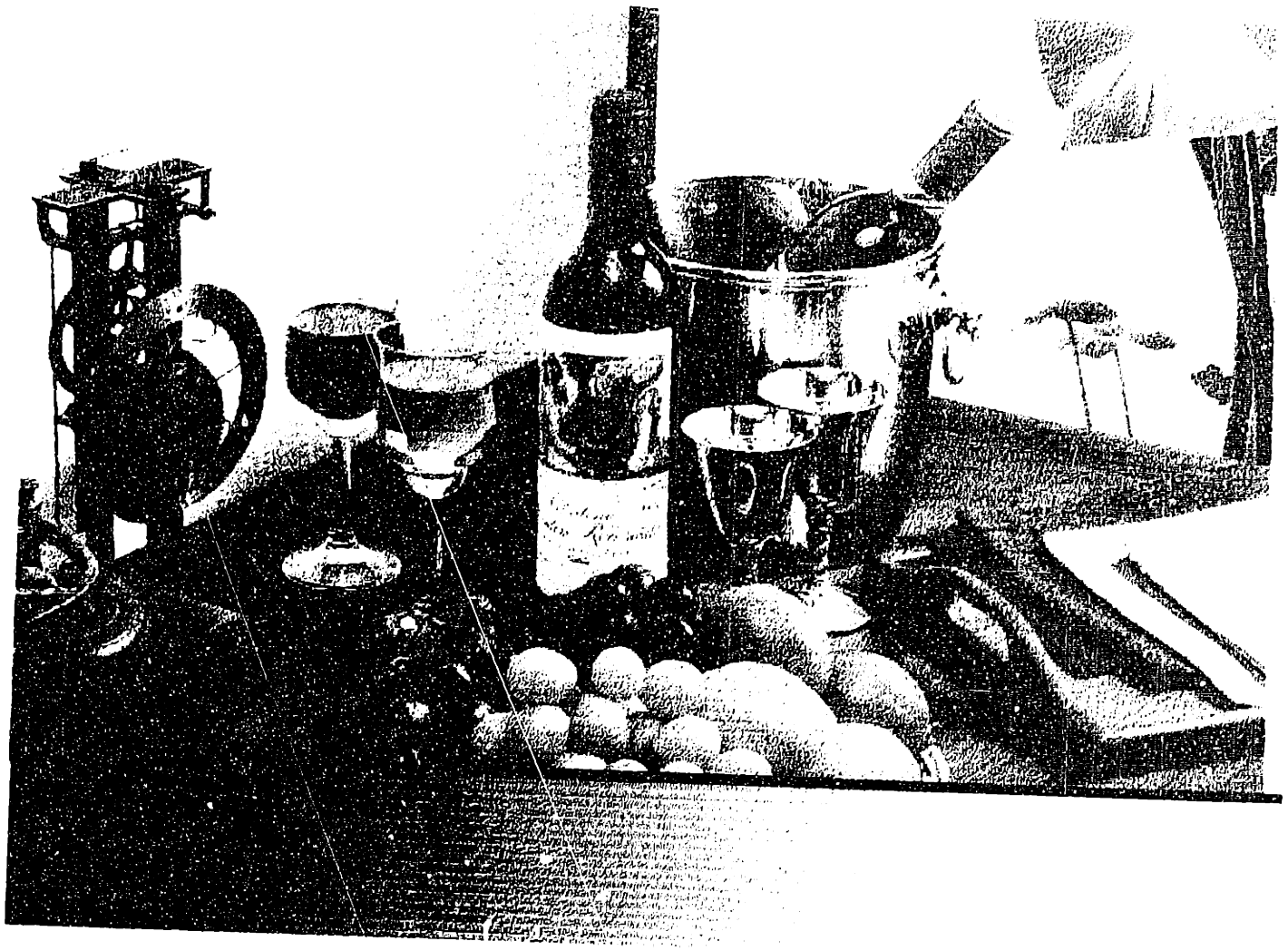
(a)

(b)

FIGURE B-1







Bibliography

[1] R. Allison, Y. Gur, F. X. D. O'Donnell, "Color Reproduction Using Error Diffusion," in *SPSE's 40th Annual Conference Summaries*, pp. 82–85, May 17–22, 1987.

[2] C. J. Bartleson, E. J. Breneman, "Brightness Perception in Complex Fields," in *Journal of the Optical Society of America*, vol. 57, pp. 953–957, July 1967.

[3] F. W. Billmeyer, M. Saltzman, *Principles of Color Technology.* Second Edition. New York: John Wiley & Sons, 1981.

[4] J. F. Boulter, *Use of Spatial Filtering to Match Wide Dynamic Range Grayscale Imagery to a Lower Resolution Display.* Research and Development Branch, Department of National Defence, Canada, file: 3621J-001, January 1977.

[5] E. J. Breneman, "Corresponding chromaticities for different states of adaptation to complex visual fields," in *Journal of the Optical Society of America (A)*, vol. 4, pp. 1115–1129, June 1987.

