QUALITY AND COMPRESSIBILITY ANALYSIS OF CHECKERBOARD-BASED DIGITAL HALFTONING ALGORITHMS

THESIS

Presented to the Graduate Council of
Southwest Texas State University
in Partial Fulfillment of
the Requirements

For the Degree

Master of Science in Computer Science

By

Robert N. Villarreal

In

San Marcos, Texas

For Graduation in May of 2002

To my wife,

who stood with me upon the shoulders of giants

To my teachers,

who showed me to climb

ACKNOWLEDGEMENTS

I would like to begin by thanking my wife, Rina Villarreal, who unselfishly put my interest before her own and supported me throughout my career as a college student. Thanks also go to my parents who taught me to value education and learning.

I would also like to thank the many people and organizations that donate money for SWT scholarships. Their generosity provided the financial basis for me to realize my dreams.

I am indebted to Robert and Susie Case for their hospitality, friendship, and time. This project would have not been possible without their cooperation, communication, and support. Trish Sumbera also deserves my gratitude. Her open door and attentive ear helped me overcome the more stressful aspects of this project. Thanks also go to Carlos Gutierrez for the opportunities he offered me as an undergraduate and for his continued support.

Finally, I would like to thank each of my thesis committee members. Greg Hall offered me many enjoyable diversions from this project ranging from friendly conversations regarding hobbies to mini-projects for exploring new technologies. From Carol Hazelwood, I received excellent advice on life, the real world, and continuing my education. Wilbon Davis deserves my deepest gratitude. He is one of the most intelligent and well-rounded individuals that I have ever met. Our frequent short yet

fruitful discussions made this project blossom. His support has contributed greatly to my success.

I am a better person because of each of these people. As such, I consider them friends.

This manuscript was submitted on January 31, 2002.

Table of Contents

page

List of Figures

page

List of Tables

page

List of Equations

page

List of Charts

page

ABSTRACT

This study considers the quality and compressibility of bi-level images produced by checkerboard-based halftoning algorithms. The specific halftone methods of this study are those described by R.M. Case. The bi-level images are created using C/C++ implementations of these algorithms (Villarreal, 2001). Image quality is assessed using human surveys and statistical objective metrics. Correlations (or lack of) between the subjective and objective data are also observed. Image compressibility is studied via the C/C++ implementation of two new and original run length encoding (RLE) methods (Villarreal, 2001). These RLE methods are used to remove the positional redundancy from the pixels composing the bi-level images. Other methods are subsequently used to remove statistical redundancies form the resulting files. Mock image file formats based on Case's halftoning methods and the above compressions are compared to other popular image formats to assess the potential of this technology.

CHAPTER 1

INTRODUCTION

This research considers the implementation, quality, and compressibility of various images produced by Case's checkerboard-based halftoning algorithms (1993, 2001). Other than the patents issued and pending for these methods, they are relatively absent from technical literature. This study is an exercise in digital halftoning, image processing and software development.

The remainder of this document is organized into two sections: an image processing background and a research analysis. The image processing background is subsequently divided into the following chapters:

- *The Human Visual System*—This chapter discusses the physics and psychophysics issues related to digital halftoning
- *Digital Image Representation*—This chapter describes the issues regarding the acquiring and processing of digital imagery.
- *Dithering*—This chapter surveys several digital halftoning methods from technical literature.
- *Compression*—This chapter surveys the previously documented methods of compression used on digital imagery.

The research analysis portion of this document is divided as follows:

- *New Methods*—This chapter describes Case's new halftoning methods and the new compression methods researched in this study.
- *Project Management*—This chapter explains the organizational methods used in the management of this study.
- *Research Methods*—This chapter describes the methods used to acquire data for this study.
- *Data Analysis*—This chapter analyzes the properties of the studied methods along with data taken from the methods of the previous chapter.

CHAPTER 2

HUMAN VISUAL SYSTEM

The human visual system (HVS) is the basis of computer graphics and digital imaging. It is the complex standard by which all methods and models are judged. Its study reveals how strengths and limitations can be exploited to produce many sensory effects. Interestingly, the properties of the HVS can be viewed from several perspectives each of which contributing a unique piece to our understanding of vision. The following sections briefly outline some perspectives of the HVS as a backdrop to this study.

Physics of Visual Signals

Visible light can be considered as signals from our surroundings. From this perspective, the HVS is decomposed into a series of simple processes that interact in order to produce our sense of vision. Several authors in the literature model the HVS with this approach (Higgins, 1977; Daly, 1997; Girod, 1997; Lubin, 1997; Taylor, Allenbach, and Pizlo, 1998; Zhang & Wandell, 2000).

Electromagnetic Radiation

Vision begins with light. Visible light is electromagnetic radiation (EMR) with wavelengths ranging between 4,000 and 7,000 Angstroms (10^{-10} m). EMR interacts with matter in many ways. Two ways of particular interest include the reflection and emission of EMR.

EMR Reflection

Matter reflects light in two ways: specular reflection and diffuse reflection. Figure 2.1 illustrates these methods of reflection.

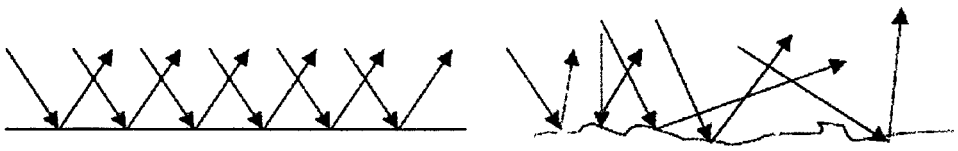

Figure 2.1—Specular and Diffuse Reflection

Specular reflection of light occurs on smooth surfaces such as the face of a mirror. Light reflects off such surfaces according to the law of reflection. Figure 2.2 illustrates this law.

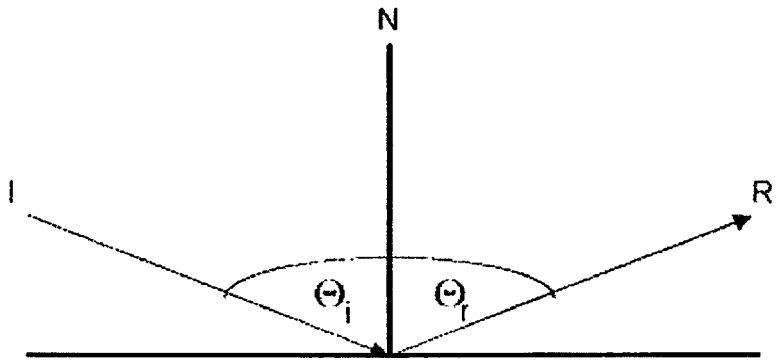

Figure 2 2—The Law of Reflection

In this diagram, **I** is the incident light ray and **R** is the reflected ray. **N** is the surface normal. This law states that **N** is a bisector for the angle made by **R** and **I**. In other words, θ_i and θ_r are equal. In specular reflection, nearly all of the incident photons (light particles) striking a surface "bounce" (Watt, 1998) off without interacting with the molecules of the surface.

Diffuse reflection occurs on "rough" surfaces. As its name implies, diffuse reflection uniformly scatters the incident light in all directions. Unlike specular reflection, an interaction occurs between the molecules of the surface and the incident photons. Figure 2.3 uses the Bohr model of the atom to illustrate this interaction.

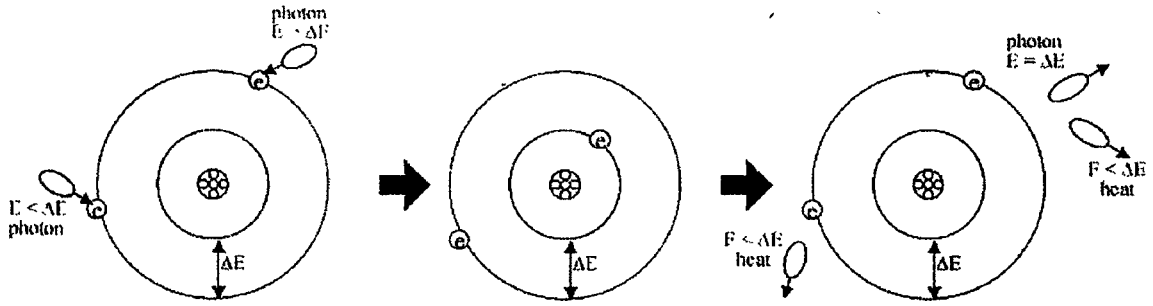

Figure 2.3—Interaction between Electrons and Photons during Diffuse Reflection

The figure shows that most photons induce extra vibrations in the atom that are subsequently dissipated as heat energy. However, if an incident photon has an energy equal to some "resonance" energy, then the photon is absorbed by an electron in the atom. This absorption causes the electron to "jump" to a higher energy level. The energy of this "jump" is equal resonance energy of the atom. For the atom to return to a state of equilibrium, the excited electron must emit a photon with an energy equal to the resonance energy. The direction of this emitted photon is nondeterministic; thus, the incident light is effectively scattered ("reflected") by this absorption and subsequent emission.

As an example of EMR reflection, imagine an image printed on glossy paper (perhaps a photograph). The reflections caused by the glossy coating are a specular while the "reflection" caused by the pigments that make up the image are diffuse.

EMR Emission

EMR emission is a process similar to diffuse reflection. EMR emission processes begin when electrons are driven into excited (higher energy) atomic states. In diffuse reflection, incident (visible light) photons are the instigators of these excited electrons; however, such photons are not the only way to produce non-equilibrium states in atoms.

The various image output devices considered in this study emit EMR in different ways. Cathode ray tubes (CRTs) use high voltages to accelerate electrons from an emitter into phosphors that coat the screen. The energy from these accelerated electrons excite the phosphor atoms which return to equilibrium by releasing photons that the human eye collective detects as an image. Thin film transistor monitors (TFTs) emit

light in a very different manner. This technology uses voltage sensitive liquid crystals as polarizers to control how a backlighting (the photon emitting source) is visible to a user.

Eye Optics

Most of the light in the environment is either reflected or emitted. However, once light reaches the eye, the laws of refraction take over to focus images on to the retina. Refraction governs the way that light travels through different transparent media.

Simple Refraction

Figure 2.4 illustrates the law of refraction.

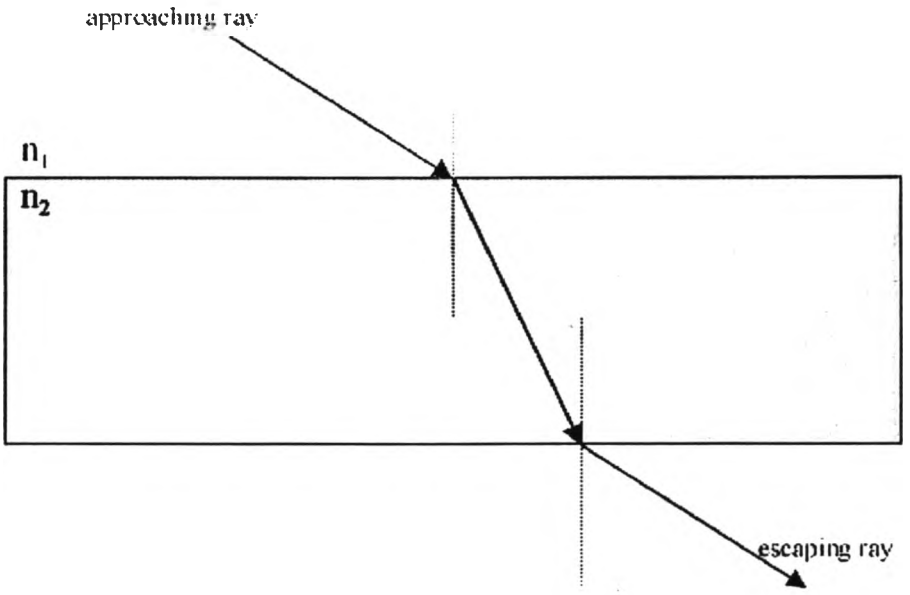

Figure 2.4—Law of Refraction

In these figure, a ray of light travels through a medium. If **n**, the index of refraction, is greater in the second medium, then the light will "bend" towards the normal as illustrated

by the approaching ray. The opposite happens for $\mathbf{n_2}$ less than $\mathbf{n_1}$ as illustrated by the escaping ray.

Snell's Law determines the angles that the incident and refracted light rays make with the surface normal. This law is stated as:

$$n_i \sin(\theta_i) = n_r \sin(\theta_r)$$

Equation 2 1—Snell's Law

Application of Snell's Law can describe the nature of light passing through a lens (or system of lenses).

The Cornea and Lens

The eye's cornea and the lens represent it means of refracting light to form images on the retina. This lens system refracts light similar to a doubly convex lens. Figure 2.5 shows how a doubly convex lens focuses light from an object to a point.

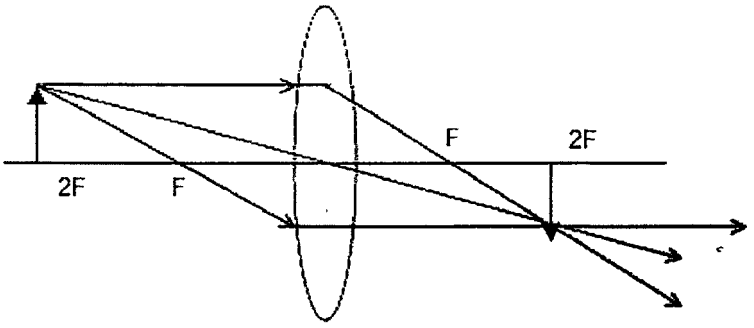

Figure 2 5—Doubly Convex Lens

The point at which the rays intersect is the image location (note that it is upside-down).

The ideal situation of the previous paragraph describes perfectly focused images. In actuality, the eye (and any real lens system) suffers from imperfections. These imperfections cause a phenomenon known as aberration. Aberration blurs images. As a result, a point source of light focuses on the retina as a "spread" of light. Figure 2.6 gives a formal representation of this spread known as a point spread function.

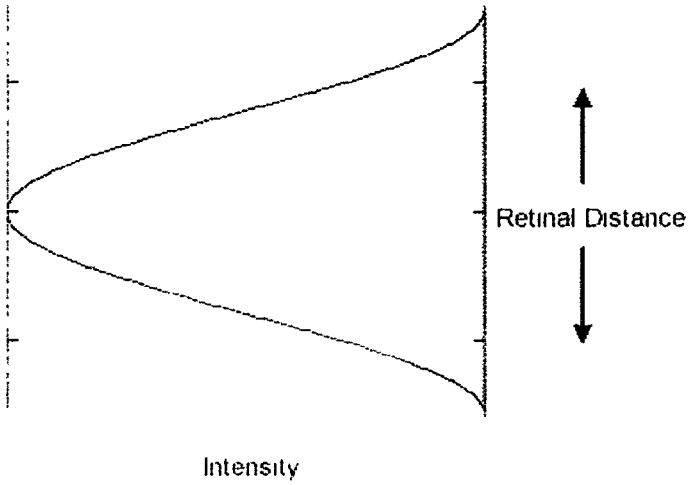

Figure 2 6—Point Spread Function

The success of image dithering methods can be partly contributed to the point spread functions in the eye.

The eye's lens system has two interesting properties. Firstly, the cornea refracts light 80% more than any other part of the eye (Wandel, 1995). This property may seem counterintuitive. However, the change in the index of refraction at the air-cornea interface is much greater than any other interface in the eye. Secondly, muscles connected to the eye's lens can change its shape; thus, changing the refractive properties of the lens. These adjustments occur so that light rays converge to stimulate the

photoreceptors of the retina which always lay a fixed distance from the cornea-lens system. This ability is known as accommodation.

The Retina

The retina represents another interface in the eye. At this interface, the eye's photoreceptors convert electromagnetic signals (light) into electrochemical signals that travel through the optic nerve to the brain. Light reacts with pigments inside of the photoreceptors to produce these electrochemical signals.

Two kinds of photoreceptors exist: rods and cones. Rods are particularly sensitive to light intensity and are generally located around the peripheral of the retina. They provide coarse and colorless vision. Cones only respond to relatively bright light; however, they allow the eye to detect detail and color. Most of the retina's cones are concentrated directly behind the lens in an area known as the fovea. As a result, the fovea primarily determines the eye's sensitivity to spatial frequencies in images.

Human Perception of Imagery

The optics of the eye determines the way in which images are formed on the retina; however, optics alone does not describe how humans perceive their surroundings.

Visual Acuity

Visual acuity measures the spatial resolution that the eye can discern. Qualitatively, it describes the eye's ability to see fine detail. The common Snellen eye

metric (e.g. "20/20") refers to visual acuity. This metric implies the minimum angle of resolution (MAR) that the eye can resolve. Normal visual acuity, 20/20 vision, is defined as the ability for a person to perceive a spatial pattern separated by an angle 1/60° from a viewing distance of 20 feet. Figure 2.7 illustrates how visual acuity is measured.

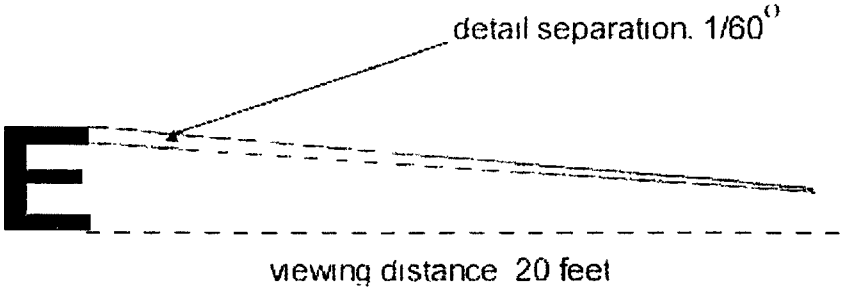

Figure 2 7—Normal Visual Acuity

In this figure, the strokes of the "E" are purposely designed with a width of 1/60°.

As previously stated, the optics of the eye are not perfect; thus, they limit human visual acuity. Raleigh's criterion describes these limitations. Recall that spherical aberration in the eye causes point sources of light to focus on the retina as a point spread functions. Raleigh's criterion states that resolution of two objects occurs only if the objects are separated by the width of their point spread functions (Scharwtz, 1999). Figure 2.8 illustrates Raleigh's Criterion in terms of point spread functions.

