INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
Color image enhancement by three-dimensional histogram modification

Mlsna, Phillip Anthony, M.S.

The University of Arizona, 1992
COLOR IMAGE ENHANCEMENT BY
THREE-DIMENSIONAL HISTOGRAM MODIFICATION

by
Phillip Anthony Mlsna

A Thesis Submitted to the Faculty of the
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
In Partial Fulfillment of the Requirements
For the Degree of
MASTER OF SCIENCE
WITH A MAJOR IN ELECTRICAL ENGINEERING
In the Graduate College
THE UNIVERSITY OF ARIZONA

1992
STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Phillip A. Mloncz

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

Jeff Rodriguez 12-8-92
Jeffrey J. Rodríguez Date
Assistant Professor of Electrical and Computer Engineering
Acknowledgments

I would like to express my sincere gratitude to my advisor, Dr. Jeffrey J. Rodríguez, for the suggestion of this research topic and for his guidance and support throughout this project. Thanks are also extended to Drs. Marcellin and Schowengerdt for their participation on the committee and for their helpful comments.

I would like to thank the members of my family, who have always supported my education with their love, encouragement, patience and support. Finally my cats, Mira and Scheherazade, who kept me firmly anchored to the real world by demanding to be fed on time every day.
To my wife, Ann,

whose love, ceaseless encouragement and unswerving optimism
always helps me to stand tall
# Table of Contents

LIST OF FIGURES ................................................. 7

ABSTRACT ......................................................... 9

1 INTRODUCTION .................................................. 10

2 BACKGROUND .................................................... 14
   2.1 Commonly Used Color Spaces .............................. 14
   2.1.1 RGB Color Space ........................................ 14
   2.1.2 Color Spaces of Hue, Saturation, and Brightness ..... 17
   2.1.3 YIQ color space ........................................... 20
   2.2 Histogram Equalization of Monochrome Images .......... 21
   2.3 Methods of Color Image Enhancement ..................... 22
      2.3.1 Independent Treatment of Color Components ....... 22
      2.3.2 Monochromatic Enhancement of Brightness .......... 25
      2.3.3 Enhancement of Both Brightness and Saturation ..... 26
      2.3.4 Decorrelation Enhancement ............................ 27
      2.3.5 Color Equalization ...................................... 32
      2.3.6 Other Methods ........................................... 33

3 HISTOGRAM EXPLOSION ....................................... 34
   3.1 Goals ....................................................... 34
   3.2 Basis of Histogram Explosion .............................. 35
   3.3 Characteristics and Options ............................... 39
      3.3.1 One-Sided Histogram .................................. 39
      3.3.2 Choice of Operating Point .............................. 42
Table of Contents - *Continued*

3.3.3 Shifting the Histogram Before Explosion ......................... 48
3.3.4 Modification Along the Rays ................................. 50
3.4 Computational Issues ........................................... 56
   3.4.1 Computing the RGB Histogram Table ....................... 57
   3.4.2 Computing the Explosion ................................ 62
   3.4.3 Possible Speed Improvements .............................. 65

4 RESULTS ................................................................... 69
   4.1 Software ......................................................... 69
   4.2 Images .......................................................... 70

5 MULTIDIMENSIONAL HISTOGRAM EQUALIZATION ......................... 83
   5.1 Goals ............................................................ 83
   5.2 Iterative Equalization of Local Volume Density ............... 84
      5.2.1 Approach ................................................ 84
      5.2.2 Computational Issues .................................. 88
   5.3 Simulated Annealing ............................................ 90
   5.4 Parallel Computing ............................................ 91

6 CONCLUSION ................................................................ 93

APPENDIX A ONE-SIDED HISTOGRAM EQUALIZATION ..................... 96
REFERENCES .................................................................. 100
List of Figures

Figure 2.1 RGB color space diagram ................................................. 15
Figure 2.2 Maxwell triangle showing rgb coordinate system ..................... 17
Figure 2.3 Independent histogram equalization shown in two dimensions .......... 24
Figure 2.4 Decorrelation stretch shown in two dimensions ....................... 28
Figure 3.1 Histogram explosion concept in two dimensions ....................... 37
Figure 3.2 Definition of histogram explosion parameters ......................... 38
Figure 3.3 One-sided histogram equalization .................................... 41
Figure 3.4 Effect of innermost bins merging at unoccupied operating point bin . 44
Figure 3.5 Effect of operating point placement on bimodal histogram ............ 46
Figure 3.6 Effect of operating point location on local output density ............. 49
Figure 3.7 1-D histogram modification along the ith ray ......................... 51
Figure 3.8 Tendency of bins to concentrate near operating point ................. 53
Figure 3.9 First order interpolation between adjacent bins in RGB space ........ 54
Figure 3.10 Comparison of two methods of constructing the histogram table ... 61
Figure 3.11 Flowchart of histogram explosion algorithm .......................... 64
Figure 4.1 Photograph and histogram plot of original image SFO ................. 73
Figure 4.2 Photograph and histogram plot of SFO following independent histogram equalization of R, G, and B components ............................ 74
Figure 4.3 Photograph and histogram plot of SFO following the decorrelation stretch ............................................................................... 75
Figure 4.4 Photograph and histogram plot of SFO after histogram explosion using linear stretch along the rays ................................................. 76
Figure 4.5 Photograph of SFO after histogram explosion by histogram equalization along the rays .......................................................... 77
Figure 4.6 Photograph of SFO following histogram explosion with the operating point centered in RGB space ................................. 78
List of Figures - Continued

Figure 4.7 Photograph of SFO after histogram equalization of only the Y component in YIQ space ............................................. 79

Figure 4.8 Photograph of original image Kenya ................................................. 80

Figure 4.9 Photograph of Kenya after the R, G, and B components have been independently equalized ................................................. 81

Figure 4.10 Photograph of Kenya after histogram explosion about the centroid . . 82
Abstract

Histogram-based color image enhancement is usually accomplished by transforming from RGB coordinates to another coordinate system, modifying the components represented in that system, and converting the results back to RGB. Although a few methods function directly in RGB space, they also attempt to reduce dimensionality from the histogram's original three dimensions. Such methods seldom yield images that use the full extent of RGB color range.

A new method called "histogram explosion" has been developed to perform true multivariate enhancement directly in RGB color space. Discussed are the algorithm's operational parameters, behavior, implementation, and possible improvements. Results show histogram explosion to be very effective and flexible in enhancing color images.

Finally, two iterative methods are suggested as possible approaches for the extension of the histogram equalization algorithm to operate upon three dimensional histograms.
Chapter 1

Introduction

The purpose of this work has been to develop new multidimensional methods for histogram modification of color images, thereby improving upon previous methods of contrast enhancement as applied to color. Improving image contrast using histogram modification techniques can substantially assist subsequent interpretation work by altering the visibility of image features. By relaxing traditional color perceptual constraints, such as the preservation of original hue, much greater freedom becomes available in the enhancement process. Detailed here is the novel method of histogram explosion which, unlike most previous approaches, processes the three color components simultaneously in a true multivariate fashion. Additionally included is a conceptual exploration of possible extensions of the histogram equalization algorithm to operate upon the multidimensional histograms of color images. Both histogram explosion and multidimensional histogram equalization attempt to increase color contrast in order to make greater use of the range of possible colors. When applied appropriately, histogram explosion provides more effective and flexible color contrast enhancement than that previously possible. The characteristics and effectiveness of histogram explosion are demonstrated by experimental results. It is expected that multidimensional histogram equalization should also produce very effective enhancement. Two fundamentally distinct approaches to multidimensional
histogram equalization are qualitatively investigated and the practicality of each is briefly examined.

Histogram-based enhancement techniques have traditionally been applied to monochrome images primarily to exaggerate features of subtly differing brightness, thus increasing the visibility of such features to human interpretation. Many of the monochromatic enhancement techniques have been extended to the processing of color images. Instead of dealing with a single spectral component as in the case of monochrome images, color images require the specification of three spectral components. This three dimensional information can be represented in many possible color spaces. The characteristics of the human visual system's color response dictate the kinds of color manipulations that are useful and subjectively desirable to assist people in interpreting color images. Most of the existing color techniques, therefore, involve transformation to and operation within color spaces that are designed to mimic certain aspects of human color perception. In this way, the nature of the enhancement becomes constrained to the parameters and characteristics of the color space employed. For example, several color enhancement methods involve converting an image into a representation in which hue serves as one of three perceptual attributes. The hue parameter then remains unchanged in order not to distort or destroy the visual cues upon which humans often rely for image interpretation purposes.

Most existing color enhancement techniques place a premium on obtaining results that people consider improved based on human perceptual subjectivity. When this perceptual bias is removed, greater freedom of enhancement becomes possible. This less constrained approach may involve the alteration of some visual cues expected by the human observers, but is important in achieving the goals of improved
color range utilization and increased visibility of features whose color contrast is rather subtle.

Of course, modifying perceptual attributes such as hue during enhancement by some new technique does not necessarily preclude the usefulness of that technique as an aid to visual interpretation. If such perceptual parameters as hue become altered in a way that exaggerates the differences between features in an image, then these differences can become more apparent to a human interpreter. Although the observer may experience some confusion because of the change in perceptual cues, in some applications this is less important than one's ability to detect subtle differences between image features. Applied judiciously, such perception-altering enhancement can be effectively used to aid visual interpretation.

After surveying several existing ways of contrast enhancement as applied to color images, the effective new method of histogram explosion is presented and analyzed. Histogram explosion attempts to expand a histogram about a chosen color position so the outermost occupied bins move radially outward to the boundaries of the color space. The remaining bins become shifted according to a specified transformation function, which may be either linear or nonlinear. In so doing, the method produces improved color contrast compared to the original image.

Histogram explosion begins by deriving the three dimensional histogram of an image. For each color present in the image, it defines a ray which passes through that color position in RGB color space and examines the histogram characteristics encountered along that ray. It then proceeds to adjust the color according to a specified transformation function, incorporating that color's ray properties. The various characteristics and options of the histogram explosion algorithm are discussed.
in detail, followed by a presentation and discussion of the results of computer implementation.

The popular method of histogram equalization has proven to be a powerful and useful method for enhancing monochrome images, mainly because of its ability to adjust a given histogram to become approximately uniform. If the same effect could be realized in the three dimensional realm of color image histograms, some of the properties and benefits of equalized one dimensional brightness histograms might then extend to color. Although histogram explosion is adept at improving the use of the color range, it is not intended to produce uniform multidimensional histograms.

Following the discussion of histogram explosion, a conceptual exploration of possible ways to accomplish multidimensional histogram equalization is presented. How such an algorithm might be constructed, how it might behave, and its benefits and difficulties are the prime topics of this discussion. Two main iterative approaches are considered. One of them involves the simulation of a system wherein the histogram bins exert mutually repulsive forces. Each occupied bin is allowed to move in reaction to the net force imparted by neighboring bins, leading the system toward a more equalized distribution at each step. The other approach uses simulated annealing to proceed toward equalization. Included is a brief exploration of the computational complexity for the two suggested approaches.
Chapter 2

Background

A number of color image enhancement methods based on histogram modification have been developed over the last couple of decades. Many of them involve transformations into other coordinate systems in an attempt to reduce the number of dimensions that must be considered by the enhancement algorithm. Before discussing the details of these methods, it would be useful to investigate several of the color coordinate systems they employ.

2.1 Commonly Used Color Spaces
2.1.1 RGB Color Space

One of the most widely used color spaces is the RGB coordinate system in which a color's red, green, and blue components are each represented as a value in the interval [0,1]. For computer use, these values are generally scaled and quantized to integer values in a range [0, L−1], where L is the number of quantization levels available. Practically all modern color computer displays are provided with RGB data as their input because their CRT's utilize red, green, and blue as the primary phosphor colors. Consequently, the machines that generate display data must convert their color information to this form from whatever alternate color representations might be used internally. Image data files also typically represent pixel data in RGB form, either directly or as a simple index to a look-up table containing the RGB values.
There are $L^3$ different color values representable in this system. The R, G, and B parameters can be visualized as the cartesian coordinate axes, forming the cube-shaped color space shown in Figure 2.1.

![RGB color space diagram](image)

Figure 2.1 RGB color space diagram. The common names of colors located at each corner of the cube are shown.

A line segment from the black corner to the white corner describes the entire set of gray values, namely those that a human observer would describe as colorless. Each of these points has component values such that $R=G=B$. The center of the gray line is also the midpoint of the color cube. The complement of a color is defined as that position in the cube lying equidistant from the midpoint but in the opposite direction. Thus cyan and red, yellow and blue, and black and white are pairs of complementary colors. One further noteworthy point is that the surface defined by the plane $R+G+B=1$ is the Maxwell color triangle.
RGB representation suffers from at least one major disadvantage. Since it is not based on qualities of human perception, it is a perceptually nonuniform color space. This means that the visually apparent color change due to a given magnitude of displacement in RGB space depends on the displacement direction. This makes color enhancement in the RGB system difficult if the aim is to improve, or at least preserve, the visual characteristics of an image.