[6] G. Buchsbaun, A. Gottschalk, "Trichromacy Opponent Color Coding and Color Transmission in the Retina," in *Proceedings of the Royal Society of London (B)*, vol. 220, pp. 89–113, 1983.

[7] R. R. Buckley Jr. , "Reproducing Pictures with Non-reproducible Colors," *S.M. Thesis*, Massachusetts Institute of Technology, February, 1978.

[8] CIE, International Commission on Illumination, *Proceedings of the Eighth Session*, Cambridge, England. Paris: Bureau Central de la CIE, 1931.

[9] T. N. Cornsweet, *Visual Perception.* Orlando, FL: Academic Press, 1970.

[10] P. G. Engeldrum, "Computing Color Gamuts of Ink-Jet Printing Systems," in *SID 85 Digest*, pp. 385–388, 1985.

[11] R. M. Evans, W. T. Hanson Jr., W. L. Brewer, *Principles of Color Photography.* New York: John Wiley & Sons, 1953.

[12] R. W. Hamming, *Digital Filters.* Englewood Cliffs, NJ: Prentice-Hall, 1977.

[13] R. W. G. Hunt, *The Reproduction of Colour in Photography, Printing & Television.* Fourth Edition. Tolworth, England: Fountain Press, 1987.

[14] R. W. G. Hunt, I. T. Pitt, L. M. Winter, "The Preferred Reproduction of Blue Sky, Green Grass and Caucasian Skin in Colour Photography," in *Journal of Photographic Science*, vol. 22, pp. 144–149, May/June 1974.

[15] J. Kenney, "Careful Color Matching Makes Hardcopy Output Conform To CRT Display," in *Computer Technology Review*, pp. 167–175, Fall 1985.

[16] J. S. Lim, *Two-Dimensional Signal and Image Processing.* Englewood Cliffs, NJ: Prentice-Hall, to be published.

[17] A. V. Oppenheim, R W. Schafer, *Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1975.

[18] A. V. Oppenheim, R. W. Schafer, T. G. Stockham, "Nonlinear Filtering of Multiplied and Convolved Signals," in *Proceedings of the IEEE*, vol. 56, pp. 1264–1291, August 1968.

[19] Y. Ovchinnikov, I. Fainberg, R. Litvan, I. Solntsev, N. Avatkova, "A new approach to programming in photomechanical reproduction," in *Twelfth International Conference of Printing Research Institutes.*

[20] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, *Numerical Recipes in C.* New York: Cambridge University Press, pp. 467–470, 1988.

[21] L. R. Rabiner, B. Gold, *Theory and Application of Digital Signal Processing.* Englewood Cliffs, NJ: Prentice–Hall, 1975.

[22] J. J. Sara, "The Automated Reproduction of Pictures with Nonreproducible Colors," *Ph. D. Thesis*, Massachusetts Institute of Technology, August 1984.

[23] W. F. Schreiber, "A Color Prepress System Using Appearance Variables," in *Journal of Imaging Technology*, vol. 12, pp. 200–211, August 1986.

[24] W. F. Schreiber, *Fundamentals of Electronic Imaging Systems.* New York: Springer-Verlag, 1986.

[25] W. F. Schreiber, "Image processing for quality improvement," in *Proceedings of the IEEE*, vol. 66, no. 12, December 1978.

[26] R. D. Solomon, "Color Coding for a Facsimile System," *Ph. D. Thesis*, Massachusetts Institute of Technology, August 1975.

[27] M. F. Southworth, *Color Separation Techniques.* Second Edition. Livonia, NY: Graphic Arts Publishing, 1979.

[28] T. G. Stockham, "Image Processing in the Context of a Visual Model," in *Proceedings of the IEEE*, vol. 60, pp. 828–842, July 1972.

[29] M. C. Stone, W. B. Cowan, J. C. Beatty, "Color Gamut Mapping and the Printing of Digital Color Images," ©Xerox Corporation, April 1988.

[30] M. C. Stone, W. B. Cowan, J. C. Beatty, "A Description of the Color-Reproduction Methods Used for This Issue of Color Research and Application," in *Color Research and Application*, vol. 11, pp. S83–S88, supplement 1986.

[31] D. E. Troxel, W. F. Schreiber, "Interactive enhancement of tone scale," in *Optical Engineering*, vol. 21, pp. 841–846, September/October, 1982.

[32] R. Ulichney, *Digital Halftoning*. Cambridge, Massachusetts: MIT Press, 1987.

[33] G. Vachon, "Modeling the Mixing Behavior of Inks with Polynomials," in *Color research and application*, vol. 13, pp. 46–49, February 1988.

[34] B. Woo, "A Survey of Halftoning Algorithms and Investigation of the Error Diffusion Technique," *S.B. Thesis*, Massachusetts Institute of Technology, May 1984.

[35] W. D. Wright, *Researches on Normal and Defective Colour Vision*. London: Kimpton, 1946.

[36] G. Wyszecki, W. S. Stiles, *Color Science*. Second Edition. New York: John Wiley & Sons, 1982.

[37] J. A. C. Yule, *Principles of Color Reproduction*. New York: John Wiley & Sons, 1967.