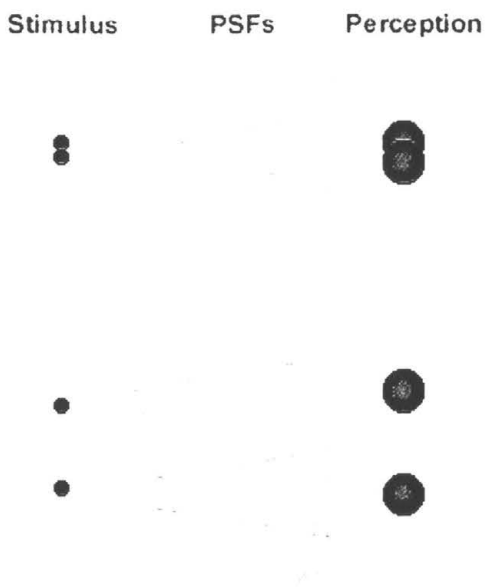

Figure 2.8— Raleigh's Criterion

The closely spaced stimuli in the top half of Figure 2.8 cannot be resolved as two individual stimuli. The closely spaced dots of image halftoning methods use Raleigh's criterion to produce the illusion of image continuity.

Spatial Integration

Spatial integration occurs past the limits of human visual acuity. It describes the eye's ability to collectively interpret stimuli smaller than the eye's acuity threshold. This collective interpretation arises from the physiology of the retina.

The eye's photoreceptors perform "signal transductions" that convert an EMR signal into a neural signal (Scharwtz, 1999). These neural signals are transmitted via ganglion cells to the brain. Spatial integration occurs because signals from several photoreceptors converge on to each ganglion cell (Wandel, 1995). These signals

effectively combine to form one transmitted signal. Figure 2.9 illustrates this phenomenon using a pattern of alternating colors.

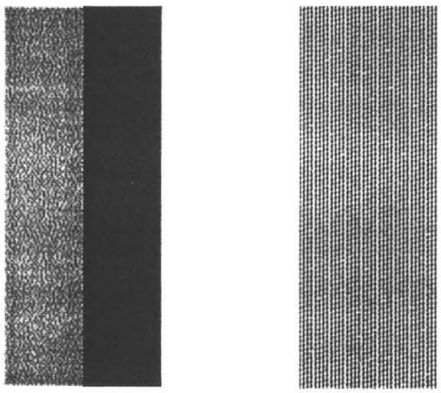

Figure 2.9—Spatial Integration

In this figure, the block on the right is not actually purple; instead, it is composed of lines of red and blue that alternate frequently. Similarly, frequently alternating pixels of white and black will produce grays. Dithering methods exploit this property to produce images with a wider range of perceptible intensities.

CHAPTER 3

DIGITAL IMAGE REPRESENTATION

Several issues are involved in the digital representation of images. Digitization is a process that approximates the continuous signals that exist in the macroscopic world. The quality of this approximation depends on the amount of resources dedicated to representing the signal. The infinite amount of information lost through digitization is compensated by the improved ability to process the discrete version of the image. This chapter reviews the quality and processability issues that related to this study.

Pixels

For many, the term "pixel" means "little square"; however, a pixel actually represents a discrete sample of the continuous world. This sampling is necessary because computers can operate using only discrete structures. Thus, a digital image is merely the combination of thousands of these samples to produce an acceptable estimate of the continuous image.

In fact, digital image quality is directly affected by the rate at which the continuous scene is sampled. Fourier analysis allows a continuous image to be viewed as a two-dimensional signal made up of many frequencies and the Nyquist Theorem imposes a lower bound on an accurate sampling rate for this signal. According to Nyquist, a continuous signal can be accurately reconstructed only if the signal is sampled at a frequency greater than two times its highest frequency component.

$$f_{Nyquist} > 2f_h$$

Equation 3.1—Nyquist Frequency

Figure 3.1 illustrates this idea (note the placement of the dots).

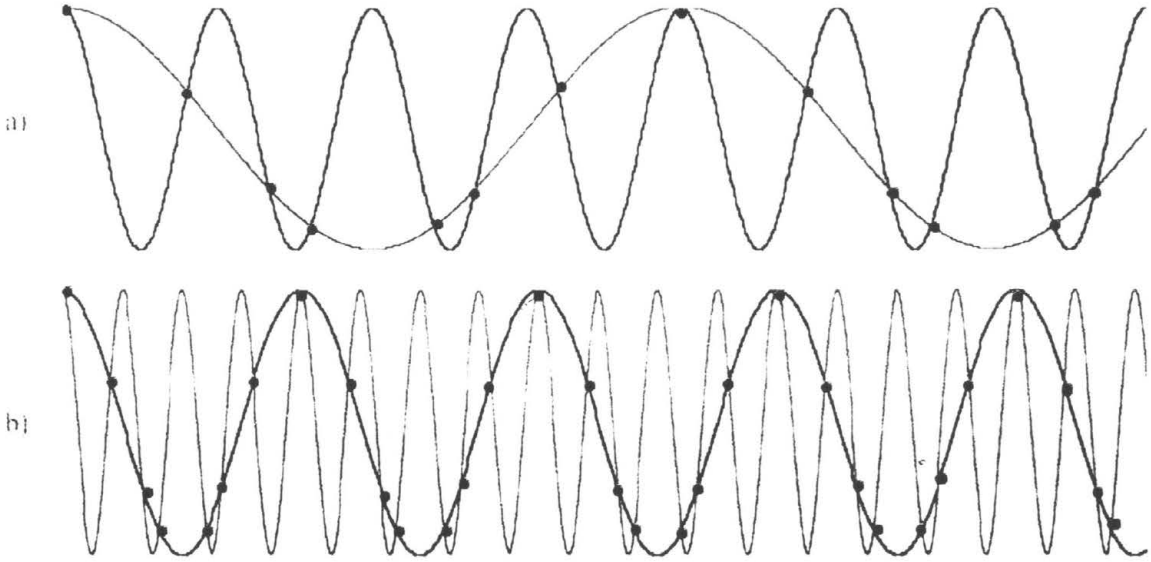

Figure 3.1—Nyquist Theorem

In this figure, the red line represents the sampling frequency. Figure 3.1(a) depicts an undersampled signal. Undersampling causes high frequencies in the original signal to

appear as low frequencies in the reconstructed signal. Figure 3.1(b) shows an adequately sampled signal. The case in which $f_{Nyquist} = 2f_h$ (not shown) is special because the accuracy of the reconstruction depends on the phase of the sampler.

However, despite the Nyquist Theorem aliasing artifacts appear in digital images. Simply put, aliasing in digital images refers to noticeable errors such as "jaggies." These errors occur mainly because pixels are discrete samples and always estimates of continuous signals.

Another limitation imposed by the discrete nature of pixels is a discrete color and intensity spectrum. The number of bits used to store each pixel determines the depth of these spectra. Grayscale images are typically composed of 1-, 2-, 4-, or 8-bit pixels. Color images are more complex and require 8-, 24-, or 32-bits. Raster graphics are simply $H \times W$ matrices of n-bit pixels stored as bitmaps.

This study considers grayscale images stored at 8-bits per pixel or less. Most of these images are converted 24-bit RGB color images. The 24-bit RGB format allocates one byte to the red, green, and blue intensities of the pixel. Conversion from this format to grayscale intensity occurs according to the following equation.

$$I_{grayscale} = 0.3I_{red} + 0.6I_{green} + 0.1I_{blue}$$

Equation 3.2—Color Intensity to Grayscale Intensity

The color sensitivities implied above arise from the concentration and distribution of cones cells in the human eye.

Resolution

Resolution is a rough gauge of image fidelity. It specifies how well the discrete domain samples the original continuous image. Specifically, resolution identifies the number of pixels contained by an image. It also describes precision; thus, as resolution increases, digital images become clearer.

For computer screens, resolution is often denoted in pixels per inch (ppi). For print displays, it is communicated in dots per inch (dpi). Dot and pixels typically represent the device's smallest displayable unit.

The limitations of discrete samples often appear in images as aliasing artifacts with high frequencies. However, increased resolution does not remove these artifacts. Instead, it limits the degree to which they are noticeable because the artifacts occur at even higher (less perceivable) frequencies. Figure 3.2 depicts a classic illustration of this phenomena using and infinite checkerboard (Watt, 1997, 1998; Foley, 1990).

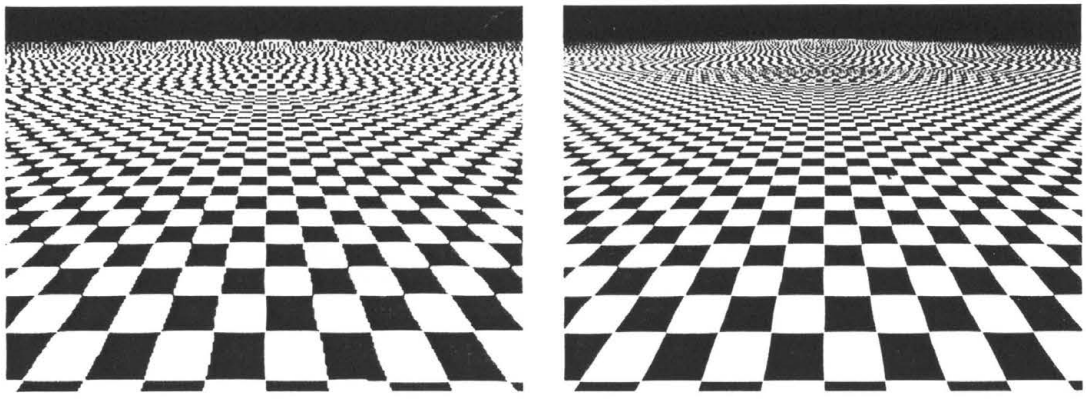

Figure 3.2—Infinite Checkerboard

Image Metrics

Digital images have many properties other than resolution and sampling frequencies. Often these properties are measured or described via numerical metrics. The following sections explain a few of these metrics that are relevant to this study.

Histograms

Image histograms are statistical distributions of the intensity (or color) values present in an image. For an 8-bit grayscale image, histograms illustrate how many pixels of each 8-bit intensity are present. Figure 3.3 shows an example of such an image and its histogram.

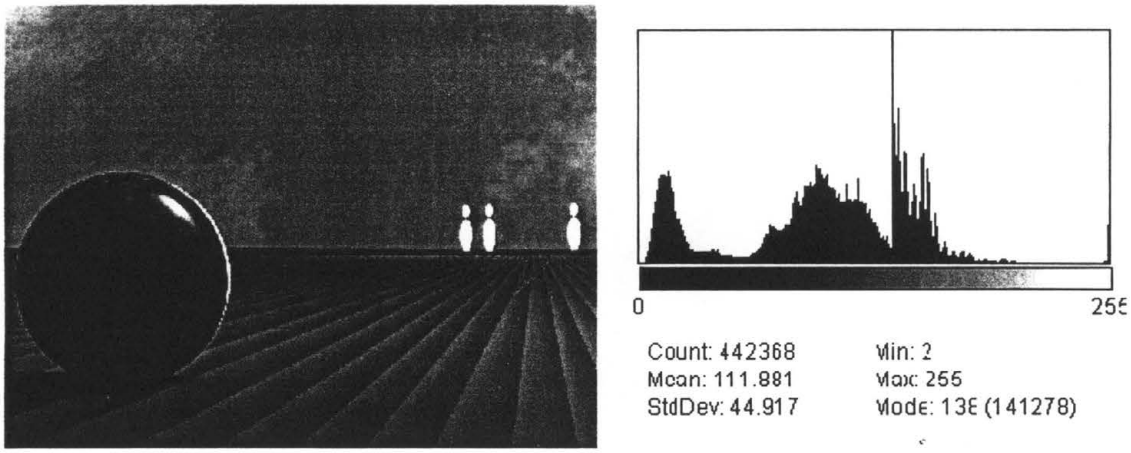

Figure 3.3—An Image an its Histogram

The shaded area under the histogram curve represents the total number of pixels present in the image. Qualitatively, histograms convey an image's overall contrast. As such, they can be used to enhance images will poor contrast. Many algorithms exist for histogram manipulation and Figure 3.5 of the next sections gives an example.

Quality Metrics

Many different types of image quality metrics exist. Some metrics are based on an image's statistical or spectral (Fourier) properties while others gauge quality by modeling the human visual system. Avcýbas and Sankur (1999) give a nice overview of many of them.

Each type of metric focuses on a set of attributes that contribute to the image's quality. Many of these metrics attempt to condense the idea of subjective quality (as determined by a human) into a single machine friendly number. Such methods are easy to compute and offer significant information about an image. However, due to this ease of computation, such metrics do not always correlate well to the human determination of quality.

This study considers the correlation of two statistical quality metrics to human survey data. According to Johnson (1984), correlation coefficients empirically determine the likelihood of a relationship between a pair of data sets. A correlation coefficient near 1 or –1 implies that the data sets are highly correlated (directly and inversely, respectively) and are likely related. However, since correlation is a statistical concept, there is no guaranteed relationship, only a highly probable one. The following table briefly describes these metrics.

Metric	*Summary*
RMS Error	The RMS metric is the root mean squared error between an original image and a distorted image. This number can be derived by the following equation: $$RMS = \frac{1}{N^2}\sqrt{\sum_{j=0}^{N}\sum_{i=0}^{N}[I(i,j)-D(i,j)]^2}$$
Universal Quality Index	Zhou's and Bovik's (2000) Universal Quality Index is a statistically based image quality metric that is defined by the following equation: $$Q = \frac{\sigma_{xy}}{\sigma_x\sigma_y}\cdot\frac{2\bar{x}\,\bar{y}}{(\bar{x})^2+(\bar{y})^2}\cdot\frac{2\sigma_x\sigma_y}{\sigma_z^2+\sigma_y^2}$$ The first term measures the degree of linear correlation between the original image pixels, **x**, and the distorted image pixels, **y**. The second term measures the mean luminance with respect to the pixels sets. Finally, the third term measures the similarity of the images' contrast.

Table 3.1—The Statistical Quality Metrics Used in this Study

Image Transforms

Image transforms are used to enhance and impair images and to accentuate specific properties of an image. Each transform operates in a unique way, but different transforms can produce similar results. The following sections describe some of the image transformation algorithms considered in this study.

Pixel Manipulation

The simplest image transforms manipulate individual pixels. In particular, they modify pixel intensities or positions without any information pertaining to neighboring pixels. They are used for image enhancement and are often computationally simple.

Homogeneous pixel transformations have no dependence on pixel position while inhomogeneous transformations do.

Thresholding is possibly the simplest of pixel manipulation processes. This process maps pixels greater than some threshold value to 1 (white) and those less than the threshold to 0 (black). Figure 3.4 illustrates thresholding.

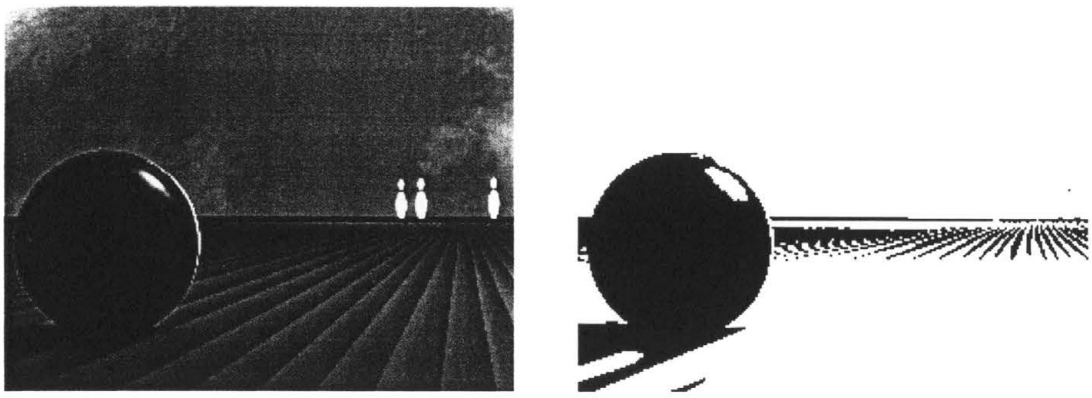

Figure 3.4—Pixel Thresholding

Thresholding can be combined with more other processes in order to produce transforms that are more complex.

Image subtraction is a homogeneous process that finds the differences between pixels of two images. If A and B are both M×N pixel matrices, then image subtraction can be represented as:

$$\begin{matrix} a_{11} & \Lambda & a_{1n} \\ \mathrm{M} & \mathrm{O} & \mathrm{M} \\ a_{m1} & \Lambda & a_{mn} \end{matrix} - \begin{matrix} b_{11} & \Lambda & b_{1n} \\ \mathrm{M} & \mathrm{O} & \mathrm{M} \\ b_{m1} & \Lambda & b_{mn} \end{matrix} = \begin{matrix} (a-b)_{11} & \Lambda & (a-b)_{1n} \\ \mathrm{M} & \mathrm{O} & \mathrm{M} \\ (a-b)_{m1} & \Lambda & (a-b)_{mn} \end{matrix}$$

Equation 3.3—Image Subtraction

Figure 3.5 shows an example of image subtraction.

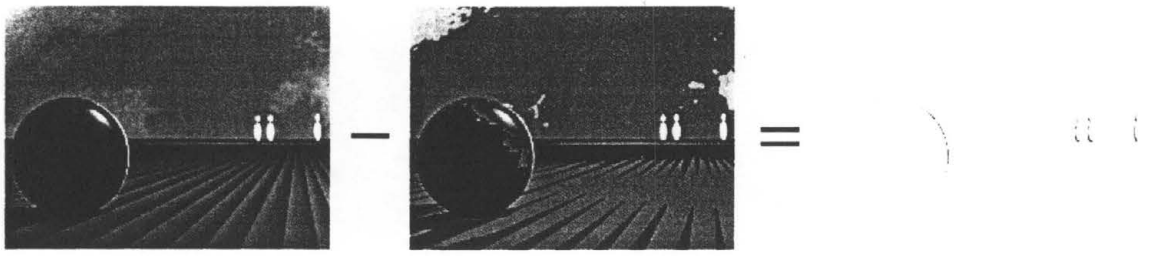

Figure 3.5—Image Subtraction Example

This study uses image subtraction to help assess image fidelity between an original image and a processed image.