One occasionally encounters rgb tristimulus values, which are simply normalized RGB values:

\[
\begin{align*}
  r &= \frac{R}{R + G + B}, \\
  g &= \frac{G}{R + G + B}, \\
  b &= \frac{B}{R + G + B}
\end{align*}
\]  

(2.1)

assuming \(R+G+B > 0\). Note that \(r+g+b = 1\). The rgb representation is the coordinate system of the Maxwell color triangle, shown in Figure 2.2. Converting from RGB to rgb requires mapping a cube to an equilateral triangle, thus reducing the number of dimensions by one. Therefore, converting from rgb tristimulus values back to RGB coordinates does not allow full recovery of the correct RGB values unless some additional information, namely the brightness component, is reintroduced.
2.1.2 Color Spaces of Hue, Saturation, and Brightness

Several different coordinate systems are based on the general perceptual concepts of "hue", "saturation", and "brightness", although the systems differ in their precise models and terminology. Intensity, lightness, luminance, and value are some of the terms used to describe the amount of brightness. The remaining two attributes, hue and saturation, specify the chromaticity characteristics that differentiate color from grayness. Hue refers to the perceived colors associated with different visible wavelengths, such as blue, yellow, or green. Saturation refers to the purity of the hue. A fully saturated color differs in the greatest possible degree from a gray having equivalent brightness. As saturation decreases, the color appears to fade in purity until becoming fully gray when saturation reaches zero.
One of the simplest of this type of perceptually-based color system is the HSL, described in Thorell and Smith [1] and Glassner [2]. The HSL parameters of hue, saturation, and lightness define a double hexcone in cylindrical coordinates. A hexcone can be visualized as a cone whose cross section is a regular hexagon. By joining two identical hexcones at their hexagonal faces, a double hexcone is created. The HSL system can be described in terms of RGB values by the following equations, where m=max(R,G,B) and n=min(R,G,B):

\[
L = \frac{m+n}{2} \tag{2.2.a}
\]

\[
S = \begin{cases} 
\frac{m-n}{2}, & L \leq 0.5 \\
\frac{m-n}{2-m-n}, & L > 0.5 
\end{cases} \tag{2.2.b}
\]

\[
H = \frac{\pi}{3(m-n)} \begin{cases} 
5+m-B, & R=m & G=n \\
1-m+G, & R=m & B=n \\
1+m-R, & G=m & B=n \\
3-m+B, & G=m & R=n \\
3+m-G, & B=m & R=n \\
5-m+R, & B=m & G=n 
\end{cases} \tag{2.2.c}
\]

A main advantage of HSL is its onto mapping to the RGB color cube, allowing transformations between the two systems to always generate permissible values. The L axis corresponds to the gray line of the RGB cube, with black and white located at opposite ends of this axis and mapping to 0 and 1 respectively. S values lie on the interval [0,\(a\)], where \(a=1\) when \(L=0.5\) and \(a\) approaches zero as \(L\) approaches zero or one. In RGB space, the saturation \(S\) describes the radial distance from the gray line to the particular color point. Hue can be envisioned as an angle lying within a plane perpendicular to the gray line and whose vertex lies on that line. The hue of any color
in a plane of constant $L$ is the angle relative to some arbitrary reference direction, which the HSL system defines as red.

Since $H$ directly corresponds to an angle about the gray line axis, it should be clear that surfaces of constant hue are actually half-planes in RGB space, radiating from the gray line at various angles. The surface of constant saturation can be visualized in the RGB system as a cylinder of radius $S$, centered on the gray line. One of the extremes is the case of maximum saturation, where the $S=1$ surface intersects the RGB cube only at the six corners that do not lie on the gray axis. The other extreme is that of minimum saturation, $S=0$, where the cylindrical surface has collapsed to the gray line itself.

Another perceptual space, described in both Thorell and Smith [1] and Glassner [2], has the parameters of hue, saturation, and value. The main advantage HSV has over HSL is a simpler transformation calculation back to RGB. HSV specifies the cylindrical coordinates defining a hexcone-shaped region as follows:

$$V = \max(R,G,B)$$

$$S = \begin{cases} 
\frac{\max(R,G,B) - \min(R,G,B)}{\max(R,G,B)}, & \max(R,G,B) > 0 \\
0, & \max(R,G,B) = 0 
\end{cases}$$

$$H = \frac{\pi}{3} \cdot \begin{cases} 
\frac{G - B}{\max(R,G,B) - \min(R,G,B)}, & R = \max(R,G,B) \\
\frac{B - R}{\max(R,G,B) - \min(R,G,B)}, & G = \max(R,G,B) \\
\frac{R - G}{\max(R,G,B) - \min(R,G,B)}, & B = \max(R,G,B) \\
\text{undefined}, & S = 0 
\end{cases}$$
Gillespie [3] has described the HSI system, whose definition of hue, saturation, and intensity is somewhat more complicated than that of either HSL or HSV. The LHS coordinate system, defining lightness, hue, and saturation somewhat differently, was introduced by Strickland [4, 5]. This space is also similar in basic concept to HSL and HSV. Gillespie and Strickland have discussed certain types of color image enhancement methods using their respective systems.

2.1.3 YIQ color space

The YIQ system is commonly used in color television and was developed so that color broadcasts would be compatible with black-and-white receivers. The Y component is based on human perception of brightness as a function of wavelength, with the green component weighted more heavily than those of red or blue. The I and Q components are the chromaticity parameters chosen to minimize transmission bandwidth based on a model of human color sensitivities.

Conversion between RGB and YIQ spaces is a simple matter because of the linear transformation involved.

\[
\begin{bmatrix}
Y \\
I \\
Q
\end{bmatrix} = \begin{bmatrix}
0.299 & 0.587 & 0.114 \\
0.596 & -0.274 & -0.322 \\
0.211 & -0.523 & 0.312
\end{bmatrix}
\begin{bmatrix}
R \\
G \\
B
\end{bmatrix}
\]

(2.4.a)

\[
\begin{bmatrix}
R \\
G \\
B
\end{bmatrix} = \begin{bmatrix}
1.000 & 0.956 & 0.621 \\
1.000 & -0.272 & -0.647 \\
1.000 & -1.106 & 1.703
\end{bmatrix}
\begin{bmatrix}
Y \\
I \\
Q
\end{bmatrix}
\]

(2.4.b)

Note that some of the colors representable in YIQ space do not have a corresponding color inside the RGB cube. For example, YIQ = (0.1, 1.0, 0.0) transforms to RGB = (1.056, -0.172, -1.006), which lies far outside the boundary of the RGB cube. Therefore, for display purposes this color must somehow be altered so it lies inside the
RGB cube. Distortion is the result; the perceptibility of this distortion depends on several factors including how many pixels are thus modified and how far they must be shifted. In order to avoid this distortion entirely, it is necessary to carefully restrict the enhancement process in YIQ space to ensure that all modified colors are invertible to proper RGB values.

2.2 Histogram Equalization of Monochrome Images

One of the most widely employed methods of histogram modification for monochromatic images is histogram equalization, which attempts to achieve an approximately uniform gray-level histogram by nonlinear stretching of an original histogram. Forcing its histogram to become approximately uniform can enable an image to reveal subtle details by exaggerating differences in gray levels based on their frequencies of occurrence. Also, equalization forces the image to utilize the full extent of the gray level range. A brief review of the theoretical foundation of histogram equalization can be found at the beginning of Appendix A.

The details of histogram equalization are discussed in Gonzalez and Wintz [6]. First, the histogram is constructed from the image data:

$$h(x_k) = \frac{n_k}{N}$$

(2.5)

where $x_k$ is the kth gray level within the range $[0,L-1]$, $N$ is the total number of pixels in the image, and $n_k$ is the number of pixels having gray level $x_k$. Next, the cumulative distribution is computed:

$$T(x_k) = \sum_{j=0}^{k} h(x_j)$$

(2.6)
The cumulative function $T(x_k)$ lies on the interval $[0,1]$ and is a monotonic non-decreasing function of $x_k$. Transformation of all pixels can then be accomplished according to the mapping:

$$x_k \Rightarrow (L-1) T(x_k) \quad (2.7)$$

The monotonic non-decreasing nature of the transformation function preserves the original order of the bins.

### 2.3 Methods of Color Image Enhancement
#### 2.3.1 Independent Treatment of Color Components

There are many possible ways to reduce the dimensionality of the color histogram data so that enhancement processing can be greatly simplified and thus proceed quickly. One of these is quite straightforward and does not require a transformation from RGB into another color space. Simply by treating the R, G, and B components independently, we can separately process them with any desired monochromatic algorithm and merge the three sets of results into a new color composite. This method demands basically the same amount of computation as for processing three monochrome images. Extra computation may be required only for extracting the R, G, and B components from the color image data before processing and later combining their enhanced versions into a new RGB composite.

Separate processing of the three component histograms assumes that the components are mutually independent and therefore separable, requiring the multidimensional histogram to be a product of the component histograms:

$$h(R,G,B) = h(R) \cdot h(G) \cdot h(B) \quad (2.8)$$
If the three components were independent, then they could be separately processed to obtain results identical to those of a truly three-dimensional version of the same algorithm. That would mean multidimensional approaches would be unnecessary because the task could always be separated into a series of one dimensional problems.

In general, however, separability is a poor assumption for all but a very few color images because few have truly independent components. In fact, the overwhelming majority of color images have non-zero amounts of inter-band cross-channel correlation, prohibiting any possibility of their components being mutually independent. Therefore, processing accomplished by separating the color components is quite likely to cause distortions in unexpected and uncontrolled ways. Some images might show very pleasing results while others exhibit severe color anomalies or other kinds of artifacts. The product of three 1-D histograms does not define a unique 3-D histogram. In other words, when the image components are not mutually independent, important information is lost by processing them separately.

When histogram equalization is applied separately to the R, G, and B components of a color image, the results have a few interesting characteristics. No matter what the shape of the original histogram’s data cluster in 3-space, each of the six boundary faces of the RGB cube will be touched by at least one occupied bin of the modified data cluster. This is a consequence of the fact that, except for the degenerate case involving only one occupied component bin, histogram equalization will force the three marginal histograms to use the full extents of their available ranges. However, as illustrated in Figure 2.3, the new 3-D histogram data cluster will seldom use the full extent of RGB space.
One property of histogram equalization in one dimension is that the relative order of the bins is maintained because the transformation function is monotonic non-decreasing. This characteristic is important in preserving as much of the information of the original image as possible. Independently equalizing the marginal histograms also preserves the original multidimensional bin order in the sense that no reordering of the marginal histograms is allowed. Specifically, one may choose any two occupied bins $i$ and $j$ in the original histogram and note in which octant $j$ lies relative to $i$ (e.g., $R_i < R_j$, $G_i > G_j$, $B_i < B_j$). After separately equalizing the components, the new $j$ must appear in the same octant relative to the new $i$ because the same relationships must hold between their components.
2.3.2 Monochromatic Enhancement of Brightness

A second general method of reducing dimensionality is to transform into another color coordinate system, then independently process the new parameters. A simple example is to transform into HSL space, modify the histogram of the lightness component only, then transform the results back to RGB space. Similarly, one might transform to YIQ coordinates and enhance only the Y component or modify the V component of the image in HSV space. Such schemes attempt to improve the contrast of the brightness component while leaving the chromaticity information unaltered. In general, enhancement of the brightness component tends to be reasonably well-behaved, primarily because its similarity to the enhancement undergone by monochrome images. Alteration of other perceptual aspects, such as hue, is avoided.

The technique is quick and efficient because computation of the two coordinate transformations is all that is added to the job of the desired monochrome enhancement method.

The approach of brightness component enhancement can have a major drawback arising from the lack of a one-to-one mapping between the RGB space and the space in which histogram modification is to take place. Enhancing the Y component of the YIQ color space is one example of this effect, but there are others as well. Some bins of the modified histogram in this new coordinate space may transform to RGB positions outside the boundaries. If this occurs, some accommodation scheme must be invoked so that all occupied bins are inverse transformed to legitimate RGB values. One such scheme is simply clipping to the nearest valid RGB position, but this is likely to cause artifacts in the output image. Another scheme is to constrain the brightness enhancement in the alternate color space so the entire data cluster is transformable to legitimate RGB positions.
Another limitation of brightness-only enhancement is that there is no opportunity for enhancing the chromaticity information. In fact, increasing only the brightness contrast of a color image can diminish the effect of color perceived in that image. The chromaticity components tend to appear more subdued than in the original, mainly because the increased brightness contrast is now more dominant relative to the unchanged chromatic elements.

2.3.3 Enhancement of Both Brightness and Saturation

Somewhat more sophisticated methods have been discussed by Gillespie and Strickland involving modification of both the brightness and saturation components. Strickland argues that adjustment of saturation must be done carefully to avoid visually objectionable results. Both Strickland and Gillespie believe hue should remain constant in order to retain the basis of color cues expected by the people who will interpret the images. In effect, their methods consist of processing in two of the three dimensions of an HSV-like color space.

Gillespie advocates transforming to HSI space, stretching I and S independently, then transforming back to RGB. The quantization effects in this procedure can alter H slightly, but the effect is usually insignificant. The goal is primarily to exaggerate color saturation and next to improve brightness contrast, all without affecting hue. Strickland's method involves transforming to LHS coordinates, then adjusting L and S in the following manner:

\[
L'(x,y) = L(x,y) + k_1 \left[ L(x,y) - \overline{L(x,y)} \right] + k_2 \left[ S(x,y) - \overline{S(x,y)} \right] \quad (2.9.a)
\]

or

\[
L'(x,y) = L(x,y) + k_3 \left[ \max \left\{ L(x,y) - \overline{L(x,y)} ; S(x,y) - \overline{S(x,y)} \right\} \right] \quad (2.9.b)
\]
where $\overline{L(x,y)}$ and $\overline{S'(x,y)}$ are the means of $L(x,y)$ and $S'(x,y)$ over a local 3x3 pixel window.

Neither of these two approaches yield histogram data that uses the full extent of the available color range. The methods are intended primarily for visual enhancement, and therefore are more constrained in their effects. The issue of separability of the color components once again arises here, too. Transforming to a hue-saturation color coordinate system will not force the coordinate parameters to become mutually independent.

### 2.3.4 Decorrelation Enhancement

Gillespie [3] has also discussed what is called the "decorrelation stretch", developed by Taylor [7] and refined by Soha and Schwartz [8], designed to enhance images that exhibit large correlation between their RGB color components. The three dimensional histogram data clusters of such images tend to have elongated shapes, causing the images to poorly represent the full range of color. The decorrelation stretch employs the approach of the Karhunen-Loeve or principal components transformation, which is based on eigen analysis, to determine the directions of maximum correlation in RGB color space. The eigenvectors describe a new coordinate system that is simply a rotation of the original system. The image is transformed into this new coordinate system, then the data cluster is stretched independently in the directions of the secondary and tertiary eigenvectors so their corresponding variances (the eigenvalues) become equal. The final step is to transform the results back to the original RGB space. Many images processed in this way can become more colorful because the output data cluster often extends across a
greater volume of color space than the original one. Figure 2.4 illustrates the basic principles of the decorrelation stretch.

Figure 2.4 Decorrelation shown in 2 dimensions. \( v_1 \) and \( v_2 \) are the primary and secondary eigenvectors. Expansion occurs along the \( v_2 \) axis so its variance matches that along the \( v_1 \) axis. Once decorrelated, further stretching can be done equally on the components as long as no bin ventures beyond the RGB limits.

If \( N \) is the number of pixels in the image, the point \( (\mu_R, \mu_G, \mu_B) \) is the centroid of the image's histogram data cluster, and \( f(R_j) \) is the fraction of total pixels having a red component value of \( R_j \), then the covariance matrix is constructed as follows:

\[
\sigma_R^2 = \sum_{i=1}^{N} f(R_i)(R_i - \mu_R)^2, \quad \sigma_{RG} = \sum_{i=1}^{N} \sum_{j=1}^{N} \left[f(R_i, G_j) R_i G_j - \mu_R \mu_G\right]
\]

\[C = \begin{bmatrix}
\sigma_R^2 & \sigma_{RG} & \sigma_{RB} \\
\sigma_{RG} & \sigma_G^2 & \sigma_{GB} \\
\sigma_{RB} & \sigma_{GB} & \sigma_B^2
\end{bmatrix}, \text{ where } \mu_R = \frac{\sum_{j=1}^{N} R_j f(R_j)}{\sum_{j=1}^{N} R_j}
\]

(2.10)
If $C$ is a nonsingular matrix, then its eigenvalues and eigenvectors can be computed and the decorrelation accomplished. Let $\Lambda$ be the covariance matrix of a rotated coordinate system determined by eigen decomposition. If $V$ is the orthonormal matrix necessary to determine the principal components of $C$, then the transformation to the eigenvector-based coordinate system is:

$$
\Lambda = V C V^T = \begin{bmatrix}
\alpha_{x1}^2 & 0 & 0 \\
0 & \alpha_{x2}^2 & 0 \\
0 & 0 & \alpha_{x3}^2
\end{bmatrix}, \quad \alpha_{x1}^2 \geq \alpha_{x2}^2 \geq \alpha_{x3}^2
$$

(2.12)

$\alpha_{x1}^2$, $\alpha_{x2}^2$, and $\alpha_{x3}^2$ are the eigenvalues, that is, the variances in the directions of the eigenvectors. The eigenvectors, which are the columns of $V$, determine the directions in which the histogram data cluster is to be stretched.

One can now modify these three new eigenvector components independently using any desired method, the easiest of which is simple linear scaling. If the scaling is applied so the three variances become equal, then the eigenvalue matrix becomes:

$$
\Lambda' = \begin{bmatrix}
k_1 \alpha_{x1}^2 & 0 & 0 \\
0 & k_2 \alpha_{x2}^2 & 0 \\
0 & 0 & k_3 \alpha_{x3}^2
\end{bmatrix} = \begin{bmatrix}
\alpha_{x}^2 & 0 & 0 \\
0 & \alpha_{x}^2 & 0 \\
0 & 0 & \alpha_{x}^2
\end{bmatrix} = \alpha_{x}^2 I
$$

(2.13)

where the $k_2$ and $k_3$ are the scale factors applied along the second and third eigenvectors to match their variances with that of the first eigenvector. The factor $k_1$ controls the stretching of the entire data cluster.

At this point, any rotation can be accomplished while maintaining uncorrelated components. If an arbitrary rotation matrix $R$ has orthonormal columns, then the covariance matrix resulting from rotation by $R$ is:

$$
C_R' = R \Lambda' R^T = \alpha_{x}^2 R I R^T = \alpha_{x}^2 I
$$

(2.14)
Therefore, the rotated histogram data cluster also possesses uncorrelated RGB components. The case of unequalized eigenvalues, however, would be unlikely to withstand this inverse rotation without introducing some correlation between the RGB components. Still, a greater spread of the data cluster should usually result if the principal components have been stretched properly.

Practically speaking, the scale factor $k_1$ will always have an upper bound because of the boundaries of the RGB cube. If $k_1$ is too large, some of the occupied bins will be relocated outside the RGB boundaries. One solution is to clip these wayward bins to the boundary surfaces, but this would create undesired artifacts. To help maximize $k_1$, translation of the entire data cluster can be incorporated into the decorrelation stretch procedure. By shifting the cluster so that its centroid moves to the center of RGB space, a larger value of the scale factor $k_1$ is likely to be usable before any bin is forced outside the RGB cube. The maximum usable value of $k_1$ is obtained by translating the cluster so occupied bins reach two opposing boundary faces of the cube. One significant drawback of translating a histogram data cluster is the potential for undesired alteration of certain perceptual characteristics, such as hue or brightness. This effect is described in more detail in Chapter 3.

The decorrelation stretch does not necessarily enable the output data cluster to use the full extent of the available color range. Under decorrelation, the 3-D histogram modification problem reduces to a set of three 1-D problems in the new eigenvector-based coordinate system. In most images, the visual effect is mainly to exaggerate some combination of the saturation and brightness components, although some hue change can occur. Large changes in hue are possible in images whose skewed histograms require translation in order to maximize the stretch factor $k_1$. 
Although conceptually straightforward, the decorrelation stretch demands care in its implementation. The goal is to spread the data cluster along the eigenvectors without venturing outside the RGB space. To accomplish this, the stretched data must be checked against the RGB boundaries and the factor $k_1$ must be decreased if these limits are exceeded. In some cases, $k_1<1$ may be the best way to balance the three eigenvalues while constraining the data to remain inside the RGB boundaries. Finally, the output data cluster will seldom be able to use the full extent of RGB space. In fact, there is no guarantee that the separate histograms of the generated R, G, and B components will fully extend across their available ranges.

Applying the decorrelation stretch to color images whose RGB parameters are initially uncorrelated is identical to the previously described enhancement method involving independent component modification. In fact, independently adjusting the principal components is the same kind of separate processing now applied in coordinates other than the original RGB. Recall that the results of separate processing of components is somewhat unpredictable because of the assumption of separability based on mere lack of correlation. Since zero correlation does not necessarily imply independence, there is likely to be some difference between the results of decorrelation and those of a truly joint 3-D treatment. Enhancing the components after decorrelation should help minimize the amount of distortion caused by the separate treatment of mutually-dependent components.

One briefly mentioned difficulty with decorrelation enhancement arises from the possible translation or shifting of the histogram data cluster to reduce skew. Such translation is not specifically part of the principal components transformation, but needs to be tailored to the results intended by the operator. Translating the cluster will alter the overall color bias of the image and can easily change perceptual
characteristics, as explained earlier. Skipping the translation step when the data cluster is well off-center can constrain the amount of enhancement that is possible.

2.3.5 Color Equalization

Bockstein [9] has proposed an enhancement method intended for producing uniform brightness and saturation histograms in the Ylφ system. The brightness function of that system is identical to that of the Y component of the YIQ color system. Saturation l is the distance from the center of the Maxwell triangle to the color position. The hue parameter φ is similar to those of the other color spaces discussed, but Bockstein has simplified the definition somewhat. Instead of defining the hue angle in terms of trigonometric functions, it is defined as a base hue angle (red, blue, or green) plus $2\pi/3$ times the normalized distance of the color position along the side of the Maxwell triangle. This roughly approximates the angular hue definition used in many other coordinate systems, but reduces the complexity of conversion to and from RGB.

Color equalization proceeds as follows. After conversion from RGB to Ylφ coordinates, the brightness histogram $h(Y)$ is calculated. Also, a series of K conditional saturation histograms $q_{kφ}(l)$ are calculated along the K discrete sections of hue. These sections can be visualized as wedges radiating from the center of the Maxwell triangle. Then, $h(Y)$ and all K of the $q_{kφ}(l)$ histograms are equalized to obtain $Z(Y)$ and $Z_{kφ}(l)$. Finally, the original Y and l parameters of each pixel are mapped to their new positions on the $Z(Y)$ and the appropriate $Z_{kφ}(l)$ histograms. Finally, this new Ylφ value is converted back to RGB coordinates. This color equalization method does not involve a change of hue.
The results of color equalization are roughly similar to those of Gillespie's and Strickland's enhancement methods for brightness and saturation. Since hue is unaffected, the color equalization method cannot ensure that the output data cluster uses the full extent of RGB color space.

2.3.6 Other Methods

Gillespie [10] has discussed the color ratio method, which is based on the idea that two correlated component images, say two bands of a multispectral set, can reveal hidden color detail if their normalized quotient is constructed on a pixel by pixel basis. Any three such ratio components can be composited for conventional display. The ratio process tends to eliminate the similarities and greatly exaggerate the differences between the original bands.

Several additional color spaces not discussed here may have usefulness for image enhancement. They are primarily perceptual spaces, some of which are intended to be perceptually uniform. Such spaces have advantages for optimizing interpretation by humans, but impose constraints on any enhancement that is accomplished within them. These constraints frequently work against the desires of both utilizing the full extent of the color range and increasing the visible contrast between subtly different color features.

All of the above methods of color image enhancement attempt to reduce the dimensionality of the histogram before the application of the enhancement process. There appear to be two main reasons why all three dimensions are not adjusted simultaneously: added computational complexity and the lack of an appropriate 3-D algorithm. It is likely that ways would eventually be devised to overcome the computational impracticalities of any useful three dimensional algorithm.
Chapter 3

Histogram Explosion

3.1 Goals

The visibility of features in a color image can be improved by increasing the differences between the colors of these features. The methods discussed in Chapter 2 are generally useful for achieving increased color contrast, but none are able to consistently make use of the full extent of representable and displayable colors. Feature visibility could be increased still further if the full extent of the color range could be used. For this reason, it is desirable to develop an enhancement algorithm that is capable of modifying an image to make use of the full extent of the color space.

In order to provide the greatest flexibility, the new algorithm need not be constrained by human perceptual qualities. Relaxing this perceptual constraint eliminates most of the reasons for the enhancement operation to function within one of the many perceptually-based color spaces used by most existing enhancement methods. Therefore, it becomes desirable to adjust the histogram directly in RGB space, averting the need for color space conversions. This preserves the maximum numerical resolution available in the original data because it avoids the inherent requantization that occurs during color space conversion. Also evaded is the need for scaling or clipping the histogram data during conversion to ensure conformity to the limits of the destination color space. Consequently, the goal becomes the expansion of a given histogram to make fullest practical use of the RGB color cube.
Even though human perceptual constraints are relaxed for the purpose of constructing a general algorithm, it is still desirable to enhance color images to aid human interpreters who desire retention of certain original perceptual qualities. By understanding how the general algorithm proceeds, one might apply it in a way that prevents alteration of certain visual characteristics. Such a method may be visually useful if applied with appropriate understanding.

Use of the full extent of the color space is a characteristic usually unattainable by means of separate component processing. Accordingly, the algorithm must operate in a truly three-dimensional manner, which is inherently much more computationally complex than for processing color components separately or even in pairs. The penalty paid in speed of computation relative to other methods should be acceptable when weighed against the improvements achieved.

It is not adequate to simply rearrange the occupied histogram bins in any order or organization that achieves full use of the color space. Also inadequate, in most cases, is merely relocating all occupied histogram bins to new positions along the boundary surfaces of the RGB cube. Both of these approaches usually destroy important information contained in the relative positioning of the occupied bins. Consequently, the enhancement process should try to minimize any tendency for rearranging the bins' relative positions. The relative brightness and chromatic variations in the enhanced image should, as much as possible, reflect those in the original.

3.2 Basis of Histogram Explosion

The well-known histogram equalization method, commonly used for enhancing monochrome images, operates by adjusting an image’s histogram to become as
uniform as possible over the entire range of possible gray levels. After equalization, there will be some pixels having the lowest and others having the highest possible gray level values. In other words, the boundaries of the histogram’s one-dimensional data cluster are expanded to coincide with the boundaries of the one-dimensional gray level space itself. Because of the monotonic non-decreasing transformation function used, the ordering of the histogram bins does not change; only their positions and relative spacings are adjusted.