Contrast stretching is a homogeneous pixel manipulation which enhances images that have low contrast. Low contrast images use only a small part of their intensity spectrums and often appear to be "too dark" or "too bright." Contrast stretching simply maps intensity inputs to other intensities in the spectrum using a piecewise linear function. Figure 3.6 illustrates contrast stretching.

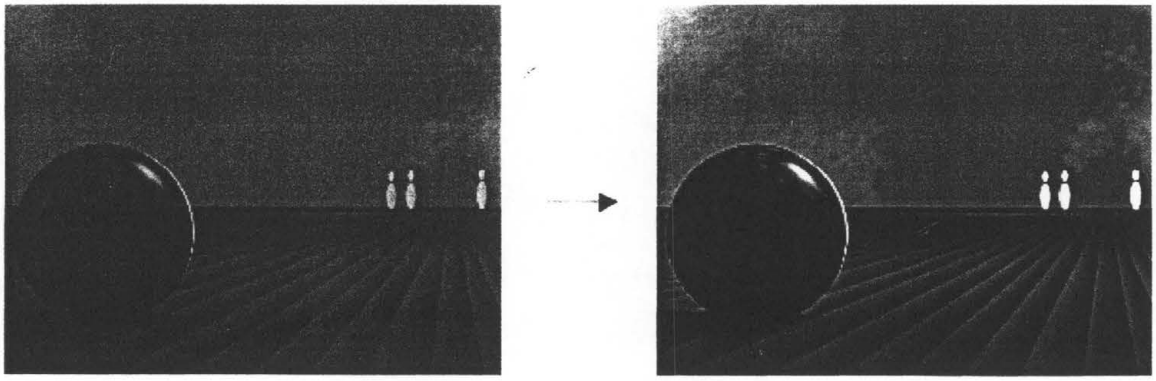

Figure 3.6—Contrast Stretching

This study uses contrast stretching to illustrate how images produced by Case's dithers appear to "grow" out of a 50% gray checkerboard.

Rotation of pixels is an inhomogeneous process that displaces pixels by a specified angle. It is a good example of a process that can be implemented in a two-pass manner. Two-pass transforms apply one transform along one orientation of the image and another along another orthogonal orientation. In effect, they build a more complex transform from simpler operations. For rotation, these transforms occur along the horizontal and vertical axes of the image. In this study, rotation is used to animate images in order to check for moirés.

Fourier Transforms

Since digital images can be viewed as spatial signals, Fourier analysis can be applied to them. Fourier transforms map the two-dimensional image signals onto the spatial frequency domain. These transforms are simply different representations of the original data and are completely invertible; thus, no information is loss by converting between the two domains. Figure 3.7 shows an image with its Fourier transform

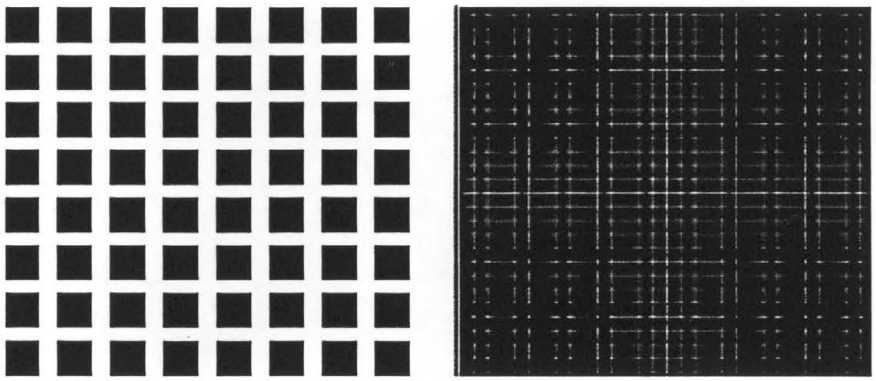

Figure 3.7—Image and its Fourier Transform

In the Fourier image, the circumference of a circle centered on the origin, specifies a set of spatial frequencies of equal rates of undulation. This property can be

used to filter out unwanted frequencies before reconstructing the image with the inverse transform. Computationally, filtering in the Fourier domain often outperforms transforms with the same effect applied in the image domain despite the overhead involved in applying and inverting the Fourier transform.

Fourier transformations also have the ability to "separate" information that is "spread" throughout the image domain (Watt, 1998). Closer inspection of the Fourier image in Figure 3.7 makes this quality apparent. The texture of the image produces coherences in the Fourier domain. These coherences are the diagonal bands that crisscross at the origin.

In many disciplines, Fourier transforms are performed on continuous functions; however, digital imaging requires the use of a special form of the transform known as the Discrete Fourier Transform (DFT). Equation 3.4 shows how to compute the DFT of an image function ***I(x, y)*** and how reconstruct the image from its Fourier transform ***F(u, v)***.

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I(x,y)\, e^{(-i2\pi(\frac{ux}{M}+\frac{vy}{N}))}$$

$$I(x,y) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} F(u,v)\, e^{(i2\pi(\frac{ux}{M}+\frac{vy}{N}))}$$

Equation 3.4—The Discrete Fourier Transform and Its Inverse

Note how the DFT is a complex quantity. Thus, images such as Figure 3.7 are often derived using the square modulus of the Fourier transform[1]. Such images are called *power spectra.*

Image Filtering

An important property exists between the image and Fourier domains: convolution. Convolution is an image domain operation that replaces a pixel value with the weighted average of neighboring pixels values. The number of neighbors involved in the operation depends on the size of the convolution kernel. For symmetry, the kernel is generally odd sized so that it has a center pixel.

This operation applies directly on the image; however, the convolution theorem states that convolution in the image domain is simply a multiplication in the Fourier domain. Thus, convolution filters use this fact to modify images based on frequency considerations without converting the image to its Fourier representation. Often these filters can be viewed as empirical versions of applying a frequency mask in the Fourier transform. These filters are often used in smoothing and edging operations. Figure 3.8 depicts the kernels used in such operations

[1] The square modulus of a complex number is the real number product of a complex number with its complex conjugate.

Smoothing

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Sharpening

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure 3.8—Smoothing (left) and Edging (right) Filters

The edge filter will extract edges from the image. For sharpening images, this extracted information is reapplied to the original image. Median filters are a non-linear class of filters that replace a pixel with the median value of its neighbors. Figure 3.9 illustrates this operation.

Cell C

$c_{1,1}$	$c_{1,2}$	$c_{1,3}$
$c_{2,1}$	$\text{med}(c_{ij})$	$c_{2,3}$
$c_{3,1}$	$c_{3,2}$	$c_{3,3}$

Figure 3.9—Median Filter

Median filters are often used to remove various forms of noise from images. This study considers how post-processing with these filters affects subjective image quality.

Interpolation

Interpolation methods are used to predict data where only limited data exists. In this study, interpolation is used to produce non-standard zooms of dithered images. The

images studied here are stored at resolutions greater than 72 ppi. However, after dithering the images, the high resolution pixels are averaged together to produce a 72 ppi "zoom" of the image. This lower resolution zoom is subsequently compared to the original image to image objectively and subjectively.

Some zooms (called standard zooms in this document) are easily derived from the higher resolution dithered images. For instance, a 1:1 zoom of the original image can be derived by averaging 36 pixels (a 6×6 cell) of a 432 ppi image to produce a single 72 ppi pixel.

However, non-standard zooms require more effort to derive. These zooms are identified by non-integer cell sizes. To compute such a zoom, divide the original resolution by the intended resolution (i.e. 432 ppi/ 72 ppi). This division gives the 1:1 zoom cell size (432/72 → 6×6 cell). Now, to find cell size for the zoom, divide the 1:1 cell size by the desired zoom ratio. Continuing the previous example, for a 255% zoom, compute 6/2.55=2.3529411. To circumvent the decimal portion of this cell size, the cell is enlarged by a factor k using interpolation and averaged down using a $kN \times kN$ sized cell similar to the standard zooms.

CHAPTER 4

DIGITAL HALFTONING

Halftoning algorithms use spatial resolution to compensate for insufficient color depth in digital imagery. For example, most newspaper images display only two colors: black and white. To simulate regions of gray in such images, black and white dots can be alternated frequently. At high enough frequencies, the human eye cannot detect this alternation and, instead, perceives a shade of gray between the two intensities (see "Spatial Summation" from Chapter 2). The number of simulated grays depends directly on the amount of spatial resolution (i.e. details) sacrificed for dithering. Dithering images for binary displays is often called halftoning. This study focuses on such bi-level dither algorithms. The sections that follow describe the previously documented dithering methods of interest in this study.

Thresholding

Thresholding similar to that described in Chapter 3 can be used to dither images. This method of halftoning is a pixel process that simply compares the values of the image pixels against those of a threshold pattern. Mathematically, this process can be represented by Equation 4.1.

$$Pixel\big(I(x,y),T(x,y)\big)=\begin{cases}1 & if\ I(x,y)>T(x,y)\\ 0 & if\ I(x,y)<T(x,y)\end{cases}$$

Equation 4.1—Pixel Thresholding

The threshold pattern, *T(x,y)*, is a function in the image domain. Figure 4.1 illustrates three threshold patterns along side the resulting dithered image.

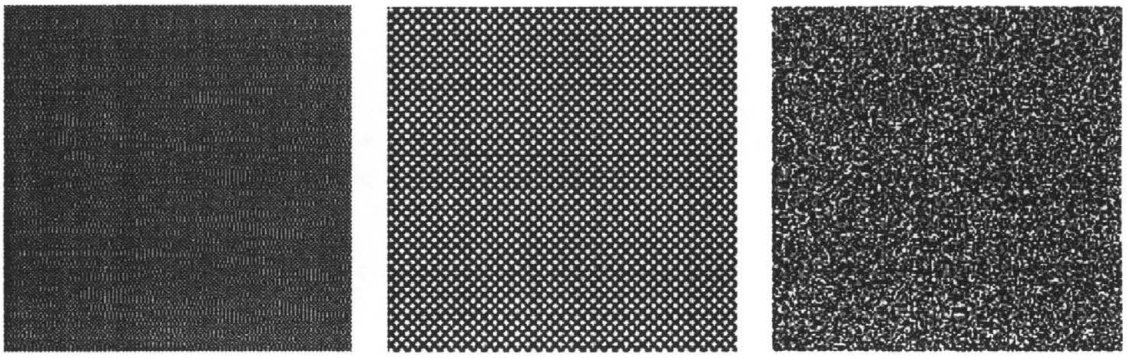

Figure 4.1—Threshold patterns

Figure 4.1a represents a threshold of constant intensity (I=127). In Figure 4.1b, the threshold intensity varies sinusoidally along both axis. Figure 4.1c illustrates a random threshold pattern. Dithering with a random threshold pattern is also known as a white noise dither (Ulichney, 1987).

Ordered Dithers

Ordered dithers work similar to threshold patterns. Both are pixel processes; however, instead of thresholding against a function, ordered dithers use dither matrices to represent intensities. Figure 4.2 illustrates a 2×2 dither matrix and its intensity patterns.

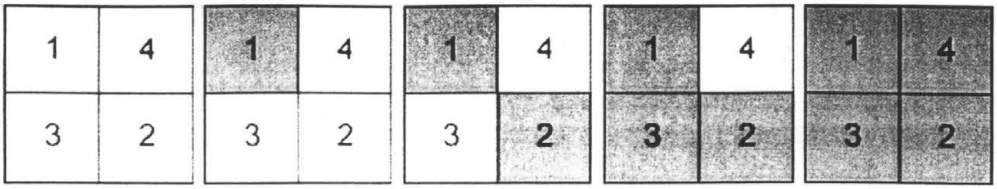

Figure 4.2—2×2 Dither Matrix

As shown, this 2×2 group of pixels can simulate 5 ($=2^2+1$) intensities. In general, intensity, *N*, is displayed by activating the pixels in the dither matrix with order less than *N*. The ordering of the dither matrix determines the nature of the dither.

Clustered-dot ordered dithers use dither matrices ordered to produce adjoining pixels for each intensity level. Often, these dither matrices grow in a spiral from the center of the matrix similar to the growth of a classical halftone dot. Figure 4.3 illustrates this point with an example of a 3×3 clustered-dot ordered dither matrix.

7	4	8
3	1	5
6	2	9

Figure 4.3—Clustered-dot Ordered Dither Matrix

For this type of dither, care must be taken to produce dither matrices which do not introduce moiré effects for regions of certain intensities. Figure 4.4 displays a pattern that will produce offending vertical artifacts in image regions of 33% gray.

8	1	6
4	2	7
6	3	9

Figure 4 4— Moiré Inducing Dithering Pattern

Clustered dithers are widely used in printing because many printing devices cannot accurately display "stray" pixels produced by dispersed-dot ordered dithers.

Dispersed-dot ordered dithers use dither matrices with more isolated pixels to represent increasing intensities. Bayer (1973) devised a method for generating dispersed-dot dither matrices of even size. Equation 4.2 gives Bayer's recursive relation.

$$Given\ D_2 = \begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix}\ and\ U_n = \begin{bmatrix} 1 & \Lambda & 1 \\ \mathrm{M} & \mathrm{O} & \mathrm{M} \\ 1 & \Lambda & 1 \end{bmatrix},\ then$$

$$D_n = \begin{bmatrix} 4D_{n/2} + U_{n/2} & 4D_{n/2} + 3U_{n/2} \\ 4D_{n/2} + 4U_{n/2} & 4D_{n/2} + 2U_{n/2} \end{bmatrix}$$

Equation 4.2—Bayer Disperse-dot Dither Recurrence Relation

Figure 4.5 shows four intensities of a 4×4 disperse-dot dither.

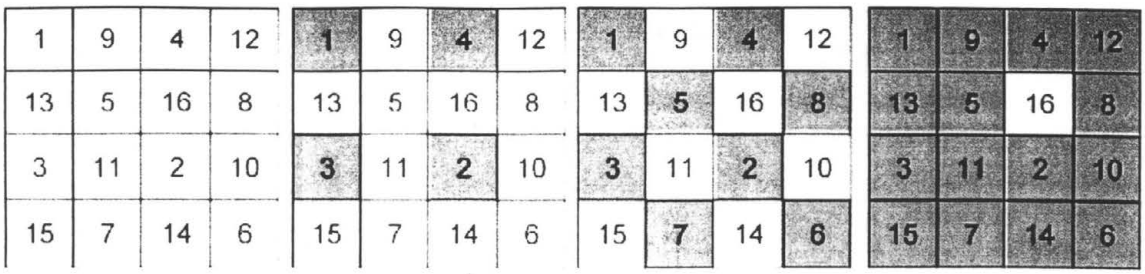

Figure 4.5—Dispersed-dot Dither Patterns

Dispersed-dot dithers are preferred over clustered-dot dithers because they produce better images if no limitations exist in the display hardware. Many printers do not handle dispersed dot dithers well.

Error Diffusion

Floyd and Steinberg first presented the idea of error diffusion in 1975 (Ulichney, 1987). Error diffusion resembles dithering in that both methods produce images that sacrifice spatial resolution to simulate extra intensity levels; however, the rational for error diffusion is quite different. Error diffusion methods are neighborhood processes that distribute errors due to intensity depth reductions to surrounding pixels. These errors are the difference between the exact pixel value and the approximated value actually displayed (Ulichney, 1987). The distribution of these errors helps to retain image details by preserving image information.

Floyd-Steinberg error diffusion is the simplest of these algorithms and distributes errors downward and to the right of the pixel in question. Figure 4.6 illustrate how the algorithm passes error compensations on to future pixels.

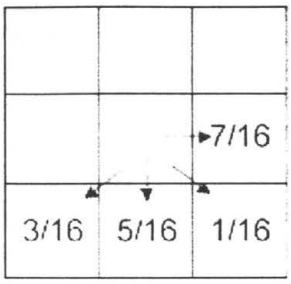

Figure 4.6—Floyd-Steinberg Distribution of Error

The above representation of the error distribution is known as the *error filter*. As can be seen in Figure 4.7, Floyd-Steinberg error diffusion produces desirable results.

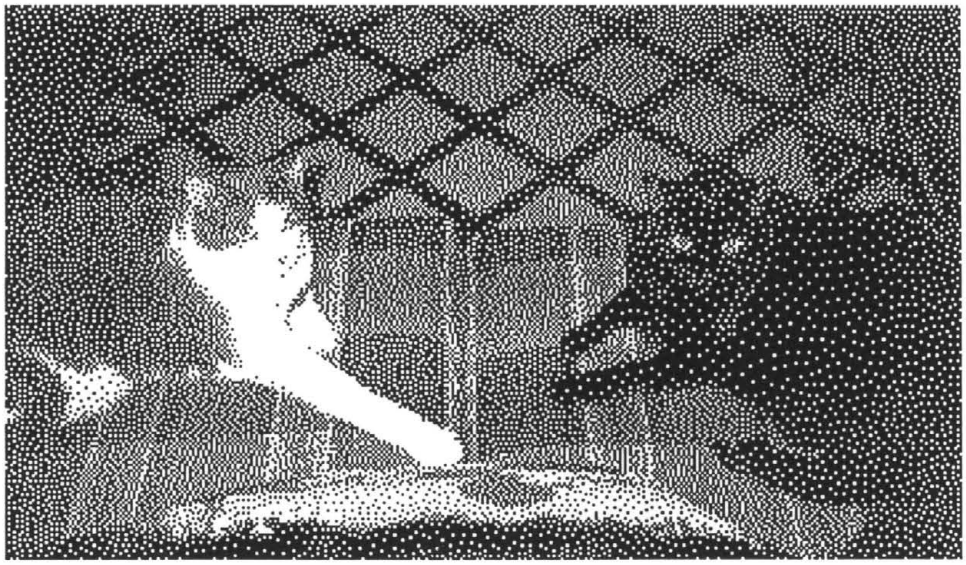

Figure 4.7—Floyd-Steinberg Error Diffusion

However, Ulichney (1987) argues that error diffusion methods typically suffer from "correlated artifacts" in gray level patterns and "transient behavior" along image edges.