Something roughly similar can be accomplished in the three-dimensional RGB color space. By reducing the 3-D histogramming problem to a series of 1-D problems, conventional methods such as histogram equalization or linear contrast adjustment can be employed to achieve the kind of enhancement desired. The idea, shown twodimensionally in Figure 3.1, is to perform a desired histogram modification along a series of rays emanating from a chosen operating point in RGB color space. Wide flexibility in the operation of the method is mainly provided by the freedom in choosing both the operating point location and the 1-D histogram modification function.
Figure 3.1 Illustration of the histogram explosion process in two dimensions. All colors present in the image lie inside the data cluster boundary, but some interior bins may represent unused colors. Explosion causes the outermost occupied bins, those on the data cluster boundary surface, to expand outward to the color space boundaries. Interior bins shift along the rays according to the ray transformation used.
Figure 3.2 The technique of histogram explosion showing the important parameters involved. The ray extends from the operating point $P$ to the boundary intersection at $B_i$. $C_i$ is initial position of the bin of interest and $C_i'$ is its final position. Uppercase denotes vector quantities in RGB space; lowercase parameters are their mappings to the 1-D ray histogram.

The essentials of the histogram explosion algorithm, illustrated in Figure 3.2, are:

1. From the image, construct the 3-D RGB histogram consisting of the set of all colors $C_i$ and counts $f_i$.
2. Choose desired operating point $P$ somewhere in RGB space.
3. For each color $C_i$ with $f_i \neq 0$,
   a. define a ray from $P$ extending through $C_i$ to the color space boundary, point $B_i$.
   b. construct a 1-D histogram along the ray interpolated from the 3-D histogram data.
   c. modify this 1-D histogram using a desired method confined to the interval $[p,b_i]$.
   d. map the new ray position $c_i'$ to its position $C_i'$ in RGB coordinates.
   Enter this into the transformation table.
4. Using the transformation table in step 3, perform mapping \( C_i \Rightarrow C_i' \) for all pixels.

Notice that instead of separating the 3-D histogram into three 1-D ones, as in most of the existing approaches, histogram explosion decomposes it into a large number of them. Each occurring color receives its own tailored adjustment. In effect, the explosion process operates in a multivariate sense.

The name "histogram explosion" is descriptive of the nature of the technique. Each histogram bin is able to move only directly toward or away from the operating point in the RGB color space. In most images, depending upon the 1-D transformation method employed, the vast majority of bins will move outward from the operating point rather than inward toward it. Although the analogy is far from perfect, one can visualize this effect as resembling the starburst of an aerial fireworks explosion.

Most of the characteristics discussed in this chapter are more clearly understandable if illustrated in two dimensions instead of three. One can mentally extrapolate most of the accompanying figures to three dimensions for a more accurate understanding of the actual situations depicted. Therefore, the 2-D illustration of histogram explosion in Figures 3.1 and 3.2 can be thought of as a 3-D histogram in RGB space with the ray no longer confined to the R-G plane.

3.3 Characteristics and Options

3.3.1. One-Sided Histogram

Some thought on the above algorithm will reveal why rays instead of lines are used as the basis for the 1-D histogram computations. Consider the line in 3-space
defined by the operating point and a particular occupied histogram bin. It is desired to compute the new RGB coordinates of this particular subject bin. After determining the 1-D histogram along the line, this histogram is to be altered according to a selected method. All bins in this 1-D histogram, including that of the operating point itself, are then free to move as the method applies its modification. Thus, it is possible for the subject bin to shift across the original position of the operating point. Since the bins’ new positions are governed by individual line histograms that might differ greatly, two originally closely spaced bins lying in similar directions from the operating point could settle on opposite sides of it. This effect is usually undesirable because the 3-D bin order can be greatly altered. It would also prevent control over the alteration of hue characteristics when such control is desired.

By constraining the process so no bin is allowed to cross the operating point’s original position, the reordering problem diminishes significantly. This is especially true where the operating point is concerned because bins are no longer able to change their directions relative to it. The improvement in characteristics is obtained by using operating point rays rather than lines. For a practical example, consider that explosion using rays can be accomplished such that hue is preserved. All that is necessary is to choose an operating point lying somewhere along the gray line. If explosion were accomplished using lines instead of rays, some bins might shift across the gray line and cause drastic hue changes.

Consider again the conventional 1-D histogram equalization method. Instead of equalizing the entire histogram at once, the process might be broken into two parts by choosing a bin, fixing its position at a desired place, and then performing separate histogram equalization computations outward from it in opposite directions. Figure 3.3 illustrates the behavior of the one-sided form of histogram equalization. This
procedure is not guaranteed to give the same results as conventional histogram
equalization unless the chosen bin is fixed at the correct final gray level position. A
fairly close approximation to a "true" histogram equalization is usually obtained by
selecting the bin having the most nearly equal accumulated bin counts to either side.
After shifting the entire histogram so this median bin lies at the center of the range,
one-sided equalization can be performed in each direction. The properties of one-
sided histogram equalization are derived in Appendix A.

![Image]

Figure 3.3 (a) Original histogram in which, for illustration purposes, all occupied
bins have equal counts. Point $d$ indicates the median bin. (b) One-sided
histogram equalization with point $b$ as the operating point. Points $a$ and $c$ have
moved to the ends of the range. Note the density imbalance. (c) True histogram
equalization. Results approximately equal to this can also be obtained by shifting
the original histogram so point $d$ moves to the center of the range, then
performing one-sided equalization in each direction.

The arbitrary choice of an operating point still permits one-sided equalization to
retain many of the characteristics of "true" 1-D histogram equalization. For instance,
as long as the operating point is chosen to lie between the outermost occupied bins,
one-sided processing will move these outermost bins to the extremes of the gray level
range. Also, the 1-D order of the histogram bins will be unchanged no matter where
the operating point is located. The main drawback is the possible creation of disparity
between the overall density levels on either side of the chosen operating point. See Figure 3.3(b).

Extending this idea of one-sided histogram equalization to the three dimensional problem, one can allow bins to slide only along rays that radiate outward from the fixed operating point. None of the bins along a given ray is able to cross the operating point to the counterpart ray. Density imbalances are possible about this operating point along a line consisting of an opposing pair of rays. However, the operating point position has tremendous influence on the achieved 3-D density. Usually, the density imbalances between opposite rays are less important than the overall 3-D density, which is the characteristic of chief interest.

3.3.2 Choice of Operating Point

Which point should be selected as the operating point for an explosion calculation? Since the direction of the ray through a particular occupied histogram bin depends on the location of the operating point, and since the bin movement can only occur along that ray, the operating point location has great influence on the 3-D histogram modification achieved. The operating point serves as the focus of all 1-D histogram rays; it is the radiant point of the explosion process.

For purposes of this discussion, the data cluster of a histogram refers to the set of occupied bins. There may be unoccupied bins scattered amongst the occupied ones, but these voids are not elements of the data cluster. The boundary surface of a 3-D data cluster can be visualized as a smooth, continuous surface enclosing the entire data cluster with minimum volume. When the data cluster consists of two or more distinctly separate regions, it is better to enclose each cluster region with its own
boundary surface. A more precise definition for the boundary surface is unnecessary at this point.

In general, it is desired to choose an operating point lying somewhere inside the 3-D histogram data cluster’s boundary surface. This ensures the boundary will expand outward in roughly all directions so that a much greater extent of the RGB color space becomes employed. The occupied bins lying on the boundary surface, namely those outermost from the operating point along the rays, will be relocated along the boundaries of the RGB cube. If an operating point outside the data cluster boundary is chosen, the expansion will not occur in all directions; significant sections of the color space will still lie outside the final boundary surface. In certain applications, this effect might be desired.

For several reasons, the centroid of the histogram’s data cluster is usually a good operating point choice. The centroid almost always lies inside the data cluster boundary and therefore will generally cause omnidirectional expansion of the boundary surface. In the directions of the outermost occupied bins, the boundary surface expands all the way to the color space boundary itself. The centroid also tends to give the best freedom from directional bias in the overall set of 1-D histograms computed. Resulting skew in the final 3-D histogram will tend to be lower compared to that caused by an operating point lying some distance from the centroid. However, there is no guarantee that explosion about any particular operating point will result in a 3-D histogram that is unbiased or uniform.

If the operating point happens to correspond to an unoccupied bin in the 3-D histogram, an interesting effect can occur. Each of the rays will encounter its first occupied bin at some point other than the operating point. If histogram equalization is applied along the ray, this first bin will shift to the beginning of the available range.
That position maps back to the operating point's location in 3-space. The innermost occupied bin, namely the bin closest to the operating point in a given direction, moves to that operating point position. Therefore, many such bins can merge into a single bin at the operating point position. Figure 3.4 illustrates this effect. This merging of bins may cause widely separate colors to map to a single new color, causing some loss of information and the creation of artifacts in the output image. It is possible to avoid this merging effect if a method is used that does not shift the innermost occupied bin to the end of range. One possible such method is the simple linear scaling described in Section 3.3.4. Another way to avoid merging at the operating point is to always choose an occupied bin as the operating point. If the centroid happens to be an unoccupied bin, one solution is simply to locate the nearest occupied one and proceed from that position.

Figure 3.4 Here, the operating point corresponds to an unoccupied bin. Explosion using histogram equalization along the rays will cause the innermost bins to merge at the operating point.
An operating point may sometimes be desirable far from the centroid position. A multimodal or clustered histogram can receive enhancement tailored for a particular cluster by proper selection of the explosion radiant position. By exploding about a point inside one cluster, that cluster becomes greatly exaggerated because its bins become widely spread in all directions through the color space. The other clusters tend to be flung outward, each in its own general direction, where they tend to become confined to relatively small regions near the color space boundaries. Similarly, exploding about a point between two clusters can help separate them. Figure 3.5 illustrates these effects.
Figure 3.5 (a) Bimodal histogram with clusters A and B. (b) Result of explosion using centroid as operating point. (c) Original bimodal histogram with the operating point placed inside cluster A. (d) Explosion output showing widely scattered A and relatively suppressed B regions.

The operating point may also be selected so as to preserve certain desired characteristics of the original image. For instance, recall from Chapter 2 that several hue-based color spaces, such as HSL, define hue as an angle about the gray line in a plane perpendicular to that line. Any radial bin movement relative to the gray line will not affect the hue as long as the gray line itself is not crossed. Consequently, hue can be preserved completely by choosing an explosion operating point that lies on the gray line, making all points along a given ray have the same position angle and, therefore, the same hue. If this is done, the enhancement involves saturation and brightness
only. Note that the precise nature of the saturation and brightness modification still depends on the operating point position on the gray line as well as on the shape of the original histogram.

However, if the operating point does not lie on the gray line, hue can very well be altered. The amount of such hue modification depends on the distance from the operating point to the gray line as well as the size and shape of the original histogram. Only slight hue changes will occur for the vast majority of bins if the operating point is close to the gray line and the histogram itself is generally well distributed. This is evident from the fact that most rays would tend to deviate from the surfaces of constant hue by small angles. Also, a well distributed histogram does not undergo an extremely large adjustment when the explosion algorithm is applied. A tightly packed histogram, however, can be adjusted by a much greater amount and might suffer more significant changes in hue compared to the well distributed case. If the operating point lies some distance from the gray line, then the angles between the surfaces of constant hue and the rays can be rather large. In this situation, only a minor adjustment along the rays can give rise to a rather significant hue shift.

If the operating point lies outside the data cluster, the explosion will not occur in all directions. This is also true if that point is actually on the data cluster boundary itself. In both of these cases, the operating point is not completely surrounded by occupied bins. Since bin movement is constrained along rays emanating from the operating point, no bins are available over a wide angular range to fill the empty space there. A more encompassing statement of this property is that the explosion method does not operate uniformly in all directions; it tailors the modification to the histogram’s characteristics in the specific directions of occupied bins relative to the operating point. Therefore, an operating point chosen just inside the data cluster
boundary might easily lead to a severe adjustment in one direction and a mild one in the opposite direction.

Although histogram explosion preserves the ordering of the histogram bins along each of the operating point vectors, it does not guarantee that the 3-D bin order is preserved. Two originally adjacent bins lying in different directions from the operating point will be governed by two entirely different 1-D histograms along their separate rays. Therefore, these bins might easily be shifted by significantly different amounts in the process, causing them to become widely separated. Meanwhile, an originally relatively distant third bin might settle between the first two.

3.3.3 Shifting the Histogram Before Explosion

Assuming the operating point is the centroid, a variation on the explosion method involves shifting the entire original histogram so the centroid moves to the center of the RGB color cube, where it is then fixed. This is analogous to the case in 1-D, one-sided histogram equalization, where shifting the median bin to the center of the gray level range is necessary to prevent unequal densities from forming on either side. As shown in Figure 3.6, centering in RGB space helps minimize the tendency of an off-center centroid to impose directional bias in the output histogram's density. This will aid in obtaining a better distribution of bins in the output histogram. However, centering the centroid in the color space does not remove the directional bias caused by the variation of cumulative bin counts among the individual rays. Instead, the effect is more global in that any given solid angle originating at the center of the RGB cube subtends the same volume of color space as its directionally opposite counterpart. When the angle is small, there is likely to be a poor balance between the number of bins on opposite sides of the central operating point. A large angle will
tend to exhibit relatively good balance. This is a manifestation of the discreteness problem, in which a large number of discrete data points better approximates a continuous process than does a small number.

![Diagram](image)

Figure 3.6 Two dimensional illustration of the effect of operating point location on the local density achieved. The off-center operating point has unequal areas A and B available. The centered case’s areas C and D are equal. Each area, combined with the total bin count it contains, influences the local densities achieved. If each angular wedge contains the same total bin count, densities in C and D will be equivalent while the density in A is higher than in B.

Shifting the entire histogram to center the centroid is actually quite easy and quick to accomplish. The obvious approach would be to pass through the entire histogram table to shift every color by the necessary fixed amount. The problem with this is the possibility of exceeding the color space limits, perhaps causing wrap-around or truncation errors if the data representation is susceptible to these effects. Special handling can circumvent these difficulties, but involves extra computation. A better alternative is to leave the histogram table unchanged, but accomplish the shift during the explosion process itself. The ray histograms are constructed in the usual manner,
but must be modified based on color space boundaries that have shifted. This is done by applying the appropriate shift to the ray histogram’s boundary point \( b_i \). By doing this, each ray’s histogram is adjusted according to the amount of room it will have after the shift takes place. Finally, the modified bin positions along each ray are mapped back to 3-space based on the shifted operating point position instead of the original one. This scheme for accomplishing the shift is more efficient because it completely eliminates the step of changing the histogram table’s color values.

As discussed in the preceding section, hue in the HSL model is altered if a bin changes angle about the gray line. Translation of the entire histogram can easily alter the hues of most bins unless movement occurs strictly parallel to the gray line. That is the only kind of movement that preserves the original hue angle of every bin. Movement of the entire histogram data cluster in another direction gives rise to some interesting hue effects. Only those few bins whose movement vector happens to intersect the gray line will not suffer any hue change unless they actually cross the gray line. Those bins that do cross the gray line have their hues complemented. Bins whose movement vectors do not intersect the gray line will suffer hue alteration according to the amount of shift and the path of travel relative to the gray line. The greater the shift, the more bins that are likely to undergo drastic hue shifts. In many cases, most bins will usually suffer only moderate amounts of hue alteration.

3.3.4 Modification Along the Rays

The heart of the explosion process is the relocation of histogram bins along the operating point rays. Many possible methods can be used to accomplish the job. In fact, virtually any method used for histogram modification of monochrome images can be applied to the ray histograms. Of course, the final results will depend on the
particular method used, so some awareness of their characteristics is important. Figure 3.7 defines the parameters involved in ray histogram modification. The purpose of computing the histogram modification along a particular ray is to determine the new position of the bin of interest, namely the occupied bin which defines that ray.

![Diagram of parameters](image)

Figure 3.7 Illustration of parameters involved in 1-D histogram modification along the ith ray. Located at the origin is \( p \), the operating point position. The bin of interest is \( c_i \), \( r_i \) is the outermost occupied bin, and \( b_i \) is the position of the boundary. \( f(k) \) is the count of each bin.

The linear stretch and histogram equalization are two of the most commonly encountered 1-D histogram methods for enhancing monochromatic images. One of the simplest forms of the linear stretch entails scaling the histogram so the highest-indexed occupied bin moves to the highest possible index position. Using this method, the new position of the bin is computed as follows:

\[
    c'_i = \frac{b_i c_i}{r_i}, \quad r_i \neq 0
\]  

(3.1)

Notice that this linear stretch does not allow any bin to move toward the operating point.
Histogram equalization consists of scaling the bins according to their cumulative bin counts instead of their bin indices. The new position of a particular bin having undergone histogram equalization can be computed as:

\[ c_i' = \frac{b_i \sum_{k=0}^{c_i} f(k)}{\sum_{k=0}^{n_i} f(k)} , \quad \sum_{k=0}^{n_i} f(k) > 0 \]  \hspace{1cm} (3.2)

Some of the properties of histogram explosion using equalized ray histograms have already been discussed, including how the innermost bins can merge together at the operating point color. That means it is better to apply histogram equalization only when the operating point is placed at the location of an occupied bin.

The fact that all of the ray histograms are equalized does not mean that the output histogram is equalized in a three dimensional sense. The density will tend to be greatest in the vicinity of the operating point, as shown in Fig. 3.8, because that is the focus of the rays. The separation of the rays increases with distance from the explosion point, thus decreasing the local density possible.
Figure 3.8 Illustration of explosion rays about an operating point. Each ray’s histogram, shown as a series of dots, is uniformly dense following histogram equalization. The 2-D histogram, which is clearly not uniform, shows the greatest density near the operating point.

For histogram equalization to operate along a ray, it is necessary to derive the complete ray histogram from the original RGB histogram table. Exactly how to construct this histogram is another topic ripe with options. A ray from the operating point through an occupied bin will quite often pass between the centers of adjacent occupied bins instead of directly intersecting one of the centers. Some sort of interpolation scheme is clearly needed. One method that works well is simple first-order interpolation, where the counts of the nearest bins to the ray are weighted according to their proximity to it.
Corners of squares are the centers of the nearest bins in the planes parallel to the x-y plane.

Figure 3.9 illustrates the situation for first order interpolation between the counts of the four nearest bins:

\[ f(e) = \beta \left[ (1 - \alpha) f(a) + \alpha f(b) \right] + (1 - \beta) \left[ (1 - \alpha) f(c) + \alpha f(d) \right] \]  

More elaborate schemes are possible, including higher orders of interpolation and more sophisticated filter functions, but the differences would tend to be subtle and probably not worth the extra computational cost. To simplify interpolation, a zeroth order approach can be used:

\[
f(e) = \begin{cases} 
  f(a), & \alpha < 0.5, \beta \geq 0.5 \\
  f(b), & \alpha \geq 0.5, \beta \geq 0.5 \\
  f(c), & \alpha < 0.5, \beta < 0.5 \\
  f(d), & \alpha \geq 0.5, \beta < 0.5 
\end{cases}
\]  

This way, the interpolation simply becomes a matter of selecting the bin nearest the ray for each ray index position. Not only does this render the interpolation calculations unnecessary, but only one of the bin counts must be determined rather than the four needed for first-order.
The linear stretching shown in Equation 3.1 does not require the derivation of a complete ray histogram. Each ray serves to govern the modification of a single bin of interest, so merely the information necessary to compute that bin's shift need be obtained. For linear stretching, only the outermost occupied bin serves to define the stretch parameters. Consequently, the locations of \( c_i \), \( r_i \), and \( b_i \) are the only required data points to be derived from the RGB histogram table. This type of linear stretching along each ray involves far less work than for histogram equalization, which demands that the entire ray histogram be determined.

More elaborate schemes might be useful in tailoring the explosion process in a desired way. As an example, recall that the explosion process tends to generate an output 3-D histogram whose local density decreases with distance from the operating point. If a 3-D histogram of more uniform density is desired, perhaps the ray histograms could be weighted to compensate for this effect. Application of a weighting function \( w(k) \) would convert the original bin counts \( f(k) \) into effective bin counts \( [w(k)f(k)] \), which would then be submitted to the desired ray processing. When histogram equalization is applied along the rays, the weighting function \( w(k) \) should be maximum at \( k=0 \) and decrease monotonically with increasing \( k \). Bins near the operating point would receive the greatest boost in their effective bin counts. Equalization of the weighted histogram would force these inner bins to space more widely than if unweighted, reducing their tendency to cluster near the operating point. An alternate approach might be to apply compensation as part of the interpolation process, with the interpolation region size becoming a function of distance from the explosion point. With properly chosen interpolation weighting, such a scheme may also help in approaching the sort of histogram desired.
Still, a truly uniform multivariate density is not guaranteed because of the variables of explosion point location, the ray modification function applied, and the original histogram itself. For approximating a 3-D uniform distribution, the weighting function would have to be based on an assumed original histogram. The greater the deviation of the input histogram from this assumed shape, the greater would be the disparity between the actual output histogram and the intended one. To better approximate a desired output histogram, it is necessary to tailor the weighting function to the specific histogram to be processed. This is highly likely to depend upon the ray directions, demanding each ray’s weighting to be determined specifically. The technique would be complicated and computationally tedious, but should be capable of approximating the desired histogram shape.

It should be clear at this point that explosion is not intended to achieve 3-D uniformity in the output histogram. Explosion operates upon a series of local histograms; the global characteristics of its output are consequences of these local modifications. There is certainly no guarantee that the histogram density will not be skewed in some manner. In fact, it is quite likely to exhibit some bias in a three dimensional sense. Although explosion-processed images more fully utilize the color space than do the unprocessed ones, the remaining skew might be judged as allowing room for even better color space use. One possible improvement in the use of this space might be the case if the histogram were uniformly dense in a three dimensional manner. This will be discussed in more detail in Chapter 5.

3.4. Computational Issues

One very important attribute of the explosion method is that the amount of computation is roughly proportional to the number of unique color values present in
the image. Obviously, each occupied bin demands the computation of a histogram modification along its own ray. However, the more occupied bins there are, the more computation is typically needed to determine each ray's histogram and the color alteration it governs. Therefore, a doubling of the number of bins will likely cause more than a doubling of computational load.

3.4.1. Computing the RGB Histogram Table

Before any version of the histogram explosion process can be applied, it is necessary to compute the three dimensional histogram from the original color image. This 3-D histogram can be constructed in a way quite similar to that of the 1-D histogram of a monochrome image. Recall that for a monochromatic image whose pixels have gray levels in the interval [0, L-1], the histogram can be constructed by counting the number of pixels for each of the L possible gray levels. The result is a table of gray level values, each with an associated pixel count. In three dimensional RGB color space, the situation is essentially a straightforward extension of this idea.

The realities of determining a three dimensional RGB histogram are often somewhat more complicated than may seem at first. If each of the red, green, and blue components has a range of gray levels in [0, L-1], then the total number of possible color values is $L^3$. Therefore, a simple table of $L^3$ color entries, each containing a color value and its count, may serve as the color histogram. A simple array with the color value acting as an index can provide extremely fast lookup of bin count data. Building the histogram array is as easy as initially zeroing all bin counts, then traversing the image while incrementing the count for every encountered color value. Unfortunately, this implementation can readily become impractical for seemingly reasonable values of L. For example if $L=256$, then the histogram of a monochrome
image needs only 256 distinct bins, but a color image's histogram requires 16,777,216 bins. Assuming each bin count requires a 4-byte word of computer memory, the color histogram demands 64 megabytes of RAM, which is easily more memory than is usually available on many modern computers. Clearly, another approach is needed.

Consider now the total number of pixels in the image compared with the total number of possible colors. If L is large enough, the total number of pixels is much less than the total number of possible colors. This is certainly true for a 512 x 512 image with L=256, where $2^{18}$ pixels is far less than the palette of $2^{24}$ available colors. If every pixel would happen to contain a unique color value, the vast majority of the palette colors would still remain unrepresented in the image. It would be an extremely inefficient use of memory to allocate space for the entire range of colors when only a small fraction of them are actually used.

A better approach is to allocate memory only for those colors truly present in the image. The histogram table then consists of a matrix of color values and corresponding bin counts. This new structure is far more memory efficient, but complicates the job of retrieving the bin count data from the completed table. The explosion algorithm performs a very large number of searches for the bin counts of colors, so it is important that the lookup process be efficient and rapid. If the color values present in the table were not arranged in some order, a painfully slow search process would be needed to locate a specific color. For B unique colors in the table, such a method must search through an average of B/2 table entries to locate the desired one. The worst case would require searching all B entries. Let M be the number of histogram table searches required during the construction of a single ray in RGB space. The value of M depends on the nature of the interpolation employed. If zeroth-order interpolation is used, one search is needed for each unit step along the
major axis, which is the component axis lying nearest to the ray's direction. For first-order interpolation, each major axis step requires 4 searches. Since the average ray through the RGB space of a 24-bit color image has approximately 128 steps along the major axis, then \( M \approx 128 \) for zeroth-order interpolation and \( M \approx 512 \) for first-order interpolation. Thus, the task of searching an unsorted histogram table is \( O(MB^2) \).

Assuming the table's color values are present in ascending order, a binary search for the desired color value becomes possible, requiring the search through a maximum of only \( \lceil \log_2(B) \rceil \) table entries. The search task in this case is \( O(MB \lceil \log_2(B) \rceil) \). Regardless of whether the histogram table is arranged in order, the improvement in memory efficiency comes at the cost of additional computation required for lookup. Clearly, however, the sorted version of the histogram table allows for much more efficient searching than does the unsorted one.

Constructing the ordered histogram table should also be accomplished efficiently. The three color component values can be combined into a single number, helping to make subsequent sorting and searching more efficient than if the three values remain separate numbers. For 24 bits of color, a convenient definition for the color value utilizes a 4-byte integer in the format \( 0\text{RGB} \), where \( R, G, \) and \( B \) denote the bytes containing the red, green, and blue values while the \( 0 \) byte contains all zeroes. Other color value formats can be equally valid as long as the search process is consistent with the format. Ideally, the color value format should be chosen to minimize computation when converting to and from the specific image data format encountered.