CHAPTER 5

DATA COMPRESSION

In its rawest form, image information typically occupies a large space in a computer's primary or secondary memory. However, this information usually exhibits redundancies of some kind. Data compression involves the removal of these redundancies. The following summarizes Storer's (1988) ideas on data compression:

> Data compression is the process of *encoding* a body of data D into a smaller body of data ΔD so that it is possible for ΔD to be decoded back to D or to an acceptable approximation of D.

This definition alludes to the two primary types of data compression: lossless and lossy. Lossless data compression is characterized by the fact that no data is lost in the decoding process. However, lossy data compression only produces an acceptable approximation of the source data during the decoding process. Similarly, image compression methods transform image information into a form that reduces redundancies in pixel information without losing the ability to accurately reconstruct the image.

Theory

An alphabet is a set of symbols, $A = \{a_1, a_2, \text{K}, a_n\}$, that can be combined to convey something meaningful (information). An information source is defined by an alphabet and the probabilities with which the symbols in the alphabet occur in the source. The alphabet and the probabilities of an information source must satisfy the following:

$$\sum_{i=1}^{n} P(a_i) = 1 \quad where \;\; 0 \le P(a_i) \le 1 \quad \forall i$$

Equation 5 1—Probability Condition for an Information Source

A *zero-memory information source* satisfies the added condition that adjacent symbols in the source are statistically independent. In other words, a symbol emitted from the source cannot be predicted using the symbols that precede (or follow) the symbol.

Entropy is a measure of information content (Shannon, 1948). Specifically, it describes the average number of bits actually needed to convey the information of an original source. Informally, low entropy implies that some state occurs often in a system. In other words, the state is redundant and contributes less overall information to the system. Most of the time, a viable estimate of this quantity is not available. However, for the grayscale images of this study, entropy can be approximated using graylevel histograms. For an image of $H \times W$ pixels, the probability of a graylevel, g, is given by:

$$P(g) = \frac{\nu(g)}{MN}$$

Equation 5.2—Probability of a Graylevel Occurrence in an Image

In this equation, $v(g)$ is the frequency with which a graylevel occurs. This value is readily available from the image histogram. Using Equation 5.2, image entropy, H_e, can be approximated for an n-bit per pixel image as:

$$H_e = \sum_{g=0}^{2^n-1} P(g) \log_2\left(\frac{1}{P(g)}\right)$$

Equation 5 3—Image Entropy

This equation assumes that no correlations exist between the graylevels of the image or between the pixels that compose the image (Gonzales, 1987).

Compression ratios (Ruth and Kreutzer, 1972) are common image compression metrics based on entropy. Compression ratios compare an n-bit image's original size to its compressed counterpart. Formally, this quantity is defined as:

$$C = \frac{n}{H_e}$$

Equation 5.4—Theoretical Compression Ratio

This quantity describes the theoretical limit that a compression method can achieve. Empirically, it can be determined by dividing an image's original size by its compressed size.

This discussion of entropy relies on the assumption of the zero-memory information model. However, this model is a generic view applicable to all information sources. Images (especially binary images) naturally exhibit repetitions of individual symbols and groups of symbols. Thus, the zero-memory model constrains them. Image

encoding methods can achieve higher compression ratios if they account for such repetitions before considering the statistical distribution of symbols. The following sections describe previously documented methods of compression. The sections are ordered on how much they relax the condition of complete statistical independence for image pixels.

Statistical Encoders

Huffman codes (Huffman, 1952) are a lossless statistical encoding technique that can achieve compression rates between 20% and 90% (Cormen, Leiserson, Rivest, 1990). The mapping of source symbols to a Huffman code is based solely on the frequency of symbol occurrences in the source. In other words, it directly assumes the zero-memory condition. The method assigns shorter codes to high frequency characters and longer codes to low frequency characters. In fact, Huffman codes represent the source with the minimum possible average number of bits.

Binary trees can be used to represent Huffman codes (Held, 1987; Cormen, et al., 1990). Figure 5.1 gives an example of such a representation. In this figure, the leaves of the tree are marked with a character and the probability of its occurrence. The labels on the inner nodes of the tree denote the sum of the character (leaf) probabilities in their respective sub-trees.

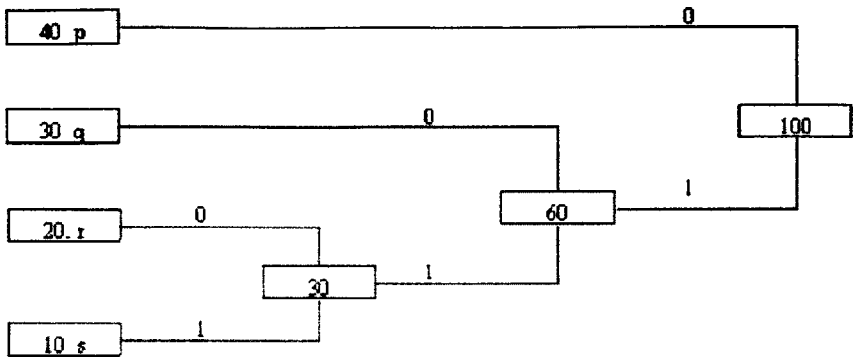

Figure 5.1—Example of a Huffman Tree

Run-Length Encoding

Run-length encoders (RLE) are a class of lossless compression algorithms that replace runs of repeated (individual) symbols with codes describing the run more concisely. As a simple example, consider the string "aaaaabbbbccccccde". A very basic RLE method could replace this string with a codeword such as "a4b3c5d0e0". In this code, the numbers imply copies of the previous letter. This example exhibits a compression ratio of 1.7 (=17/10).

Bi-level images benefit greatly from RLE compression since they often contain large runs of "on" and "off" pixels. In fact, fax machines use similar methods to decrease the amount of information transmitted over telephone lines (Russ, 1999). However, the above example also exhibits an important weakness of many RLE methods: data expansion. Notice that each run of one doubles in size after encoding. The checkerboard dithering algorithms of this study naturally contain many runs of one; thus, such a simple method would be detrimental.

When compared to other classes of compression, RLE methods often perform the weakest in terms of compression ratios. However, they are only meant to account for symbol repetition redundancies. As a result, RLE methods are often used as a precursor to another compression, usually a statistical method like a Huffman code. Run length encoders do have a significant advantage over other methods: localized decoding. RLE decoders can translate codewords individually and independent of the system's other codewords. Other image compression schemes cannot achieve this ability without considerable drops in performance, if at all. As a result, image processing applications can exploit this RLE property to reduce the footprint of image information in primary memory at relatively low computational costs.

Dictionary Encoding

Dictionary compression algorithms use look up tables (called dictionaries) to map input strings to indices codes. These look up tables store more information than the individual symbol runs of RLE methods since they track symbol groups from the start of the encoding process. However, like RLE methods, these algorithms are lossless. Figure 5.2 illustrates how such a method could process data from a bi-level image.

Typical string of bi-level image data

```
0101010001110101110 10
01 2  30 211  3 7 8 8
```

Encoded data

Look up table

String	Code	String	Code
0	0	011	6
1	1	11	7
01	2	10	8
010	3	0101	9
0100	4	111	10
00	5	101	11

Figure 5.2—Dictionary Compression Example

In this figure, the codes are represented as decimal numbers; however, a realistic implementation would use some form of binary representation. For this simple example, a four bit binary code would suffice. The codes are generated by first initializing the LUT with codes for the characters of the source alphabet. Subsequent iterations of the encoding loop generate new codes for source substrings previously unencountered. An unencountered substring is the largest substring with a LUT entry plus the following source character.

Dictionary compressors build their codes adaptively and do not store the dictionaries in the compressed data. This occurs in both the encoding and decoding processes. As a result, dictionary compression methods cannot perform localized decoding in the fashion of RLE methods. Despite this fact, these algorithms perform well on computer generated images. The GIF file format uses Limpel-Ziv dictionary encoding to achieve average compression ratios of 4 to 1 (Russ, 1999). Limpel and Ziv

popularized dictionary methods with papers in 1977 and 1978[2]. Welsh (1984) later created an improved implementation of these methods.

Frequency Transform Compressors

Frequency transform compression (FTC) methods rely heavily on the properties of the human visual system (HVS) and Fourier analysis. Discrete versions of Fourier analysis allow complex waveforms (i.e. image signals) to be represented as a truncated (infinite) series of simpler signals via the discrete Fourier transform (DFT) or the discrete cosine transform (DCT). Compression is achieved by omitting "irrelevant" information from the coefficients of such transforms (Dougherty, 1994). This activity results in a low pass filtering of the image data (Jahne, 1993). Unlike the previous methods, FTC algorithms sacrifice information to achieve compression. In general, FTC methods rely on studies of the HVS to determine expendable or "irrelevant" information.

FTC algorithms operate well on photographic images and achieve very high compression ratios relative to the previously discussed methods. However, blocking artifacts often accompany this compression since many FTC algorithms process information in increments of 8×8 or 16×16 cells to reduce computation time (Jahne, 1993). JPEG images use FTC methods (via the DCT) as their primary source of compression.

---

[2] Entitled:
"A Universal Algorithm for Sequential Data Compression,"
IEEE Transactions on Information Theory, V23, No.3, pp.337-343, May 1977.
"Compression of Individual Sequences via Variable-Rate Coding,"
IEEE Transactions on Information Theory, V24, No.5, pp.530-536, Sep. 1978.

CHAPTER 6

NEW METHODS

This chapter describes the methods that are the subject of this study. The Checkerboard Bank Sort and Reverse Diffusion halftoning algorithms are the patented intellectual property of Robert Maxwell Case (1993, 2002). The NOTA compression method also belongs to Case (2002). They are summarized here with his permission and consent. The RBE compression method is an original product of this research. It was developed through the study of Case's methods and that of digital halftoning in general.

Case Halftoning Methods

Classical (analog) halftoning uses primitives known as cells and dots. Cells determine the spatial resolution displayed by an image and range in size from 1/85 to 1/150 of an inch (Case, 1998). Classical halftone dots grow from the center of cells and coincide with an image's intensity resolution. A cell's graylevel is simply the area of the dot divided by the area of the cell. Since the dots can grow continuously, the intensity resolution in classical halftoning is only limited by the device used to display the image.

Unfortunately, similar growth of digital halftone dots using discrete pixels leads to unwanted artifacts. Several methods existing methods attempt such growth to no avail

(Bowers & Bowers, 1990; Crinion, 1987; Kotera, 1985; Tada & Kaoru, 1988). These methods fail because ***groups*** of digital pixels are more akin to classical halftone cells than they are to classical dots (Case, 1998).

Case's methods exploit this property using bi-level digital checkerboard cells. In their initial states, these checkerboard patterns represent 50% gray. To represent more intense grays, white pixels in the checkerboard are simply turned black. Likewise, lighter grays are produced when black pixels are turned white. This pixel placement process corresponds to the growth of the classical halftone dot in a cell. Figure 6.1 illustrates this idea using four 8×8 cells. The upper right cell is one shade of gray darker that the upper left and lower right cells and the lower left is one shade lighter.

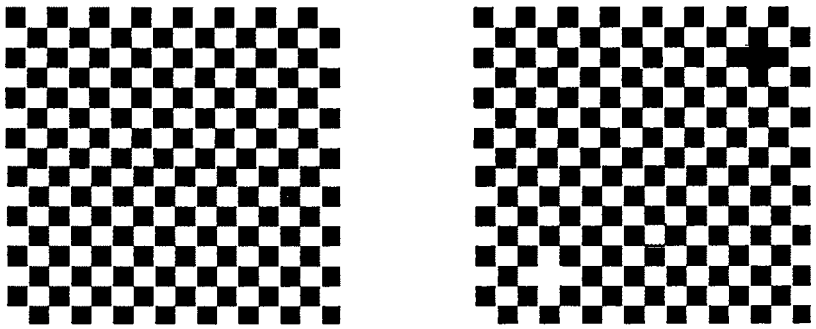

Figure 6 1—Graylevels for Checkerboard Cells

Case's methods differ only in the manner used to turn pixels on and off in the checkerboard. The significance of the 50% checkerboard can be seen in the many existing dithers that artificially attempt to produce such patterns (Bayer, 1978; Bowers & Bowers, 1990; Crinion, 1987; Kotera, 1985; Tada & Kaoru, 1988). By using these checkerboards as their natural base, Case's dithers can achieve "smoothness" not present

in other methods. This smoothness can be attributed to edges being nearly undetectable on checkerboard cells.

Figure 6.2 generalizes the cells operations of Case's methods.

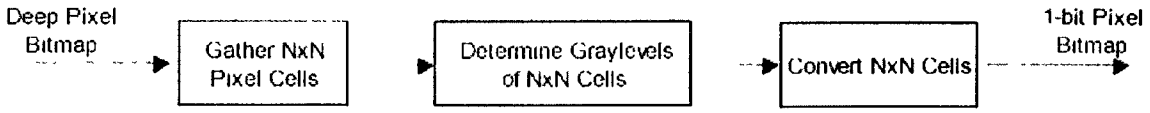

Figure 6 2—General Processes on Cells

In this figure, the term "deep pixel bitmap" refers to a bitmap that uses k bits to store each pixels, where k is typically eight or greater. In addition, N is some power of two, since a 2×2 cell is the smallest cell that can represent a checkerboard. Extracting pixels from bitmaps to form cells is an operation that is implementation specific; however, it is the same for each of Case's methods. These cell operations differ from the previously mentioned pixel and neighborhood processes in that they operate on groups of pixels rather than single pixels. The following sections describe Case's algorithms in a systematic fashion.

Checkerboard Bank Sorting

The Checkerboard Bank Sorting (BS) algorithm uses a sorting mechanism to manipulate pixels in the checkerboard cells. The sort occurs on one of the two sets of pixels present in the bi-level 50% gray checkerboard: the white bank and the black bank. Similar to other dithering methods, the BS algorithm produces a set of distinct dither patterns. However, unlike Bayer's dispersed dither (1978) or the clustered dither described by Ulichney (1987), an ordered dither matrix is not used.

Instead, the algorithm begins by determining the graylevel of a cell extracted from the deep pixel bitmap, C_{deep}. If C_{deep} is an $\delta \times \delta$ matrix, then the total number of graylevels that it can display, $T_{graylevels}$ is:

$$T_{graylevels} = \delta^2 (2^m - 1)$$

Equation 6.1—Total Number of Graylevels for C_{deep}

In this equation, δ is a power of two and m is the pixel depth of the bitmap. The percentage of gray for the cell, G_{cell}, is simply the sum of the cell's individual pixel graylevels divided by $T_{graylevels}$:

$$G_{cell} = \frac{\sum_{i=1}^{\delta} \sum_{j=1}^{\delta} c_{ij}}{T_{graylevels}}$$

Equation 6.2—Deep Bitmap Cell Graylevel

G_{cell} is now converted into the total number of pixels that will be "on" in the bi-level cell, P_{on}:

$$P_{on} = G_{cell} \cdot \delta^2$$

Equation 6.3—Number of "on" Pixels in Bi-level Cell

P_{on} is rounded to the nearest whole pixel. This rounding introduces an error of at most half a pixel to the process. Figure 6.3 depicts this process with a section of an 8-bit bitmap.

103	126	102	106	94	135
155	143	**127**	163	**149**	169
183	**200**	221	**217**	225	240
141	183	**206**	213	**216**	228
92	**83**	108	**124**	154	176
98	71	87	69	93	106

Gcell = 0.67

Pon = 11 pixels

Figure 6.3—Illustration of Cell Graylevel Determination

The conversion of the extracted cell to a bi-level cell begins with an $\delta \times \delta$ checkerboard, $C_{bi\text{-}level}$. This bi-level checkerboard cell already has half of its pixels on; thus, to determine how many more pixels to turn on or off another number, $P_{changed}$ is derived:

$$P_{changed} = P_{on} - \frac{\delta^2}{2}$$

Equation 6.4—Number of Checkerboard Pixels to Change

If $P_{changed} > 0$, then B_{white} is sorted in order of descending deep pixel graylevel and $P_{changed}$ pixels are changed to black. If $P_{changed} < 0$, then B_{black} is sorted in order of ascending deep pixel graylevel and $P_{changed}$ pixels are changed to white. If $P_{changed} = 0$, then $C_{bi\text{-}level}$ remains unchanged. Figure 6.4 illustrates the conversion by continuing the previous example.

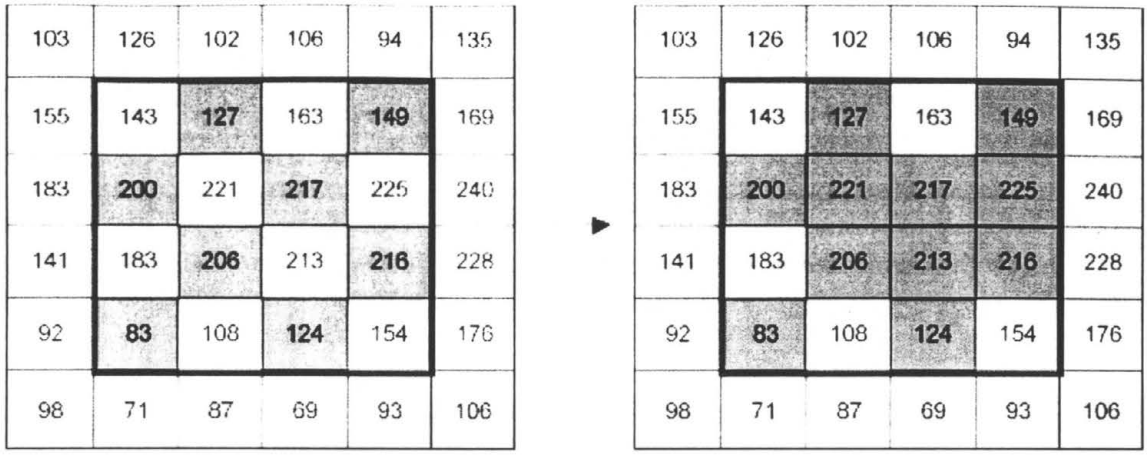

Figure 6.4—Conversion to Bi-level Cell

In this example $P_{changed} = 3$ so the pixels with graylevels 225, 221, and 213 are turned on. This process continues for each $\delta \times \delta$ cell until all cells are converted.