The recommended method of building the table begins with an efficient \textit{in situ} sort, such as quicksort, performed on the image data so the pixels are arranged into ascending row-column order of color value throughout the image. Then the sorted
image is traversed in row-column order to determine the number of unique colors present. Since the pixels are now in ascending order, this is easily determined by simply adding one to the number of color changes encountered during this traversal. Once the number of unique colors is known, the required amount of memory for containing the histogram table can be allocated and zeroed.

The histogram table itself is constructed during a second row-column traversal of the sorted image. The color of the first pixel is placed into the first table position and its bin count is initialized to one. This color is now considered the active one. As image traversal progresses, the color value of each encountered pixel is compared to the active color. If they match, the active bin count is incremented; otherwise, this new color is entered into the next table position, its bin count is initialized to one, and this color becomes the active color. Once the traversal finishes, the sorted histogram table is complete and ready for use. The computational complexity of the method is analyzed in Figure 3.10(a).
for each pixel:

\[ N \log(N) \]

allocate memory for
B colors, counts, and
new colors; zero all
counts

allocate enough
memory for table
colors, counts, and
new colors; zero all
colors present
in table?

B*Blog(B)

increment count
for color found

N

add color entry to
table; quicksort
table by color value

N

complexity:
\[ O[N \log(N)] \]

complexity:
\[ O[NB + B^2 \log(B)] \]

Figure 3.10 Comparison of two methods for constructing the histogram table. Method (a) requires first sorting the image, but is then quite efficient. Method (b) must sort the histogram table each time a new color value is added.

Why not merely sort the histogram table instead of the entire image? Consider the case of building the table directly from the original unsorted image, as shown in Figure 3.10(b). As each new pixel is examined during traversal of the unsorted image, its color must be sought in the table. If the color is not already present in the table, it must be incorporated as a new table entry. Since it is much more efficient to search a sorted histogram table rather than an unsorted one, the new color value must be placed into the table so that this order is maintained to facilitate future searching. This requires sorting the histogram table every time a new color value is added. By instead using the sorted image, the necessity for searching and sorting the histogram table during its construction completely vanishes. If an image contains relatively few
unique colors, however, the computational load of sorting the entire image can be
greater than that of searching and sorting the relatively small histogram table. Almost
always, 24-bit color images tend to have enough occupied bins to benefit from the
sorted image method of histogram table construction.

Increased speed in searching the histogram table may be purchased at the cost of
allocating additional memory, allowing the binary search to be upgraded to a binary
tree search. The tree search requires the assignment of extra memory to each color in
the table for the purpose of containing pointers to the next branches of the tree, one
pointer to the upper branch and one to the lower. The binary search described above
involves comparing a selected color in the table to the color sought. If they differ, the
next entry to be tested is found by computing its position in the table, halving the
search range at each step. The tree search avoids this branch address calculation
during searching because these addresses are precomputed and present in the table.
All that is necessary is to select the proper pointer based on the comparison results.
These pointers would ideally be computed after the table is constructed and before
searching is attempted. Assuming the color values, bin counts, and branch pointers
each require 4-byte integer representation, the binary tree search algorithm must
allocate twice the memory of the simple binary search.

3.4.2. Computing the Explosion

In order to describe the details of the explosion process in this section, the
following assumptions are made:

1. The operating point is the histogram's centroid.

2. The ray histograms are built using first order interpolation.

3. Explosion is determined by equalizing the ray histograms.
The centroid color is easily determined by independently calculating the averages of the red, green, and blue component histograms:

\[
\mu_R = \frac{1}{N} \sum_{i=0}^{L-1} R_i \, n_R(i), \quad \mu_G = \frac{1}{N} \sum_{i=0}^{L-1} G_i \, n_G(i), \quad \mu_B = \frac{1}{N} \sum_{i=0}^{L-1} B_i \, n_B(i) \quad (3.5)
\]

In the Eq. 3.5, \( L \) is the number of gray levels available for each color component, \( R_i \) is the gray level value of the \( i \)th red bin, \( N \) is the total number of pixels in the image, and \( n_R(i) \) is the number of pixels having the red component value \( R_i \). The centroid color is simply \((\mu_R, \mu_G, \mu_B)\).

The histogram table must be constructed in the manner described in the preceding section. After a quicksort of the entire image, the number of different colors is determined. Enough memory is then allocated so that each color value has associated with it a count value and a modified color value. In this work, each color value and count value was assigned a full 32-bit word of memory, although only 24 bits were needed to represent the color values themselves. Next, the color and count values are entered into the table from the sorted image. Once the histogram table is complete, the sorted image data is no longer needed and may be deallocated from memory. Image data will not be accessed again until after all of the modified color values have been computed, meaning that the transformation table is finished.

Next, each color in the table must be adjusted according to the equalized ray histogram containing it. The ray histogram can be built as follows. First, compute \( \Delta R, \Delta G, \text{ and } \Delta B \) from the centroid to the subject color, where \( \Delta R = R_p - R_i \). The component of greatest magnitude becomes the major axis and the others are minor axes. For example, the G component would be the major axis if \( |\Delta G| > |\Delta R| \) and \( |\Delta G| > |\Delta B| \). The ratios of the minor deltas to the major delta gives the relative slopes in
the two minor directions. These slopes always lie on the interval [0,1]. Figure 3.11 contains a flowchart of the entire histogram explosion algorithm.

Figure 3.11 Flowchart of histogram explosion algorithm. The left side constructs the 1-D histogram along each ray. The right side determines the modification of the ray histogram, builds the color modification table, and modifies the image accordingly. First order interpolation is used during building of the ray histogram. The ray modification is histogram equalization.

The task now is to begin at the operating point and step one unit at a time along the major axis while incrementing the minor axes by their fractional slope amounts. At each step, interpolation must be computed between the bin counts of the nearest
four colors at this major axis position. The bin counts of these four colors must be retrieved from the histogram table via the binary search described earlier. If a color is not present in the table, its bin count is zero. Three bilinear interpolation calculations (see Eq. 3.3) are required to calculate the bin count at the ray position. Finally, this bin count is entered into the ray's histogram table. When the edge of the color space is reached, the ray histogram is complete.

Before it can be equalized, the ray histogram must be confined to a range that corresponds to the distance between the centroid and the edge of the color space. This parameter has already been computed during construction of the ray histogram; it is the last valid major-axis step value before escaping the loop. Once confined to this range, the ray histogram can be equalized in the way described earlier in this chapter. Afterward, the new ray position of the subject bin is mapped to 3-space by displacing its index from the operating point along the major and minor axes, with the two minor axis displacements scaled by their slopes. This modified color becomes paired with the original one in the color transformation table.

The histogram explosion algorithm is dominated by the task of searching the histogram table to determine the bin counts when building the ray histograms. If there are $M$ such searches along each ray, then the complexity is $O(MB^2\log_2(B))$.

3.4.3. Possible Speed Improvements

Substituting zeroth order interpolation for the first order method improves speed significantly because interpolation is done in the innermost loop of the explosion algorithm. For 24-bit color images used in this work, there are an average of 128 calls of the interpolation routine executed for each occupied bin. This is based on the average ray's major axis component length of 128 units. A typical 512 x 512 color
image might contain about 100,000 different color values, leading to a total of
12,800,000 calls to the interpolation routine. A first order interpolator must perform
four lookups of the histogram table to obtain the bin counts for the four closest bins,
then must compute the interpolated bin count. The zeroth order interpolator merely
decides which of the nearest bins is closest, requiring only the lookup of a single bin
count. Each binary search of the sorted histogram table is $O(\log_2(B))$, where $B$ is the
number of unique color values in the original image. The zeroth-order interpolator
therefore executes several times faster than the first-order interpolator. Since the
interpolation process lies in the innermost loop of the explosion algorithm, it has great
influence in reducing overall execution time.

Performing a form of linear stretch instead of histogram equalization along the
rays is another way to improve speed. This way, it is not necessary to accumulate the
bin counts along the ray histogram. All that is needed is to locate the outermost
occupied bin along the ray. With that information and the ray position of the subject
bin, one can compute the scale factor that accomplishes the stretch operation. Also,
the full ray histogram need not be built. One possible method would begin
interpolation at the subject bin position and continue outward along the ray from there.
For the linear stretch, the ray bins between the origin and the subject bin are
completely irrelevant and may be skipped entirely. The potential computation load of
this scheme is significantly reduced from than that of histogram equalization because
fewer lookups of the image’s histogram table are necessary as well as the fact that
count accumulation is completely avoided. The results obtained can differ noticeably
from those derived from equalization of the ray histograms, but should be equally
useful and valid.
The proposed explosion technique is based on the idea that the position of each occupied bin is shifted along its own unique ray, requiring that the number of 1-D histogram operations equals the number of separate color values in the image. This is not entirely necessary, however, because some of the rays might perfectly intersect two or more such occupied bin positions. Some efficiency might be gained by eliminating the redundant ray histogram work and allowing each ray to govern all of the bins which it precisely pierces. In order to accomplish this, though, it is necessary to determine exactly which bins each ray should govern. The extra searching and decision process to decide this reduces the potential for overall gain in efficiency. Such an approach might even cause a net decrease in performance because relatively few rays will exactly intersect more than one occupied bin, but every ray would need to be searched for multiple bin intersections. If this approach were used, a very efficient scheme for locating the intersected bins would be required.

Perhaps a compromise could be effective. Instead of searching every ray for all bins precisely intersected, one might search only those rays lying in certain directions likely to intersect the centers of occupied bins. For example, rays parallel to the R, G, or B coordinate axes are very likely to intersect many occupied bins. Rays at 45° to pairs of axes should also be good candidates. In general, the best candidates for multiple bin intersections would be those rays described by the vector \([AR, AG, AB]\) having the following characteristics:

1. \(AR, AG, AB \in \{\text{integers}\}\)
2. \(0 < |AR| + |AG| + |AB| \leq t\)

where \(t\) is a moderately small positive integer (e.g., 10 or 20). For example, the ray \((AR, AG, AB) = (1,0,1)\) would be very likely to intersect several occupied bins while the ray \((14,-9,23)\) would not. If only the rays meeting such a criterion were to be
checked for multiple bin intersections, much of the potential for speed gain might be achievable with minimal extra computation.

The structure of the histogram explosion procedure allows it to be a good candidate for parallel processing. After the histogram table is derived, the computation for the set of color values can be accomplished mutually independently. If the job is divided properly among P processors such that inter-processor communication overhead does not limit performance, there is the potential for nearly a factor of P in speed improvement. Each processor would require access to the full histogram table. This could be either in the form of a single table accessible by all processors or providing each processor with its own local copy. The results would have to be collected into a common structure before the original image could be modified.
Chapter 4

Results

4.1 Software

During the course of this research, the histogram explosion algorithm was realized in the form of a C language computer program which was run on a Sun SparcStation 1+ computer. The program computes the 3-D histogram table from an input 24-bit color image, performs the explosion calculations to generate the transformation table, and then transforms the image accordingly. Several options are included in the software to allow experimentation with various approaches to histogram explosion. Output images are stored in disk files for later viewing.

The program allows the choice of explosion operating point as either the histogram's centroid, which the computer calculates from the input image data, or a user-specified RGB color value. If the operating point is chosen at the centroid, one can also specify whether the entire histogram is to be shifted so the operating point becomes centered in RGB space before explosion begins. For the explosion process itself, one may choose between first-order and zeroth-order interpolation methods for computing bin counts along the rays. Both the linear stretch approach and the histogram equalization method are available for governing the explosion processing along each ray.

In its current implementation, this computer program calculates the 3-D histogram from an image and holds the histogram data in main memory for use only
during program execution, without providing a means for storing this histogram for later use. Each execution therefore demands the significant computational task of determining the histogram table from the input image. If one wishes to perform a series of different explosion processes upon a single input image, the histogram table must be calculated anew at each execution. However, this histogram table depends only upon the original image data and so remains independent of any explosion processing, necessitating that the histogram calculation be performed only once. Therefore, the program could be improved upon by splitting it into two parts. The first part would calculate the histogram table and store its data in a disk file. The second would retrieve the stored histogram data and perform the explosion. Alternatively, a single program could first check the disk for the presence of a precomputed histogram table for the image. The histogram table would need to be computed only if it could not be obtained from the disk. Where possible, this approach would avoid the rather significant computation load of deriving the histogram table at every execution.

4.2 Images

The series of photographs in Figures 4.1 through 4.10 demonstrate the effectiveness of the histogram explosion method and compare it with some of the techniques discussed in Chapter 2. Several variations of the explosion process are also shown. To illustrate the effect of processing on the 3-D histogram distribution, Figures 4.1 through 4.4 each includes a perspective view of the RGB histogram plots to accompany the photographs. In these plots, the original data has been requantized to fewer bits in order to reduce the size of the data set that must be plotted.

Comparing the histogram plots in Figs. 4.1 through 4.4, it is clear that each enhancement method shown helps to spread the histogram data cluster in RGB space.
The only method that utilizes approximately the full extent of the space is histogram explosion; the other methods generally leave significant vacant volumes near the color space boundaries. Because it causes the histogram to exploit the full RGB space, one would expect histogram explosion to produce results that are subjectively more "colorful" than the other enhancement procedures. Experiments with many color images, including those shown in the photographs here, uphold this expectation. Although possessing greater brightness contrast than the original of Fig. 4.1, the output image in Fig. 4.7 contains suppressed chromatic components.

The results of a single image having been subjected to three variations of explosion are shown in Figs. 4.4 through 4.6. A comparison of the ray linear stretch versus the ray histogram equalization can be seen in Figs. 4.4 and 4.5 respectively. One major difference is the additional noise evident in the more uniform areas of Fig. 4.5, visible as grayish pixels in the blue waters of San Francisco Bay. The main cause for this color noise lies in the operation of ray histogram equalization, which can shift some of the ray's occupied bins toward the operating point. If the operating point lies at a color position that is not present in the original image, the innermost occupied bin along each ray is relocated to the operating point position. The consequent merging of some bins at the operating point will cause artifacts visible as noise. The convergence of bins in the vicinity of the operating point may also give rise to such artifacts if bin merging occurs. In contrast, the ray linear stretch function never moves bins toward the operating point, thus avoiding the chance of bin merging taking place in that vicinity. Therefore, linear stretching avoids the generation of such noisy artifacts. The linear stretch method is generally recommended over histogram equalization for this reason, in addition to its simpler and more rapid computation.
The effect of operating point position can be observed when comparing Figs. 4.5 and 4.6. The subjective visible difference is striking. The image in Fig. 4.6 has undergone explosion from the center of the color space instead of the data cluster centroid as in Fig. 4.5. Because the operating point in Fig. 4.6 lies on the gray line, original hues are preserved.

Experience with the computer program operating on a variety of images has shown that the zeroth-order and first-order interpolation methods produce output images that are perceptually virtually identical. This is why a comparison between the interpolation methods is not present in the set of photographs. Since zeroth-order is by far the more computationally simple of the two interpolation methods, its faster execution generally recommends it as the best choice. The differences would probably become noticeable only in cases of images having relatively few occupied bins separated by significant, empty regions of the color space.

A second image is shown in Figs. 4.8 through 4.10. This example shows how histogram explosion can perform enhancement for human visual use if applied carefully. In this case, the original image has relatively suppressed chromatic components. By exploding about the centroid, which lies quite close to the gray line, the image becomes much more colorful without noticeable hue shift. In contrast, Fig. 4.9 shows that independently equalizing the R, G, and B component histograms is far less effective than explosion. If one desires an enhancement of this image so that brightness is generally reduced without hue being changed, an operating point closer to the white end of the gray line will accomplish this. Histogram explosion, therefore, gives far greater flexibility in the character of the enhancement process.
Figure 4.1 (a). Image SFO, a false color composite of a multispectral satellite image of San Francisco Bay. Size is 512 x 512 pixels at 24 bits/pixel. The centroid of the histogram data cluster lies at (R,G,B) = (119,109,135).

Figure 4.1 (b). Perspective view of the RGB histogram of 4.1(a). For data reduction and better visibility of these plots, each component’s data has been requantized from the original 256 levels down to 16. Dots show occupied bins.
Figure 4.2 (a). SFO after independent histogram equalization of the R, G, and B component images. Much of the ocean has become very dark during the general increase in contrast.

Figure 4.2 (b). RGB histogram of Fig. 4.2 (a). Although there is much greater use of the color space compared to Fig. 4.1, poor use has been made of the regions near the space boundaries. Some hue alteration has occurred, but the character of this alteration depends solely upon the original histogram.
Figure 4.3 (a). SFO after decorrelation processing. Hues have shifted somewhat, due to the centering of the decorrelated histogram in the color space.

Figure 4.3 (b). Histogram of 4.3 (a). The equalized component variances have resulted in greater use of the RGB space. Further stretching is now possible by increasing the component variances equally, but only until the first occupied bin reaches a boundary.
Figure 4.4 (a). Histogram explosion of SFO about its centroid. Linear stretching has been applied along the rays.

Figure 4.4 (b). Histogram of Fig. 4.4 (a). Note the occupied bins utilize nearly the full extent of the color space.
Figure 4.5. Explosion similar to Fig. 4.4, but substituting histogram equalization for the linear stretch along the rays. Note the colored noise visible as grayish pixels in the bay waters. This noise is caused by the innermost occupied bins being merged at the operating point color.
Figure 4.6. Explosion similar to Fig. 4.5, but the operating point is now (R,G,B) = (128,128,128), the center of the color space. Original hues are therefore unaltered.
Figure 4.7. SFO enhanced by equalizing the histogram of the Y (brightness) component and recombining with the unaltered I and Q (chromatic) components. The chromatic components have been scaled back in order to prevent any occupied bin from moving outside the RGB space boundaries.
Figure 4.8. Kenya, a 1000 x 971 pixel, 24 bits/pixel true color image. The centroid lies at \((R,G,B) = (186,189,175)\).
Figure 4.9. Kenya after the R, G, and B components have been independently histogram equalized.
Figure 4.10. Result of histogram explosion about the centroid. Much greater color contrast is evident here than in Fig. 4.9. Any hue alteration is subtle because the operating point lies quite close to the gray line.
Chapter 5

Multidimensional Histogram Equalization

5.1. Goals

The conventional one dimensional method of histogram equalization forces a given histogram to become as uniform as possible. Its goal is essentially to make each of the gray levels equally likely for any pixel in the image, given the constraint that all pixels having a specific original gray level must share a common final gray level. One of the main benefits of histogram equalization has proven to be its ability to enhance contrast without operator supervision.

How might this idea be extended to the three dimensions of color images? Instead of applying the histogram equalization technique to each of the three component images independently, a true three dimensional histogram equalization method would need to manipulate all three components simultaneously. All of the occupied histogram bins would be moved in RGB space in a manner that yields the most uniform 3-D histogram. The histogram's volume density would become as uniform as allowed by its discrete nature.

There is an extremely large number of ways that the bins might be rearranged in 3-space to obtain an approximately uniform volume distribution. Most of these arrangements greatly alter the original 3-D bin ordering, causing undesired artifacts and distortion. The best overall solution should be that which minimizes reordering and thus is most faithful to the original data. Minimizing the reordering of occupied
bins becomes another constraint upon any proposed multidimensional histogram
equalization method.

Since 3-D equalization would operate in a truly multivariate sense, the migration
path of a given bin in the color space would not be constrained to some preconceived
structure. This contrasts with the earlier described enhancement methods, all of which
impose specific paths which constrain bin movement. For example, recall that
histogram explosion constrains bin movement to rays emanating from an operating
point. Decorrelation allows bins to be scaled along each of the eigenvector directions.
Methods involving adjustment of brightness and saturation, such as Bockstein’s or
Gillespie’s, constrain bin movement to planes of constant hue. Equalization in three
dimensions would shift each bin in the direction dictated solely by its position and
weight relative to all the other bins in the histogram. Perceptual qualities such as hue
are not apt to be well preserved in the process.

Just as 1-D histogram equalization transforms a given continuous probability
density function to become uniform (see Appendix A), the 3-D equalization process
must transform an arbitrary 3-D probability density function into a uniform one. Also,
the centroid of such an equalized 3-D density function must always lie at the center of
the space. Because histograms are discrete, the effects of quantization demand that
such properties can only be approximated.

5.2 Iterative Equalization of Local Volume Density

5.2.1 Approach

The task of achieving a uniform histogram in a finite space while minimizing
the tendency to change bin order is analogous to the physical situation of an enclosure
containing a gas. Any gas pressure nonuniformity inside the enclosure gives rise to
forces tending to equalize the pressure distribution. Consider that the gaseous system can be modeled as a collection of discrete gaseous regions, called parcels, which might be imagined as balloons or soap bubbles containing various amounts of gas. Each parcel moves in response to the net force exerted by its neighbors, which are other parcels and, occasionally, the surfaces of the enclosure itself. Over time, the entire system of parcels redistributes itself in a uniformly dense manner throughout the enclosure, attaining a constant overall pressure distribution.

The original 3-D histogram of a color image consists of a set of occupied bins, each having a count value and a position within the confines of a particular color space. The set of occupied bins is analogous to a set of parcels confined within an enclosure matching the shape and size of the color space. The initial positions of the parcels in the enclosure correspond to the locations of their counterpart bins in the color space. Each bin or parcel exerts a repulsive force upon its immediate neighbors. The displacement vector of each bin or parcel can be computed from the sum of the forces acting upon it from the bins or parcels adjoining it. Simulation of the system over a series of time steps produces a distribution that approaches uniformity.

Assume that the ith bin having position $\vec{p}_i$ is influenced by each of its immediate neighbors $\vec{p}_j$ with a repulsive force $\vec{F}_{ij}$. The net force $\vec{F}_i$ exerted upon this bin is:

$$\vec{F}_i = \sum_{j \neq i} \vec{F}_{ij} = c \sum_{j \neq i} \frac{[h(\vec{p}_i) + h(\vec{p}_j)] (\vec{p}_i - \vec{p}_j)}{\|\vec{p}_i - \vec{p}_j\|^2}$$

(5.1)

In equation 5.1, the bin counts are denoted as the function $h(\cdot)$. The factor $c$ is an arbitrary positive constant. The force between two neighboring bins is thus proportional to the sum of their bin counts and inversely proportional to their
separation distance. This relationship applies only within a small neighborhood of the
subject bin.

The system is simulated over a sequence of time steps, with the movement of
each bin directed by the net force it receives at the most recent step. At every step, it
is necessary to compute the new force vectors for all bins in the histogram. As the
simulation progresses, local imbalances in density will tend to diminish. Gradually,
the global density of the system will approach uniformity. When the magnitudes of all
force vectors fall below a threshold value, iteration ceases. The desired transformation
needed to equalize the original 3-D histogram is then a simple mapping from each
bin’s initial position to its final position.

Notice that the bins must be constrained to remain inside the color space; any
bin that reaches a boundary cannot be allowed beyond it. This satisfies the goal of
obtaining a result having all bins inside the space. Without this boundary constraint,
the histogram bins would fly apart forever and leave virtually nothing within the
legitimate color space region.

What is the behavior of the overall system simulation? Except in rare cases, the
original histogram data cluster occupies only a fraction of full extent of the color
space. Therefore, the bins at the data cluster boundary will expand outward, allowing
the interior bins to also begin expansion from the cluster centroid. Whenever a bin
reaches a boundary surface, it becomes constrained to remain on that surface. The
outward expansion ceases when enough bins lie along the boundaries to repel further
outward movement of the others. As the distribution becomes closer to uniform, the
magnitudes of the net forces decrease. Eventually, none of the forces will exceed a
preset threshold value, calling for an end to processing. The discrete 3-D histogram is
now as uniform as it can be given the parameters of time step size and force threshold.
Since it is desired that 3-D bin order be preserved as much as possible, one should choose a fine enough step size to avoid having bins pass through each other from one step to the next. If the time step size is too coarse or the constant c too great, then the goal of minimal reordering will be compromised. For good overall simulation accuracy, the step size must be even smaller than that required to avoid interchange of position. Too small a step size demands extra computation to gain an unimportant degree of accuracy. Too coarse a step size can greatly alter the final outcome because of accumulated error. The greatest error tends to occur when two or more bins pass in close proximity. This situation demands a fine time step for accurate simulation. When adjacent bins are widely separated, their mutual interaction is far less, allowing a coarser time step to be used. So, choosing the proper step size is a critical part of this method because it involves a compromise between accuracy and speed of computation.

Notice that the speed or momentum of the bins is not involved in the computations. The physical system of gaseous parcels must not violate the principle of conservation of energy as so must preserve both potential and kinetic forms of energy. In contrast, the stepwise histogram modification has the goal of seeking the bin positions which comprise a uniform histogram, not in preserving the dynamics of the bin motions. By displacing the bins along their net force vectors at each time step, each bin proceeds "downhill" and causes a lowering in overall potential energy. If energy was conserved, it is likely that the momentum of some bins would carry them beyond their proper final positions and cause a change in the outcome. Conservation of energy would demand that a lowering of potential energy be countered by a rise in kinetic energy. As the system proceeds to lower its potential energy on the way toward a uniform arrangement, the presence of kinetic energy would prohibit the
system from settling down to a solution. Therefore, conservation of energy should not be a constraint on this method of seeking an equalized distribution.

In this iterative approach, the bins having relatively large counts would tend to shift in rather mildly varying paths while those having relatively small counts are likely to undergo large gyrations in position. It is quite possible that some of the "lightest" bins will follow complicated and circuitous paths as they are pushed aside by their "heavier" neighbors. On occasion, some of these light bins may come to rest on the other side of a group of massive bins, causing the order to be changed from the original. To help guard against this, it might be necessary to impose some sort of constraint on the ordering so that no bin is allowed to stray too far away from its original position relative to its neighbors. Whether this kind of constraint is needed and what form it might take are both unknown.

In certain degenerate cases, the iterative local equalization method may not produce an output histogram that is approximately equalized overall. One example of this effect is the case of an input histogram having most or all of its occupied bins already located on the boundary surfaces of the color space. Since the technique does not allow bins to escape the boundaries, it may become impossible to approach an equalized distribution throughout the interior of the space.

5.2.2 Computational Issues

A straightforward implementation of the method of iterative equalization of local volume density would involve choosing a conservatively small size for the time step. The histograms derived from most color images would probably require a large number of time steps to converge on a solution. For a given bin at a particular time
value, it is necessary to calculate the force acting upon it due to each of the bins in a properly chosen local neighborhood.