Reverse Diffusion

Reverse diffusion (RD) is a technique that in many ways combines aspects of dithering and error diffusion. Like the previous method, this algorithm acts on cells of pixels extracted from a deep pixel bitmap. However, unlike the BS algorithm, these cells do not result in a distinct set of $\delta \times \delta$ dither patterns. Instead, RD works with an $N \times N$ neighborhood of $\delta \times \delta$ cells to improve the performance of a typical dither by compensating for the information lost when considering the $\delta \times \delta$ cells individually. The following paragraphs describe how RD can be wrapped around an ordered dispersed-dot dither for better performance.

RD operates by pushing *partial pixel errors* down recursively to subcells. These partial pixels are the remainders generated by integer division operations present in this

scheme. The recursion stops when subcells reach the size of an ordered dither matrix. The dither matrix determines the order in which pixels are filled and is patterned after a checkerboard. The predictability of this ordered dither is used to facilitate subsequent compression steps. As previously stated, it may be replaced by similar methods including the aforementioned BS algorithm.

Before describing this algorithm, the following notation is developed. Let $C_{\log_2(N),0}$ be an $N \times N$ matrix of pixel graylevels where N is a power of two (the subscripts are described shortly). Define $Q(C_{r,\omega_r})$ as an operation that quarters C_{r,ω_r} into an ordered matrix of subcells (also matrices) as follows:

$$Q(C_{r,\omega_r}) = \begin{bmatrix} C_{r-1,\omega_{r-1}+3} & C_{r-1,\omega_{r-1}} \\ C_{r-1,\omega_{r-1}+1} & C_{r-1,\omega_{r-1}+2} \end{bmatrix} \quad , \quad where\, \omega_{r-1} = 4\omega_r$$

Equation 6 5—Cell Quartering Operation

In this notation, r describes the number of times that Q can be applied to C_{r,ω_r} before reaching a matrix of 1×1 subcells (the graylevel values themselves). The ω_r subscript represents the order imposed on these subcells by Q.

Now, let D_δ be a $\delta \times \delta$ dither matrix where $\delta < N$ is also a power of two. Since the smallest possible checkerboard is $D_2 = \begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix}$ (filling pixels in white to black order), D_δ can be recursively defined by:

$$D_\delta = \begin{bmatrix} 4\left(D_{\frac{n}{2}} - U_{\frac{n}{2}}\right) + 3U_{\frac{n}{2}} & 4\left(D_{\frac{n}{2}} - U_{\frac{n}{2}}\right) \\ 4\left(D_{\frac{n}{2}} - U_{\frac{n}{2}}\right) + 1U_{\frac{n}{2}} & 4\left(D_{\frac{n}{2}} - U_{\frac{n}{2}}\right) + 2U_{\frac{n}{2}} \end{bmatrix}$$

Equation 6.6—Recursive Definition of a White to Black Checkerboard Dither Matrix

This matrix is equivalent to the transpose of the matrix given by Bayer (1978). Note that $R = \log_2(N) - \log_2(\delta)$ is the number of times that Q must be applied to C_{r,ω_r} so that C_{r,ω_r} is divided into subcells of size δ.

The RD algorithm begins by gathering and storing the graylevel sums of the entire $N \times N$ cell and R levels of subcells. In general, these sums are denoted as S_{r,ω_r} and defined as follows:

$$S_{r,\omega_r} = \left\{ \sum_{k=4\omega_r}^{4(\omega_r+1)-1} C_{r-1,k} \;\middle|\; 0 \le \omega_r < 4^{\log_2(N)-r} \text{ and } \omega_r \in \mathbf{Z}^+ \right\}$$

Equation 6.7—Definition of Subcell Sums

With this notation, the sums of interest are all those with an r subscript ranging from $log_2(N)$ to $log_2(\delta)$.

Once the subcell sums are gathered, the cell conversion process begins by determining the value of a whole pixel (versus a partial pixel), B:

$$B = 2^b - 1$$

Equation 6.8—A "Whole" Pixel

In this equation, b is the pixel depth of the deep pixel bitmap. Now, to find the number of whole pixels, W, which will be "on" in the bi-level cell, $S_{\log_2(N),0}$ is divided by B.

$$W = \frac{S_{\log_2(N),0}}{B}$$

Equation 6.9—Number of Whole Bi-level Pixels for a Deep Pixel Cell

Remember, $S_{\log_2(N),0}$ is the sum of all of the graylevels in the cell (e.g. $C_{\log_2(N),0}$). W is rounded to the nearest whole pixel since this first division of the RD process is a floating point division. Similar to the BS algorithm, this rounding introduces at most a half pixel error. W specifies the number of "on" pixels in $C_{\log_2(N),0}$'s corresponding bi-level cell, but not which pixels.

Therefore, another set of divisions is performed on the four ordered subcells produced by $Q(C_{\log_2(N),0})$. However, these are integer divisions and both the quotients (whole pixels), W_{r,ω_r}, and remainders (partial pixels), P_{r,ω_r}, are stored:

$$W_{\log_2(N)-1,0} = \frac{S_{\log_2(N)-1,0}}{B} \qquad P_{\log_2(N)-1,0} = S_{\log_2(N)-1,0} \bmod(B)$$

$$\text{M} \qquad\qquad \text{M}$$

$$W_{\log_2(N)-1,3} = \frac{S_{\log_2(N)-3,0}}{B} \qquad P_{\log_2(N)-1,3} = S_{\log_2(N)-3,0} \bmod(B)$$

Equation 6.10—Subcell Whole and Partial Pixels

Taken together, W and P represent the number of "on" pixels as a floating pointing number. However, it is not possible to fill a pixel partially. Thus, these integer divisions

generate a discrepancy between the number of "on" pixels determined by the two recursion levels. This discrepancy is termed a shortfall and is defined by:

$$\sigma_{r,\omega_r} = W_{r,\omega_r} - \sum_{k=4\omega_r}^{4(\omega_r+1)-1} W_{r-1,k}$$

Equation 6.11—Definition of an RD Shortfall

To remedy this error, one whole bi-level pixel is added to σ subcells in order of decreasing $P_{r-1,\omega_{r-1}}$. Thus, σ of the W_{r,ω_r} values are incremented by one. The partial pixels amounts, P, determine which of the subcells is most deserving of an extra shortfall pixel.

This process is carried out recursively in the order of ω_r for each b-bit graylevel subcell of $C_{\log_2(N),0}$ until level R is reached. At this level, the subcell size equals δ and its corresponding bi-level cell is filled with $W_{\log_2(N)-R,\omega_r}$ pixels according to D_δ (the checkerboard dither matrix). At this point, the question of "which pixels" is answered. A deep pixel cell is converted when all of its $\delta \times \delta$ sized subcells have been dithered.

The above generalization of reverse diffusion in terms of sets and matrices greatly facilitates the implementation of the algorithm in a programming language; however, it does little to clarify its nature. Thus, the following example is given to illustrate how RD can be applied over an ordered dither. Suppose $C_{3,0}$ is the 8×8 cell of 8-bit pixels in Figure 6.5.

Raw 8-bit Data

156	148	140	132	132	165	173	140
132	132	132	140	181	165	148	140
148	140	189	165	165	165	165	173
173	165	140	132	148	148	165	148
140	107	123	132	173	189	156	148
165	90	173	173	156	132	123	74
132	148	181	148	107	99	115	173
181	173	115	82	57	123	156	156

N = 8
delta = 2
R = 2

Figure 6.5—An Arbitrary 8×8 RD Cell

RD begins by determining the graylevel sums of the entire cell and two levels of subcells. These sums are then translated into the number of whole and partial pixels they represent. In Figure 6.6, the *P*s actually represent the numerator of the fraction $\frac{P}{B}$ where $B = 255$ (a whole pixel in this case). This fact illustrates the meaning of "partial" in the term "partial pixel errors".

Raw 8-bit Data

156	148	140	132	132	165	173	140
132	132	132	140	181	165	148	140
148	140	189	165	165	165	165	173
173	165	140	132	148	148	165	148
140	107	123	132	173	189	156	148
165	90	173	173	156	132	123	74
132	148	181	148	107	99	115	173
181	173	115	82	57	123	156	156

N = 8
delta = 2
R = 2

Figure 6.5—An Arbitrary 8×8 RD Cell

RD begins by determining the graylevel sums of the entire cell and two levels of subcells. These sums are then translated into the number of whole and partial pixels they represent. In Figure 6.6, the *P*s actually represent the numerator of the fraction $\frac{P}{B}$ where $B = 255$ (a whole pixel in this case). This fact illustrates the meaning of "partial" in the term "partial pixel errors".

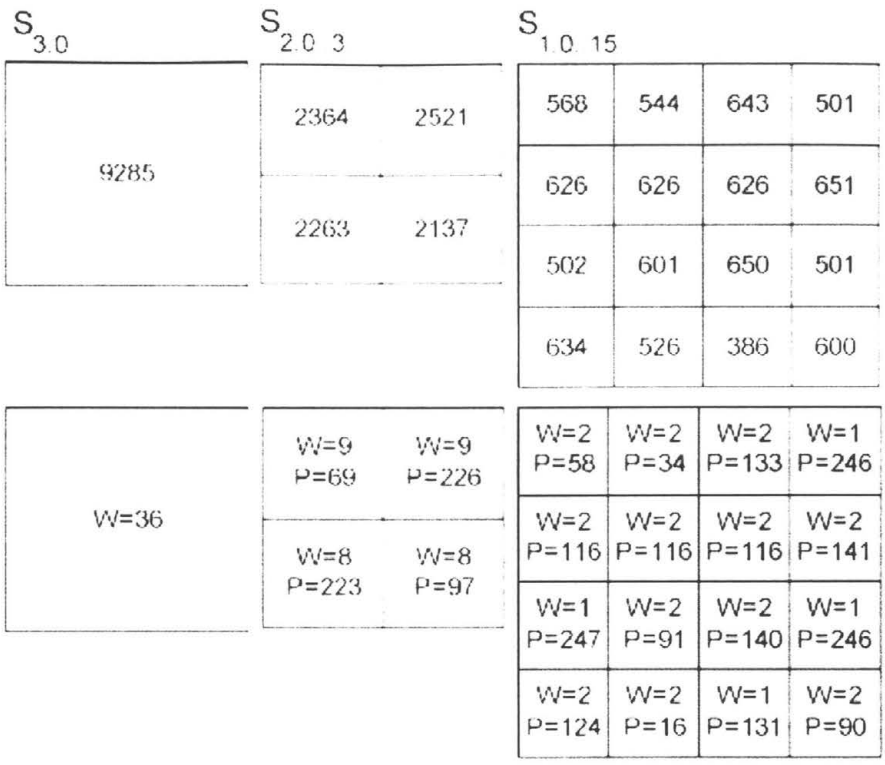

Figure 6.6—Subcell Sums, Whole and Partial Pixels

At the end of the RD process, $C_{3,0}$ must contain 36 total pixels. However, some of these pixels become "lost" in the integer divisions used to compute the W_{r,ω_r} of each subcell. Thus, shortfalls are determined for each (C_{r,ω_r} , $Q(C_{r,\omega_r})$) tuple and are used to "replace" the lost pixels. For Figure 6.6,

$$\sigma_{3,0} = 36 - (9 + 8 + 8 + 9) = 36 - 34 = 2$$

As a result, a pixel is added to the $C_{2,0}$ and $C_{2,1}$. It is important to note that a shortfall is applied before the RD process proceeds to the next subcell level (if this were not true, the pixels would stay "lost"). Figure 6.7 depicts this new situation.

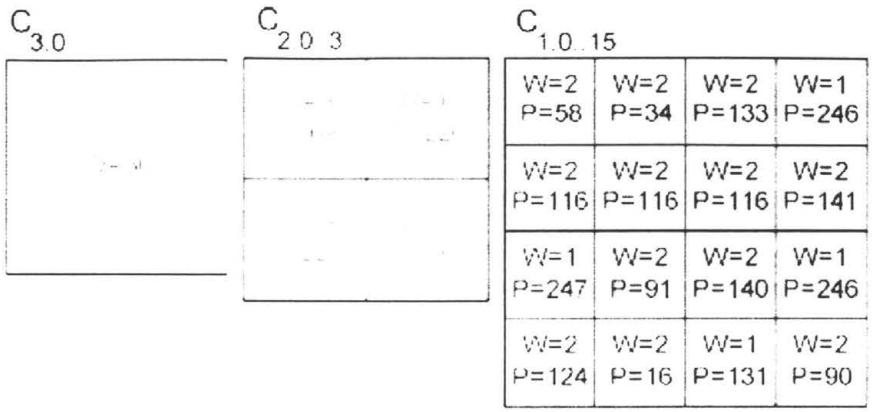

Figure 6.7—After First Shortfall Increment

Now, the process continues to recursively descend into $C_{3,0}$ until subcell level 1 is reached. Figure 6.8 illustrates how this process occurs. Notice that a tie in *P*s occurs on the last increment of Figure 6.8. This tie is resolved using the ordering imposed by the quartering operation.

Now that the lost partial pixels have been recovered, W_{1,ω_1} bi-level pixels are activated in each of the 2×2 C_{1,ω_1} subcells according to the corresponding 2×2 dither matrix. This activation of pixels is the final step to convert the deep pixel input cell to a bi-level RD output cell.

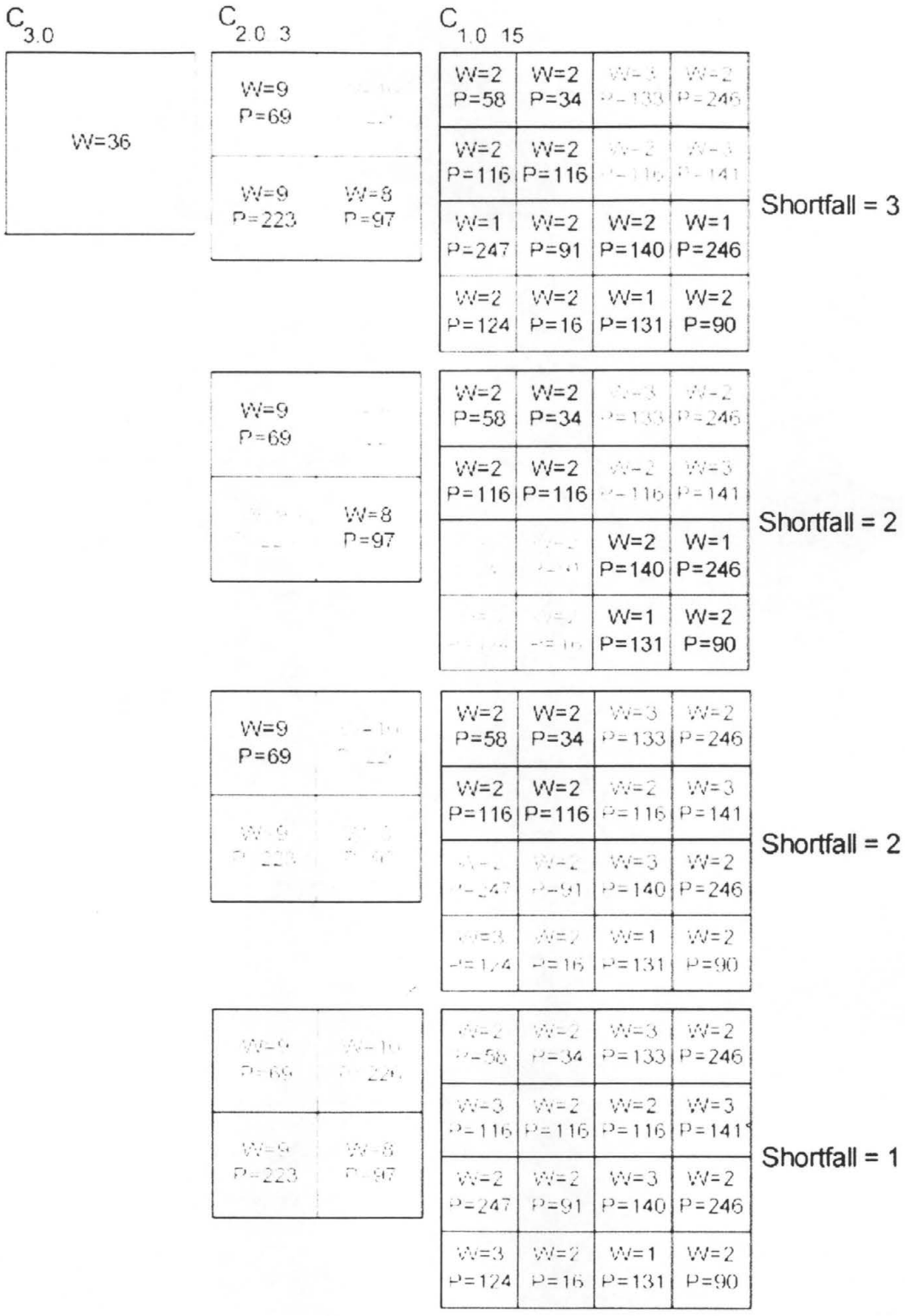

Figure 6.8—Subsequent Shortfall Increments

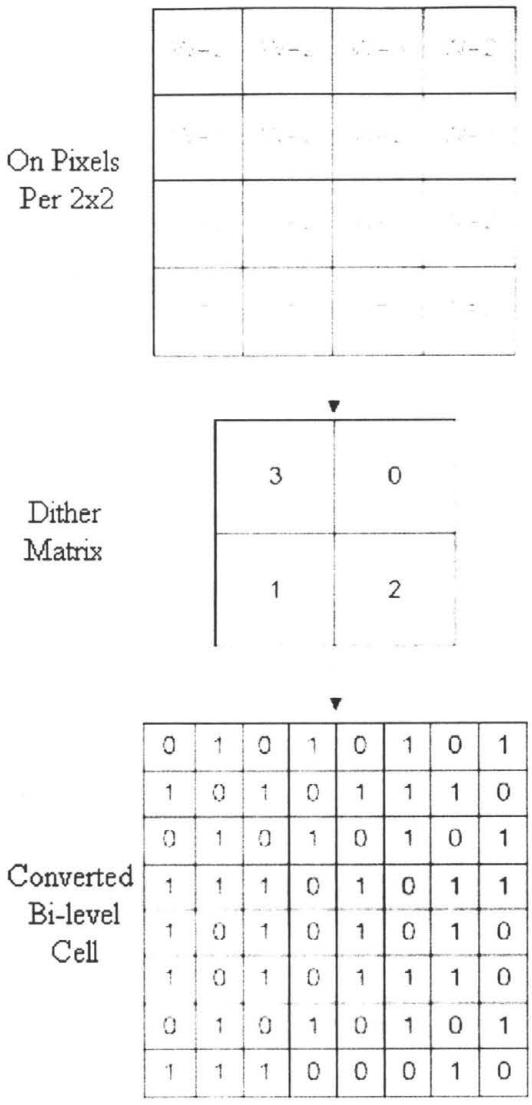

Figure 6.9—2x2 Ordered Dither to Bi-Level Pixels

New Compression Methods

The previously discussed compression methods (Chapter 5) are well documented in the literature of information theory and image encoding. The following sections introduce two new image encoding schemes. The first of them is a form of run length

encoding. The other is an attempt to produce a two-dimensional version of run length encoding for digitally halftoned images. Both methods produce codes that can be subsequently compressed using other methods.

Case's NOTA Encoding

NOTA ("None of the above") compression is a proprietary method suggested by Case (2002). It improves on other RLE methods by considering the nature of bi-level images and uses variable length codes to represent runs of pixels. Since bi-level pixels only have two states (0 or 1), pixel runs can be represented by sequences of zeros where the size of the sequence describes the run as a power of two. Ones in the sequence are used as delimiters specifying the end of a run.

The length of a variable NOTA code depends on the range in which a run of pixels exists. These ranges are represented by subcodes with lengths, a_l, defined in the following manner:

$$Given \;\; a_0 = 1, \;\; a_1 = 2, \;\; a_2 = 5,$$

$$\forall n \geq 3 \quad a_n = 8(n-2)$$

Equation 6.12—NOTA Subcode Lengths

Table 6.1 shows the code lengths for runs ranging from 1 bit to 16,777,216 bits in terms of the subcode lengths.

Variable Code Length	*Subcodes Lengths*	*Run Range*
1		[1, 1)
3	1+ a_0+	[2, 5)
8	1+2+ a_0+a_1+	[5, 32)
16	1+2+5+ a_0+a_1+a_2+	[32, 256)
32	1+2+5+8+ a_0+a_1+a_2+a_3+	[255, 65536)
56	1+2+5+8+16+ a_0+a_1+a_2+a_3+a_4+	[65535, 16777216)

Table 6 1—NOTA Variable Code Lengths

As Equation 6.12 implies, subsequent code lengths grow by the next multiple of eight.

NOTA encodes a run by comparing the run's length to the maximum length allowed for each variable sized code. If a_T is the last term of a set of subcodes lengths (highlighted in red in Table 6.1), then this maximum run length is highest number that can be represented by a_T bits minus one. The subcode produced by a_T copies of "0" is used to signal that a run cannot be encoded by the current variable size code; thus, this subcode signals that the encoder must continue to the next variable size code length. However, before trying the next variable size code, the encoder first subtracts $2^{a_T} - 1$ from the run's length and outputs a_T copies of "0." This decremented run value is used in the next comparison (which is similar to the one just described). The encoding procedure continues until a run's length is less than the maximum length allowed by the corresponding variable sized code. At this point, the run value (after being decremented

by each $2^{a_T} - 1$) is encoded as a binary number with a_T bits, the a_T bits are output, and the processes begins again for a new run.

Figure 6.10 illustrates how a typical run is encoded via NOTA.

Run Length: 3021

Compare Value	Comparisons	Code
3021	3021 > 1 (2^1 - 1) output "0"	0
3020	3020 > 3 (2^2 - 1) output "00"	000
3017	3017 > 31 (2^5 - 1) output "00000"	00000000
2986	2986 > 255 (2^8 - 1) output "00000000"	00000000 00000000
2731	2731 < 65535 (2^16 - 1) output 2731 as 16-bit binary "0000101010101011"	00000000 00000000 00001010 10101011

Figure 6.10—NOTA Encoding Procedure

NOTA's decoding procedure reads the series of variable length subcodes from a bit stream until a "1" is encountered. If s_T is subcode containing the first "1," then subcodes $s_0 \ldots s_{T-1}$ contain a_i copies of "0" (where $i = 1 \ldots T-1$). To decode a bit stream, the NOTA decoder initializes the run length, L, to the following sum:

$$L_0 = \sum_{i=1}^{T-1} 2^{a_i} - 1$$

Equation 6.13—Decoder Run Length Initialization

The remaining length of a run is specified by adding the number represented in binary by s_T to L_0. Figure 6.11 offers an example of the NOTA decoding process for clarification.

Code: 0 00 00000 00000000 0000101010101011

s_0 s_1 s_2 s_3 s_4

Process	Run Length
L = 0	0
s0 = "0", increment L by 2^a0 - 1 (1)	1
s1 = "00", increment L by 2^a1 - 1 (3)	4
s2 = "00000", increment L by 2^a2 - 1 (31)	35
s3 = "00000000", increment L by 2^a3 - 1 (255)	290
s4 = "0000101010101011", add integer(s4) to L	3021

Figure 6.11—NOTA Decoding Procedure

Recursive Block Encoding

Recursive Block Encoding (RBE) is an original product of this study that achieves compression by considering the properties of halftoned images. It is so named because the process recursively identifies blocks of homogenous $\delta \times \delta$ dither patterns used to convert a deep pixel image to a binary bitmap.

Generally, an arbitrary $\delta \times \delta$ cell of binary pixels can exist in one of 2^{δ^2} states. However, all practical dithering methods use less than this maximum set. For instance, Bayer's ordered dither (OD) (1978) and the clustered dither (CD) described by Ulichney

(1988) only use δ^2+1 states. Case's BS dither produces $2^{\frac{\delta^2}{2}+1}-1$ states. If ${}^{M}P_{\delta}$ denotes the number of states produced by dither method *M*, then the three aforementioned dithers can be described by ${}^{OD}P_{\delta}$, ${}^{CD}P_{\delta}$, and ${}^{CHBS}P_{\delta}$, respectively.

Since dither methods use less than the maximum number of $\delta\times\delta$ states, each possible $\delta\times\delta$ dither cell can be mapped to a fixed length code of $\lceil \log_2(P_{\delta}) \rceil$ bits. As a result, these cells can be represented by less than the δ^2 bits originally needed to describe the individual pixels of the cell. In fact, for the 2×2 case the OD and BS methods produce 5 and 7 patterns respectively. Since three bits will encode each of these numbers, a 25% savings is produced. Savings for the 4×4 case are more dramatic at 68% and 43% respectively. This mapping of $\delta\times\delta$ patterns to $\lceil \log_2(P_{\delta}) \rceil$ bit codes removes a first order redundancy in which $\delta\times\delta$ cells are represented with more bits than is necessary. Further compression cannot be practically achieved in this manner because dither cells larger than 4×4 pixels sacrifice too much of the deep pixel bitmap's resolution.

For normal images, neighboring pixels tend to be similar. However, dither matrix halftoning methods produce binary images where $\delta\times\delta$ sized cells are similar. Thus, further compression can occur if *blocks* of these $\delta\times\delta$ cells are considered.

For N equal to some power of two, an N×N block of pixels will contain $\left(\frac{N}{\delta}\right)^2$ δ×δ sized subcells. As previously mentioned, each δ×δ subcell exists as one of ${}^{M}P_{\delta}$

patterns; thus, each N×N block has $\left({}^{M}P_{\delta}\right)^{\left(\frac{N}{\delta}\right)^2}$ possible states. By recursively parsing through the N×N block, a unique variable length code can be produced to encode the block's state. This variable length code is built from the fixed length codes used to encode each δ×δ pattern.

Before describing the recursive block codes created for this study, the following precondition is set:

- For any sequence of *m*-bit fixed pattern codes, the code of *m* copies of the binary digit "1" is reserved for use as a *recursion delimiter*.

Figure 6.12, Figure 6.13, and Figure 6.14 illustrate the RBE scheme for four different 64×64 pixel cells with δ=2. Figure 6.12 shows the 5 patterns of ordered dither along with the fixed length code assigned to them. Figure 6.13 shows the bit streams used to encode the blocks in Figure 6.14. The "0s", "1s", "2s", "3s", and "4s" of Figure 6.14 correspond to the fixed length codes of the ordered dither. The encoding procedure is now described.

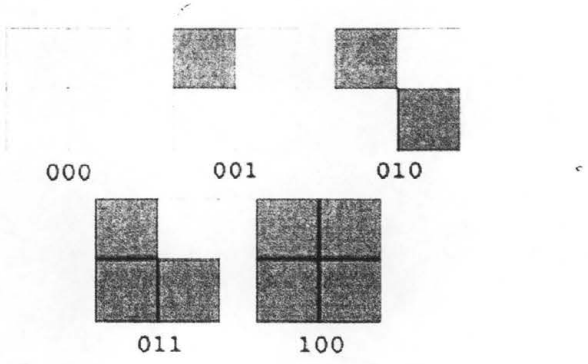

Figure 6.12—Fixed length codes for 2×2 ordered dither

a)
```
000
001
```

b)
```
001
001
001
000
010
```

c)
```
001
001
111 001 000 001 000
010
111 010 010 001 001
```

d)
```
010
001
111 111 001 111 000 001 001 000 000 001 001
010
000
001
011
010
001
000
001
001
011
001
000
001
010
```

Figure 6.13—Variable length block codes (arranged vertically by chunks)

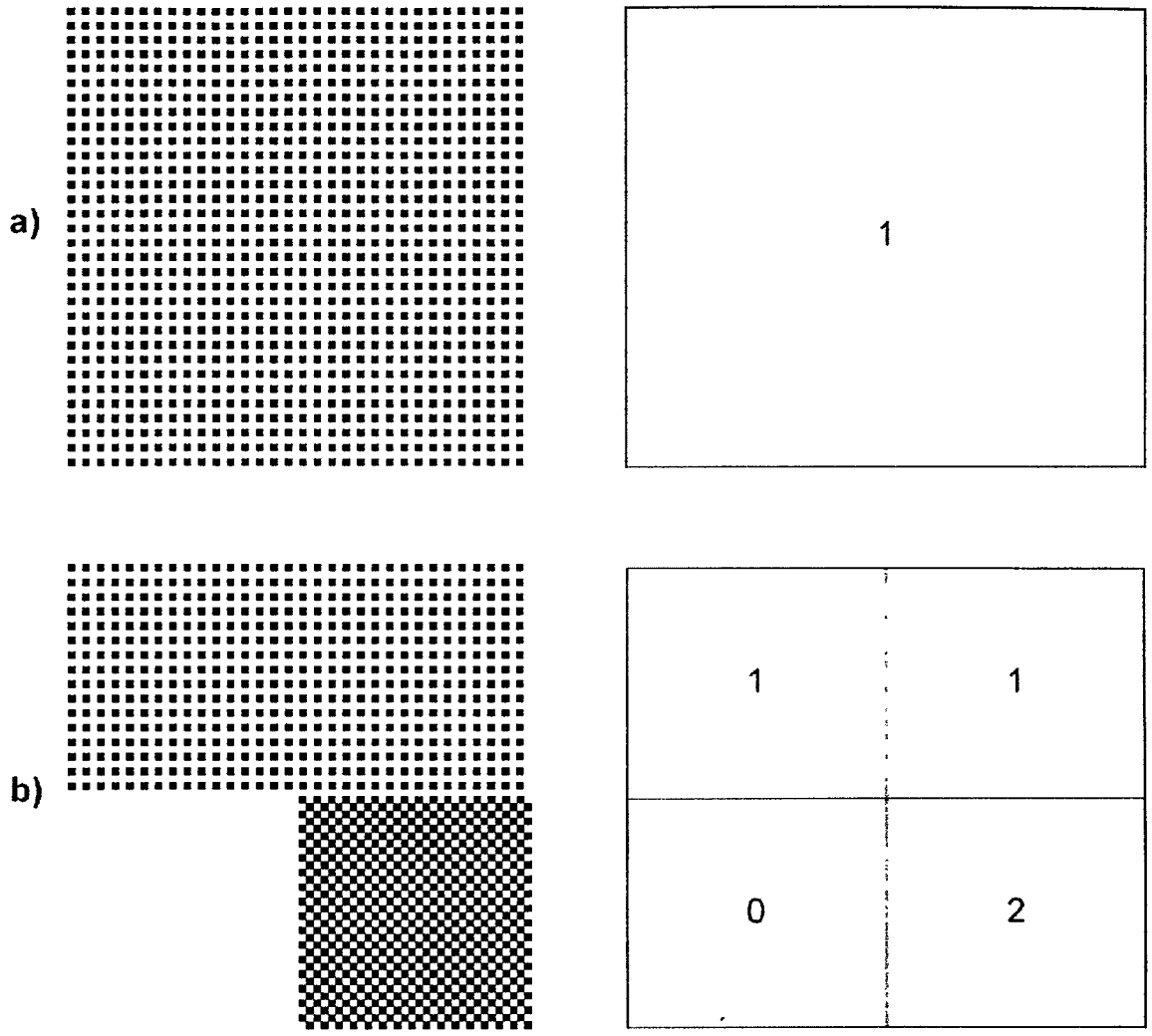

Figure 6.14 a & b—64×64 Pixel Block Example

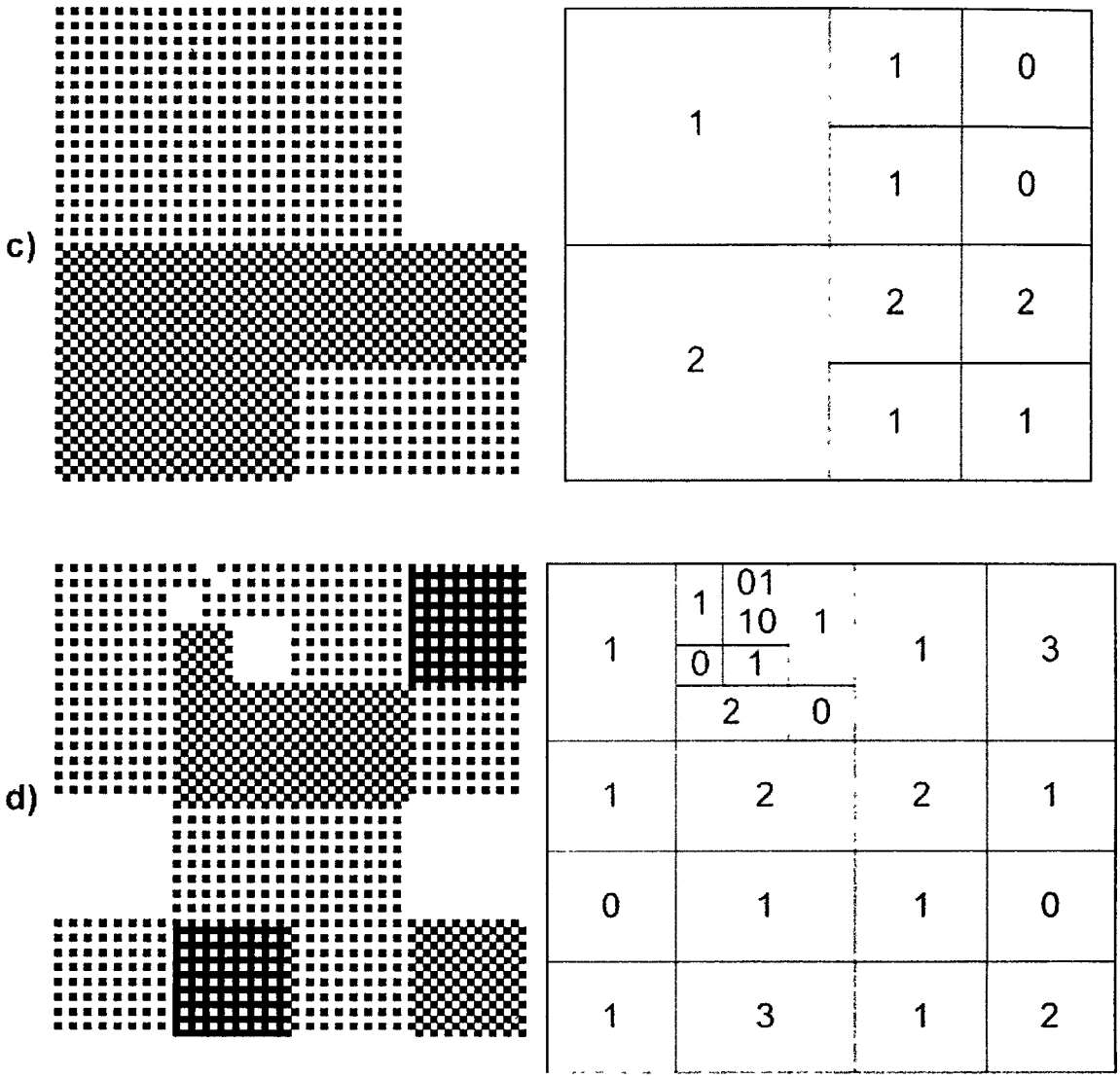

Figure 6.5 c & d—64×64 Pixel Block Example (Continued)

The codes of Figure 6.13 are arranged vertically for readability, but, in reality, they would exist as a stream of bits in a file. Each code is composed of a *prefix* and several *chunks*. The *prefix* is simply an integer represented by m bits (where $m = \lceil \log_2(P_\delta) \rceil$). In Figure 6.13 a-d, the prefixes are the first three bits at the top of each code. Following each prefix are 4^{prefix} chunks. A *chunk* is a variable length subcode used to represent a

$2^{(\log 2(N)-prefix)} \times 2^{(\log 2(N)-prefix)}$ sub-block of the N×N block. Qualitatively, a prefix code describes the "granularity" of an N×N block by specifying the size of the largest 2^k sub-block of homogenous δ×δ cells where $k = \log_2(N) - prefix$. The chunks, on the other hand, identify the characteristics of each individual "grain" in the N×N block while specifying them in "English reading order."