To analyze the computational complexity of the approach let:

1. \( B \) = number of occupied bins
2. \( T \) = number of time steps or iterations required
3. \( A \) = number of bins in the local neighborhood

As in histogram explosion, the computational task is dominated by the job of searching the histogram table to retrieve bin counts for all \( A \) bins in the local neighborhood. Therefore, the total computation is \( O(BTA \log_2(B)) \). Assuming a typical image contains 100,000 occupied bins, takes 100,000 time steps to converge to an acceptable stopping point, and the local neighborhood contains 100 bin positions, then \( 10^{12} \) histogram table searches are required. Clearly, the iterative local equalization method can involve a tremendous amount of computation. This makes it generally impractical and may confine it to the realm of interesting theoretical constructs.

It might be possible to make this iterative method more efficient, perhaps to the point of becoming viable. One approach might be to vary the step size according to the proximity of bins. Most bins would usually be separated enough to allow the use of a relatively coarse step size. When a bin begins to interact closely with its neighbors, perhaps its local region could be simulated more carefully until the close interaction diminishes. If done properly, this could preserve the desired accuracy while greatly improving execution speed, perhaps by an order of magnitude or so.

Some sort of hierarchical regional processing might also help improve efficiency. This could involve some consolidation of bins into "superbins" in a succession of greater and greater local regions in the histogram. The highest-level
superbins could be equalized first, then lower and lower levels might be equalized until finally the individual bins are treated in their local regions. While achieving some appreciable savings in calculation, such a hierarchical approach is not likely to always produce solutions similar to those of the direct method. Also, the potential exists for the introduction of histogram artifacts similar in size and shape to the hierarchical regions used. This effect would be caused by the tendency for bins to collect along the boundary surfaces of the hierarchical regions. It is unclear how visible or objectionable such artifacts might be in the output images.

5.3 Simulated Annealing

A more practical way to approximately determine the best equalized multidimensional histogram might use simulated annealing, discussed in Kirkpatrick [11]. In it, the individual bin positions are shifted by random amounts that have been scaled by a “temperature” function. An appropriately chosen cost function is computed for this new arrangement and compared against the cost function evaluated for the previous arrangement. If the cost function has decreased since the last iteration step, the new positions are accepted as an improvement and processing proceeds from that point. Otherwise, the new arrangement is rejected and another random shift is tested. As iteration proceeds, gradually the “temperature” factor is allowed to decrease, lessening the probability of large perturbations in the bin positions.

The cost function that would control the simulated annealing algorithm for seeking an equalized 3-D distribution is not well understood. Perhaps the concept of entropy could serve as the basis of a suitable cost function. Like the iterative local equalization approach, simulated annealing would demand a large number of iterations to achieve a close approximation to a uniform histogram. Simulated annealing may
have the potential for some savings in computation because the bin displacements are random instead of dependent on the net forces caused by neighboring bins. However, this reduction in computation may be offset by the frequent rejection of the new bin arrangements that do not decrease the cost function.

Unlike the equalization of local volume density, simulated annealing does not necessarily displace each bin according to its force vector in a local neighborhood of bins. Therefore, the outputs of the two methods are likely to differ, possibly by substantial amounts. Although simulated annealing is likely to eventually converge to an equalized histogram, the reordering of the bin arrangement can be much greater than that of the best possible solution.

5.4 Parallel Computing

The method of local density equalization can be structured for efficient computation by a set of parallel processors. Every iteration requires the computation of a displacement vector for each occupied bin based on the net force exerted in a local neighborhood. Since the bin displacements are mutually independent, the bins could be treated in parallel by a series of separate processors as long as each processor could access the necessary bin position and count information of the current arrangement. Computing the bin forces and displacements lies at the core of the technique, therefore it should be possible for the total computation time to be cut by a factor approaching the number of processors employed, assuming that inter-processor communication is not the limiting factor.

Similarly, the simulated annealing method can benefit from parallel processing. The random displacements of the bins are mutually independent, so the computation of these displacements and the consequent updated trial positions can be accomplished
by a set of parallel processors. It is likely that these processors could also generate a partial cost function based on the trial positions of those bins under their control. The total cost function could be constructed at a central point from these partials obtained from the set of processors. If implemented efficiently so that processing is not limited by inter-processor communication, the total computation time might be cut by a factor approaching the number of processors.
Chapter 6

Conclusion

Histogram explosion has been shown to be an effective and flexible new method of enhancing color images. The fact that explosion is not based on any model of human color perception allows it much greater freedom than previous histogram modification schemes. True multivariate operation directly upon the RGB histogram data is one of the major strengths of histogram explosion. Not only are color coordinate conversions unnecessary, but equivalent results cannot be easily achieved by processing within another color space. With a properly chosen operating point, histogram explosion can alter almost any RGB histogram to extend across nearly the entire RGB color space. All of the other approaches studied during this research are generally unable to expand histogram data clusters across the full extent of the displayable color range without resorting to boundary clipping or the generation of undisplayable color values.

The choice of operating point allows tremendous flexibility in controlling the bias of the histogram explosion algorithm. By constraining the operating point to the gray line, the explosion process becomes perceptually well-behaved in that the hue values remain constant. In this way, the method can be quite useful to enhance color images for human viewing and interpretation without objectionable or confusing hue alteration. Even with the operating point confined to the gray line, however, histogram explosion allows far greater opportunity for tailoring of the outcome that do
the other methods investigated.

The freedom to specify the operating point is both a benefit and a disadvantage. To obtain the results desired, it is sometimes necessary for the operating point to be specified by a human operator. For example, there is currently no algorithm for automatically selecting the best operating point location to produce an intended result. Often, however, good results can be obtained by placing the operating point at the data cluster's centroid.

After experimenting with several parameters in the histogram explosion algorithm, several conclusions can be drawn. One of these parameters is the choice between zeroth-order and first-order interpolation during construction of the ray histograms from the RGB histogram. The difference in the results of the two interpolation methods is usually very subtle. Therefore, zeroth-order interpolation is the method of choice based on its much simpler computation. Along the rays, both a linear stretch and histogram equalization have been explored. Here, the differences are more noticeable. Histogram equalization can sometimes cause artifacts due to bins converging at or near the operating point. The linear stretch does not give rise to such artifacts. For this reason and because of its somewhat simpler computation, the linear stretch method is usually the better choice of the two.

Unlike many of the earlier color histogram methods, histogram explosion will sometimes alter the original 3-D order of the bins. Left unchecked, this effect can cause distortion and artifacts in the output images. Also, the output histograms generated by the explosion method have distributions that are neither intended nor guaranteed to be uniform.

Since histogram explosion operates in a multivariate fashion on the 3-D histogram, it is expected to require much greater computation than methods that
operate independently on the histogram’s individual component variables in a given color space. Full multivariate processing also demands significantly more computation than methods that adjust some of the component variables independently and others dependently. In some cases, the extra computational cost of explosion can be justified by its ability to produce more useful results. The computational complexity of histogram explosion is roughly two orders of magnitude greater than those of many separate component processing methods. However, there appear to be several possibilities for significant speed improvement of the explosion technique.

Some thought has been devoted to the problem of multidimensional histogram equalization. The two main approaches investigated, iterative local volume density equalization and simulated annealing, seem to exhibit the potential for accomplishing the job. Both methods are computationally intensive and thus may be impractical with current computing resources. With further investigation into the problem, a practical algorithm for 3-D histogram equalization may be possible. Continued research into possible applications of equalized multidimensional histograms may lead to further advances in the field of image processing.
Appendix A

One-sided Histogram Equalization

Gonzalez and Wintz [6] have described the theoretical foundations of histogram equalization. To briefly review the development and then extend it to the case of one-sided histogram equalization, consider the continuous probability density function \( f_r(r) \), where \( r \in [0,1] \). Its cumulative distribution is:

\[
F_r(r) = \int_0^r f_r(w) \, dw
\]  
(A.1)

Histogram equalization is based on the transformation to the new random variable \( s \) by the function \( T \), where

\[
s = T(r) = F_r(r) \Rightarrow \frac{ds}{dr} = f_r(r)
\]  
(A.2)

By transforming from \( r \) to \( s \), the density becomes uniform:

\[
f_s(s) = f_r(r) \frac{dr}{ds} \bigg|_{r = T^{-1}(s)} = f_r(r) \frac{1}{f_r(r)} = 1
\]  
(A.3)

Now separate \( f_r(r) \) into two parts on either side of an operating point \( r_0 \in [0,1] \).

\[
f_r(r) = g_r(r) + h_r(r)
\]  
(A.4)

where:

\[
g_r(r) = \begin{cases} 
  f_r(r), & 0 \leq r \leq r_0 \\
  0, & r_0 < r \leq 1 
\end{cases}
\]  
(A.5.a)
Note that \( g(r) \) and \( h(r) \) are technically not probability density functions themselves because their integrals are not unity.

At this point, let us concentrate on the case of \( g(r) \). A probability density function can be formed from \( g(r) \) by transforming to a new random variable \( p = r/r_0 \), where \( p \in [0,1] \). Since \( r = r_0 p \):

\[
g_p(p) = \frac{g_r(r_0 p)}{G_r(r_0)} = \frac{g_r(r_0 p)}{\int_0^{r_0} g_r(w) \, dw} = \frac{f_r(r_0 p)}{\int_0^{r_0} f_r(w) \, dw}
\]

(A.6)

Hence:

\[
G_p(p) = \frac{\int_0^p f_r(r_0 w) \, dw}{\int_0^{r_0} f_r(w) \, dw}
\]

(A.7)

The boundaries of Eq. A.7 are:

\[
G_p(0) = 0 \quad , \quad G_p(1) = \frac{\int_0^{r_0} f_r(r_0 w) \, dw}{\int_0^{r_0} f_r(w) \, dw} = 1
\]

(A.8)

Therefore, \( g_p(p) \) is a probability density function.

Next, histogram equalization is performed on \( g_p(p) \) to obtain \( g_q(q) \), where:

\[
q = T(p) = G_p(p) \Rightarrow \frac{dq}{dp} = g_p(p)
\]

(A.9)
so:

\[ g_q(q) = g_{p}(p) \left[ \frac{dp}{dq} \right]_{q = T^{-1}(q)} = \frac{g_p(p)}{g_p(p)} = 1 \ , \ q \in [0,1] \]  (A.10)

The final step involves transforming to the new variable \( s \in [0,r_0] \) by scaling the uniform probability density function \( g_q(q) \) such that \( G_s(r_0) = G_s(1) = F_r(r_0) \). The result describes one-sided histogram equalization about point \( r_0 \):

\[
 g_s(s) = \begin{cases} 
 \frac{1}{r_0} \int_{0}^{r_0} g_q(w) \, dw = \frac{1}{r_0} F_r(r_0) = \frac{c_1}{r_0} , & 0 \leq s \leq r_0 \\
 0 , & r_0 < s \leq 1 
\end{cases} 
\]  (A.11)

Similarly, the operation on \( h_r(r) \) yields:

\[
 h_s(s) = \begin{cases} 
 0 , & 0 \leq s \leq r_0 \\
 \frac{1}{1-r_0} \int_{r_0}^{1} h_r(w) \, dw = \frac{1}{1-r_0} (F_r(1) - F_r(r_0)) = \frac{c_2}{1-r_0} , & r_0 < s \leq 1 
\end{cases} 
\]  (A.12)

After determining the values of \( c_1 \) and \( c_2 \), one can shift the operating point to a new position \( s_0 \in (0,1) \):

\[
 g_s(s) = \begin{cases} 
 \frac{F_r(r_0)}{s_0} = \frac{c_1}{s_0} , & 0 \leq s \leq s_0 \\
 0 , & s_0 < s \leq 1 
\end{cases} 
\]  (A.13.a)

\[
 h_s(s) = \begin{cases} 
 0 , & 0 \leq s \leq s_0 \\
 \frac{F_r(1) - F_r(r_0)}{1-s_0} = \frac{c_2}{1-s_0} , & s_0 < s \leq 1 
\end{cases} 
\]  (A.13.b)
**Theorem**

Properly shifting the operating point of one-sided histogram equalization will yield the same results as "true" histogram equalization. In effect, there must exist some operating point $s_e \in (0, 1)$ such that:

$$g_e(s) = \begin{cases} \frac{c_1}{s_e} = f_e(s), & 0 \leq s \leq s_e \\ 0, & s_e < s \leq 1 \end{cases}$$ (A.14.a)

and

$$h_e(s) = \begin{cases} 0, & 0 \leq s \leq s_e \\ \frac{c_2}{1 - s_e} = f_e(s), & s_e < s \leq 1 \end{cases}$$ (A.14.b)

**Proof**

To match the result of "true" histogram equalization, Eqs. A.14.a and A.14.b must be made equal.

$$\frac{c_1}{s_e} = \frac{c_2}{1 - s_e} \Rightarrow s_e = \frac{1}{1 + \frac{c_2}{c_1}}$$ (A.15)

Both $c_1$ and $c_2$ are always confined to the interval $[0, 1]$. Therefore,

$$c_1 + c_2 = 1 \Rightarrow s_e \in [0, 1] \quad \text{QED}$$

Notice that if $c_1 = c_2$, then $s_e = 0.5$. This situation corresponds in the discrete case to shifting the median bin to the center of the gray level range and performing one-sided histogram equalization in both directions.

In the case of discrete-valued histograms, the properties illustrated in this appendix are only approximate because of the effects of quantization.
References