Prefix codes are determined by identifying the largest block of homogeneous δ×δ cells. The chunks of an N×N block are output according to the following recursive procedure:

1. If the $2^C \times 2^C$ sub-block is homogeneous in terms of the δ×δ cells that compose it, then the fixed length code for the δ×δ cell is output as the chunk's value.
2. Else if $2^C = \delta$, then the code corresponding to the δ×δ cell is output.
3. However, if the $2^C \times 2^C$ sub-block is a heterogeneous mixture of δ×δ cell, then the recursion delimiter is output and the $2^C \times 2^C$ sub-block is quartered. The process (1, 2, and 3) is then recursively applied to each of the quartered sub-blocks of size $2^{C-1} \times 2^{C-1}$. The recursion continues until a sub-block of homogenous δ×δ cells is found or until the δ×δ level is reached.

In this procedure, sections 1 and 2 are the recursive escape conditions while section 3 makes recursive calls. Figure 6.14 can now be described.

In Figure 6.14a, each of the 2×2's composing the block are the same. Thus, "000" is output as the prefix because there is no need to descend into the sub-blocks and "001" is output as the required chunk

Figure 6.14b is similar; however, each of the first level of sub-blocks (size 32×32) is homogeneous (instead of the entire block). Thus, "001" is the prefix since one level of recursion is needed to reach the largest homogenous structure and four (4^1) non-delimited chunks describe the four homogeneous sub-blocks.

Figure 6.14c illustrate a situation in which two 32×32 sub-blocks are the largest homogenous structures of the entire block. As a result, "001" is output and the 32×32 homogeneous chunks are output similar to Figure 6.14b. However, homogeneity in the remaining blocks does not occur until the 16×16 level is reached. Thus, for each of these chunks, the recursion delimiter is output to signify a drop to the next recursive level and four codes are output for each of the homogenous 16×16 sub-blocks.

Figure 6.14d depicts a more "average case" in which homogeneity until the third level of recursion (16×16 sized sub-blocks). For this case, "010" is outputted followed by 16 (4^2) chunks. This example show how several recursion delimiters can be nested inside of a chunk.

The decoding of the RBE bitstream is simpler than the encoding process. The code is unambiguous because the δ×δ cell codes are fixed in size and the prefix and recursion delimiters determine exactly the number of fixed codes are present in a string. To decode an RBE string, first, read the prefix code to determine the "starting sub-block size." Once the prefix code is read, fill the N×N block (where N is the same as used in the encoding step) according to each of the 4^{prefix} chunks. When no recursion delimiter is present, the sub-block that the chunk represents is filled with $\frac{2^{\log 2(N)-prefix}}{\delta}$ copies of the

$\delta \times \delta$ cell specified by the chunk. When recursion delimiters are present, the sub-block is filled in the same manner; however, the sub-block size decreases by half as does the number of necessary copies for each encountered recursion delimiter. The chunks are mapped back to the N×N block in normal "English reading order."

The variable length codes of this particular implementation of the RBE idea remove a first order redundancy in which excessive bits are used to identify a $\delta \times \delta$ dither cell and a second order redundancy in which neighboring $\delta \times \delta$ cells are often similar. However, it is also designed to promote further compression using a statistical encoding method. Note how only eight possible 3-bit symbols can occur in each code. A Huffman coder that uses these eight symbols as its alphabet can provide this extra compression. Discussion of this idea is deferred until Chapter 9 where its analysis is actually carried out.

CHAPTER 7

PROJECT MANAGEMENT

The management and organization of this study is influenced by Davis's and Sitaram's concurrent software development model. This model decomposes the entire process of software development into a set of "major technical activities...and their associated states" (Pressman, 1997). It can be readily applied to research because, like software development, research can be modeled as a set of related autonomous activities. Application of a concurrent management system also provides an understandable, yet "fine grain" view of a project's monthly status.

The concurrent management scheme used in this study is not formalized in any way. However, its description is offered because it has been a useful tool. In particular, its development and application has allowed for topics from several different fields to be researched and considered simultaneously. The following sections describe this study in terms of general activities, specific tasks, and milestones.

Activities

In this chapter, the term "activity" will imply a generic, abstract process (to be applied). Thus, activities are similar to the idea of a class in object-oriented

methodologies. The term "task", on the other hand, will imply a specific instantiation of an activity. Thus, every task has a specific state and can trigger state transitions in itself or other tasks.

Figure 7.1 illustrates the generic activities occurring during this study. Each activity contains several internal nodes that describe the possible states of the activities. In this model, relations between activities are denoted when an internal node references another activity. For example, the <*IP*> activity contains one state which reference the <*LR*> activity and another state which references the <*C*> activity. Note how some activities contain a state called "other." This state adds flexibility to the specification of activities that are less linear or less predictable.

Figure 7.2 is a rough chronological mapping of the specific tasks that have occurred during the months of this study. It illustrates the concurrent nature of this project. Table 7.1 defines the milestones listed on the right side of the figure. Table 7.2 details the monthly (as of the first day of the month) status of this study in terms of active tasks.

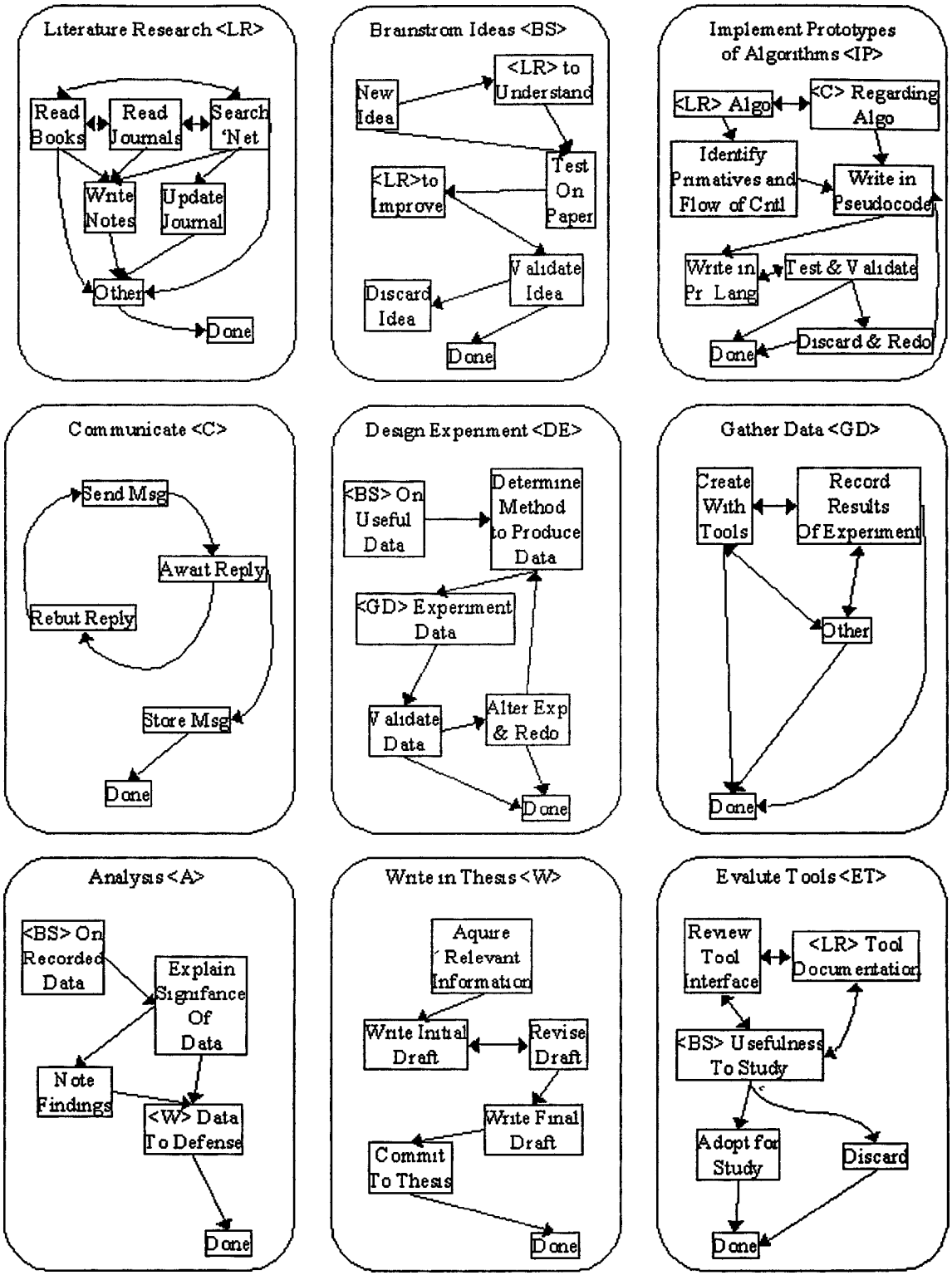

Figure 7.1—General Thesis Activities

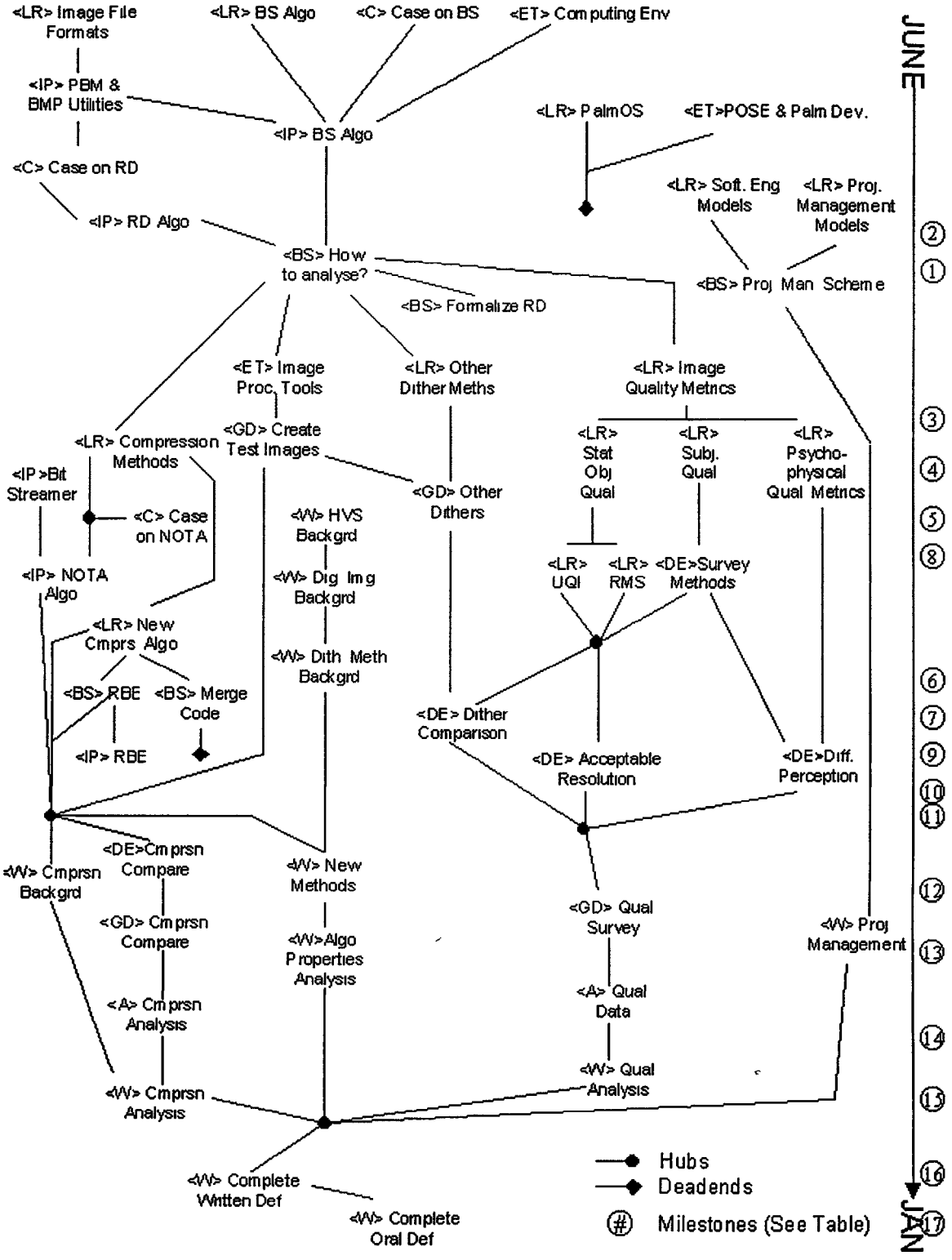

Figure 7.2—Specific Thesis Tasks

Number	*Description*	*Date Achieved*
1	Identify the "shape and direction" of the study.	Aug. 6, 2001
2	Understand and implement Case's Dithers.	Aug. 3, 2001
3	Understand other dithers.	Aug. 29, 2001
4	Understand compression methods.	Sept. 14, 2001
5	Understand Fourier analysis as it applies to image processing.	Sept. 29, 2001
6	Write most of defense background information.	Oct. 25. 2001
7	Formalize the RD algorithm.	Nov 2, 2001
8	Implement QP and NOTA algorithms.	Oct. 16, 2001
9	Conception and Implementation of RBE.	Nov. 14, 2001
10	Develop a convenient method of surveying people.	Nov. 19, 2001
11	Complete development of the experiments.	Nov. 21, 2001
12	Complete gathering of compression data.	Dec. 3, 2001
13	Complete gathering quality survey data.	Dec. 5, 2001
14	Complete analysis.	Dec. 22, 2001
15	Complete "research" section of defense.	Dec. 27, 2001
16	Complete written thesis defense.	Jan. 2, 2001
17	Complete oral thesis defense	Jan. 7, 2001

Table 7.1—Thesis Milestones

Month	*Task (State)*
June	<LR>Image File Formats (Done), <LR> BS Algo. (Done), <C> Case on BS (Rebutting Replies), <ET> Computing Environment (Done)
July	<IP> PBM/BMP Utils. (Test and Validate), <IP> BS Algo (Write in Pseudocode), <C> Case on RD (Rebutting Replies), <LR> PalmOS (Reading Books), <ET> POSE and Palm Dev. (<LR> Tool Documentation)
Aug.	<IP>PBM/BMP Utils (Done), <IP> BS Algo (Done), <LR> PalmOS (Other, no longer relevant), <ET> POSE and Palm Dev. (Discard), <C> Case on RD (Done), <IP> RD Algo (Write in Pr. Lang.), <LR> Soft Eng. Models (Reading Books), <LR> Proj Management Models (Reading Books), <BS> How to analyze? (<LR> to Improve)
Sept.	<LR> Compression Methods (Read Books), <C> Case on NOTA (Rebutting Replies), <LR> Other Dither Methods (Read Books), <LR> Image Quality Metrics (Search 'Net), <IP> Bit Streamer (Write in Pseudocode), <ET> Image Proc Tools (Done), <GD> Create Test Images (Done)
Oct.	<IP> NOTA Algorithm (Write in Pseudocode), <W> HVS Background (Revising Draft), <W> Digital Image Background (Revising Draft), <W> Dith. Meth. Background (Acquire Relevant Information), <GD> Other Dithers (Done), <LR> Stat Obj Qual (Read Journals), <LR> Subj Qual (Read Books), <LR> Psychophysical Qual Metrics (Read Journals), <DE> Survey Methods (Done)
Nov.	<LR> New Cmprs Algo (other—<BS> ideas, <C> Davis on compression), <W> Dither Method Background (Commit To Thesis), <DE> Cmprs Compare (Determine Method to Produce Data), <DE> Dither Comparison (Determine Method to Produce Data), <DE> Acceptable Resolution (Determine Method to Produce Data), <DE> Diff. Perception (Determine Method to Produce Data)
Dec.	<IP> RBE (Done), <GD> Cmprs Data (Done), <GD> Qual Survey (Done), <W> New Methods (Commit to Thesis), <W> Cmprs Background (Revising Draft), <W> Proj Management (Revising Draft), <W> Algo Properties Analysis (Acquire Relevant Info)
Jan.	<A> Cmprs Analysis (Done), <A> Qual Data (Done), <W> Complete Written Defense (other—Being Reviewed by Committee), <W> Complete Oral Defense (other—Pending)
Feb	<W> Complete Written Defense (Done), <W> Complete Oral Defense (Done)

Table 7.2—Monthly Thesis Timeline

Task Elaboration

The following sections elaborate on certain tasks from Figure 7.2 which require special consideration. The description of these tasks reveals relevant information about this study that cannot be directly considered in Chapter 8 on Research Methods

<ET> Computing Environment

Two computing environments were used in this study: a dual boot (Linux/Windows) desktop computer (PC) and a Sony Vaio laptop (LTC). Most of the work of this study occurred on the PC. Specifically, it is used for implementing algorithms, processing images, and authoring this document. The LTC is used in the same manner as the PC while traveling; however, it also serves as one of the displays used in the subjective quality assessment of Case's dithers.

Other devices used in this study include a Hewlett Packard 6300C ScanJet scanner (capable of scan resolutions up to 1,200 dpi) and a Hewlett Packard 4500N Color LaserJet printer (capable of output resolutions up to 600 dpi).

<ET> Image Processing Tools

Many existing image processing tools have been evaluated for use in this study. Table 7.3 is a list describing each of the packages that are actually used.

Tool	*Description*
GIMP	GIMP is the acronym for Gnu Image Manipulation Program. GIMP is used in this study to produce all of the enhanced images used to assess the quality of Case's dithers. It also generated the "Sinus" test image used in this study.
ImageJ	ImageJ is an image manipulation program written in Java. It is used to produce certain image manipulations that GIMP does not provide.
Octave	Octave is an open source MATLAB clone. MATLAB is a very high level programming language used for matrix processing. It is used in this study to produce some image manipulations and to calculate statistical image quality metrics.
Moray	Moray is a 3-D modeling application. It produced the "Bowl" image of this study.
NetPBM	NetPBM is a set of command line utilities that can be used to manipulate images. These utilities were used for image file format conversions, image scaling, and image enhancements.

Table 7.3—Thesis Image Processing Tools

<IP> PBM/BMP, BS, RD, Bit Streamer, NOTA, RBE

The PBM and BMP utilities were created in order to read and write to the PBM and BMP image formats. They were implemented in a Linux environment (Mandrake 7.2) using the GCC compiler and the EMACS text editor. These prototype libraries include Linux specific I/O function calls; however, they should be potable to a Microsoft environment by replacing the Linux specific code with ANSI C.

The BS and RD implementations (also prototypes) use the PBM and BMP libraries for reading and writing image information. As such, they too run only in a Linux environment.

The Bit Streamer utility is used for extracting and depositing individual bits to and from files. It is written as a C++ class using Microsoft Visual Studio 5.0. It

complies with ANSI C++ standards and operates in both Linux and Microsoft environments. It serves as the implementation basis for the NOTA and RBE compression algorithms.

<C> Case on BS, RD, NOTA

Case's methods are relatively undocumented. As a result, they were explained by Case using detailed examples. This communication process resembled the gathering of user requirements in a software engineering model. This communication directly led to the design of the prototypes used to implement the algorithms.

<BS> Formalize RD Algorithm

As mentioned, Case's RD algorithm was not documented at the start of this study (although a patent was submitted before the publishing of this document). Thus, as part of this study, the RD algorithm was formalized in terms of a more precise matrix notation. This notation is also used (partially) in the description of the RBE method and as part of the basis of the object-oriented redesign of the prototypes.

<DE> Survey Methods

The initial attempts of surveying people for subjective quality data failed. The survey was too long and test subjects were not willing or able to finish it. Thus, the survey was streamlined (with parts of it completely sacrificed) until a group of simple,

yet informative survey questions were developed. Two groups of questions failed (in terms of convenience and subject cooperation), before finding an acceptable survey.

CHAPTER 8

RESEARCH METHODS

This chapter describes the methods used to analyze the dither and compression algorithms of this study. The data generated by these methods is presented and analyzed in Chapter 9.

Test Images

Figure 8.1–Figure 8.4 are the four main test images used throughout this study. All of the images originally existed as 24-bit color BMP files; however, they were converted to 8-bit grayscale with GIMP for the purposes of this study. The images are also scaled down to a width of approximately five inches so that they can be displayed conveniently in this document.

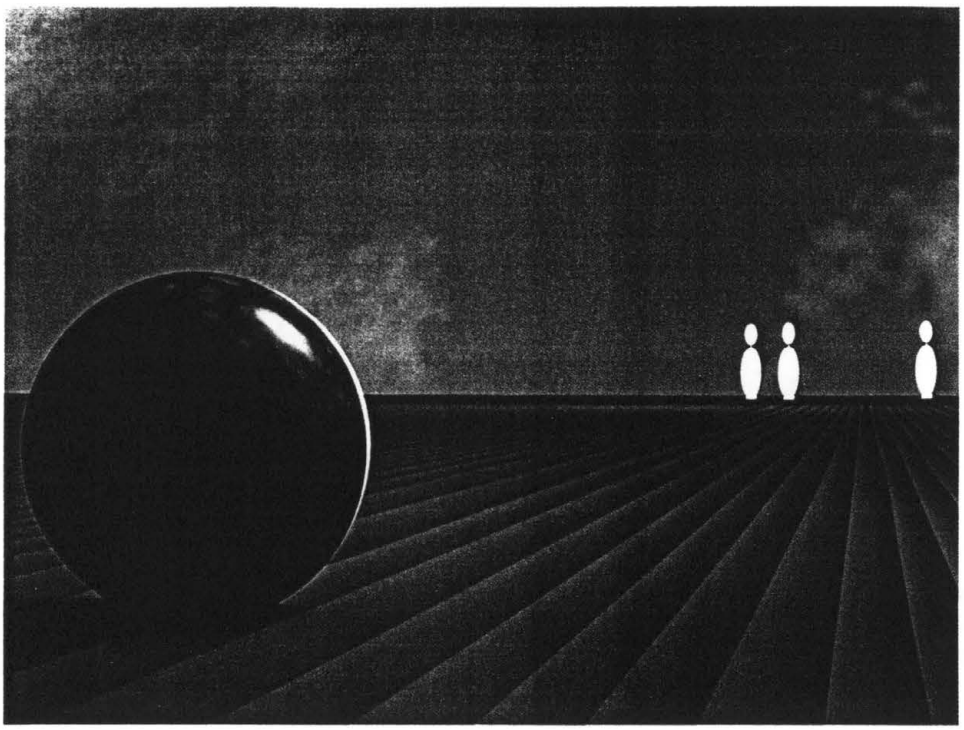

Figure 8.1—"Bowl" Test Image

Figure 8.1 is a vector image produced by Moray (subsequently called "Bowl"). Characteristic features of this image include: the depth that exists between the bowling ball and the pins; the repeating parallel patterns of the wooden floor texture; and the texture detail of the bowling balling.

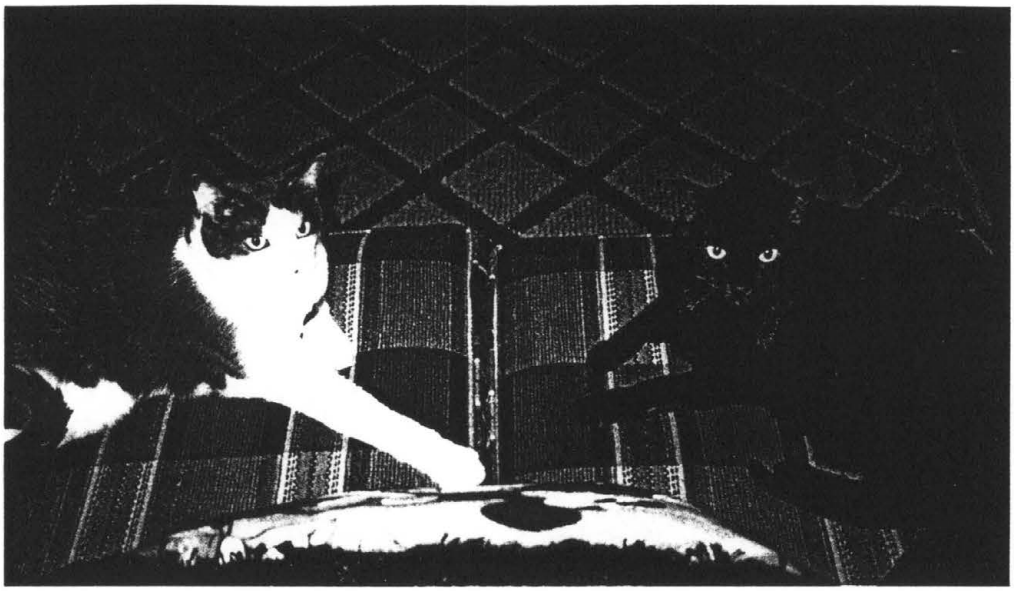

Figure 8.2—"Cats" Test Image

Figure 8.2 is a natural image originally taken with a 35mm camera (subsequently called "Cats"). The image was digitally scanned at 600 dpi using the scanner mentioned on page 76. The scanner's sharpening option was left at the default setting of "medium." Interesting features of this image include: repeating light and dark patterns in the carpet and on cushions of the couch; a spectral reflection of light from the black cat's fur; and the fine details in the face of the other cat.

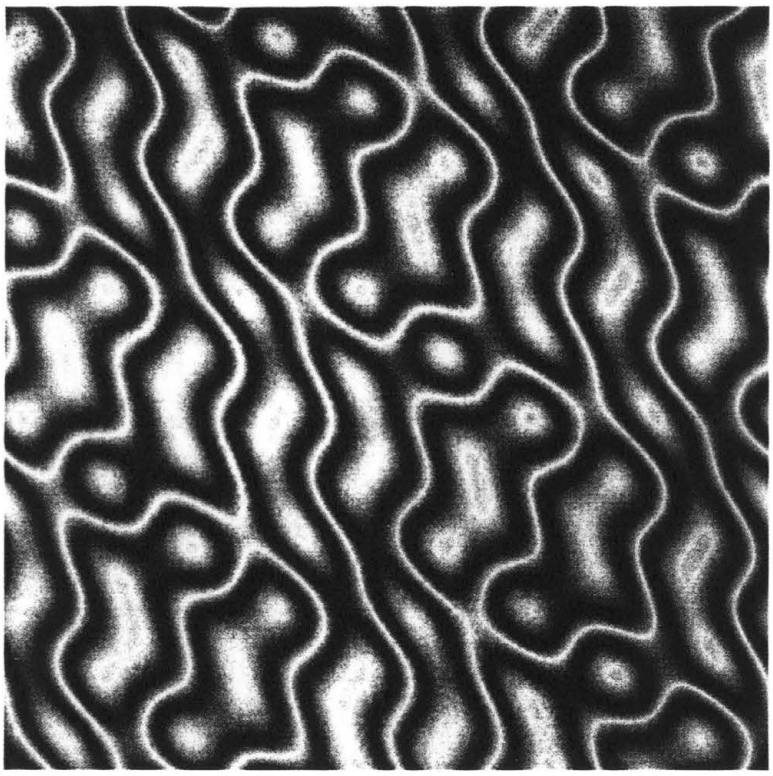

Figure 8.3—"Sinus" Test Image

Figure 8.3 is another image (subsequently called "Sinus") generated via computer with the rendering tools of GIMP. This image features soft edges on which intensities change gradually as opposed to sharply and non-uniform repetition of alternating intensity levels.

Figure 8.4—"Building" Test Image

Figure 8.4 is another natural image taken via 35mm camera (subsequently called "Building"). The scan parameters for this image are the same as the "Cats" image. This image contains many hard edges where intensities change sharply and repetition of structures in the windows of the building. This image is that of a building built upside-down.

These images have been chosen for two reasons. One, they represent "average case" images on which no preprocessing is done. Two, they display characteristics which can be observed both before and after processing.

Dither Analysis

This study analyzes Case's BS and RD algorithms in two ways. First, the general properties of these algorithms are determined along with the properties of the images that

they produce. This analysis is followed by a subjective and objective assessment of quality for images produced by the dithers. The following sections describe the data, tests, and experiments used in this assessment.

Objective Quality Metrics

The *UQI* (Zhou & Bovik, 2000) and *RMS* objective image quality metrics were gathered using MATLAB. The *UQI* data was gathered with Zhou's implementation (Zhou, 2000). The *RMS* data, on the other hand, was retrieved with an original implementation built from MATLAB's core components. This data is used along with the image survey data to determine if any correlations exist.

Image Quality Survey

This experiment is a four-part human survey that was given to two sets of people. Gonzales (1987) suggests that 20 test people be surveyed in any subjective assessment of image quality. In excess of this suggestion, a total of 25 people were surveyed using two different display methods.

Seven subjects viewed the images using the LCD screen of the Vaio laptop (page 76) as the output device. Due to the space restraints of the 12-inch LCD screen, these subjects viewed the images in a round robin fashion (comparing only two at a time). The viewing distance ranged between three and four feet. The LCD surveys required the viewing of 288 image sets and typically took 50-60 minutes to complete. This length prevented the gathering of more LCD data.

Eighteen subjects viewed the images in print form using pages output from the HP 4500N Color LaserJet printer (page 76). All of the images were printed on 20 lb white paper. Free from space constraints, these subjects viewed all of the images simultaneously and typically completed the survey in 30 minutes. Again, the viewing distance ranged between three and four feet. In addition, most of the test subjects viewed the image under the light of a 100 Watt white bulb placed roughly six to seven feet directly above the subjects. The following sections describe the details of each section of the survey. The actual forms used in the survey can be found in the appendix. As previously stated, the images related to these surveys are displayed at a width of roughly five inches for this document. Test subjects actually viewed eight-inch versions of the images.

Threshold Resolution

For this test, the BS (using 2×2 cells) and RD (using 8×8 cells over an ordered dither) algorithms were applied to 144 ppi, 288 ppi, and 432 ppi versions of the input images. The bi-level pixels of the resulting images were then averaged together to form 1:1 zooms of the image at 72 ppi. In other words, for the 144 ppi bi-level image, every 2×2 cell of pixels was averaged into one 72 ppi pixel. The 288 ppi and 432 ppi images were averaged using 4×4 and 6×6 cells, respectively. The resulting 72 ppi zoom images were shown to subjects. Figure 8.5—Figure 8.10 shows the six versions of the "Bowl" image used in the experiment.

The LCD subjects viewed 60 two image HTML pages (15 per test image) and were asked to pick which of the two images looked better. The results were subsequently tallied and images are ranked in order of which received the most votes.

The print subjects compared the images six at a time and ranked them on "visual appeal." Visual appeal is defined by the survey to be the subject's opinion of which image looked best. For the ranking scale, "6" denoted the best image and "1" marked the worst. The subjects used a 72 ppi version of the original image as a reference. In addition to ranking the images, these subjects were also asked to specify the zoom images that were acceptable (again in their own opinion) reproductions of the original images.

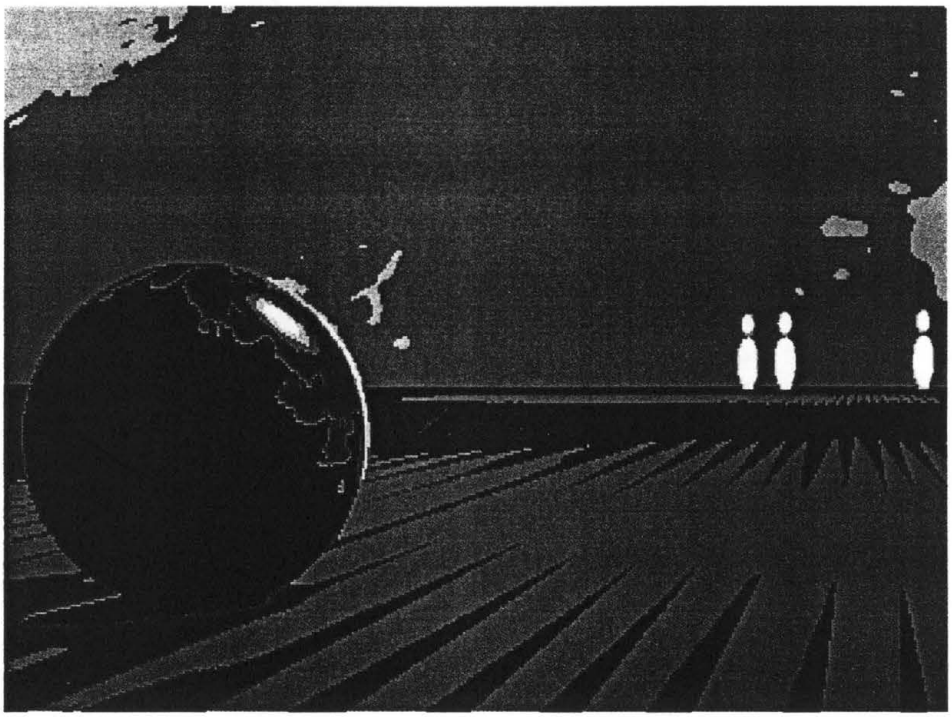

Figure 8.5—1:1 Zoom of 144 ppi 2×2 BS "Bowl" Image

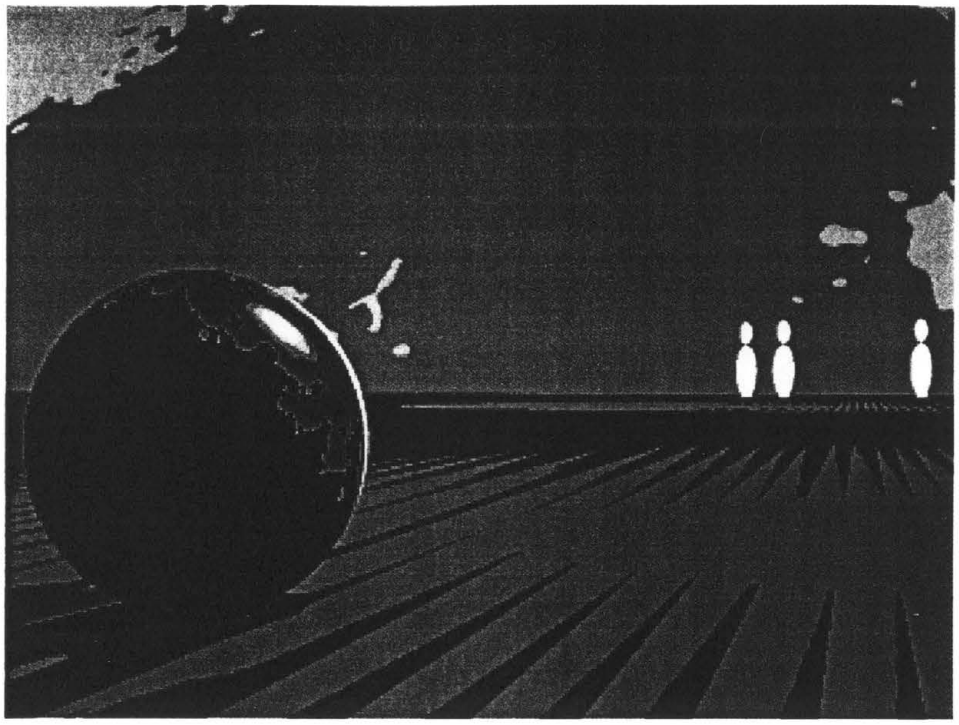

Figure 8.6—1:1 Zoom of 288 ppi 2×2 BS "Bowl" Image

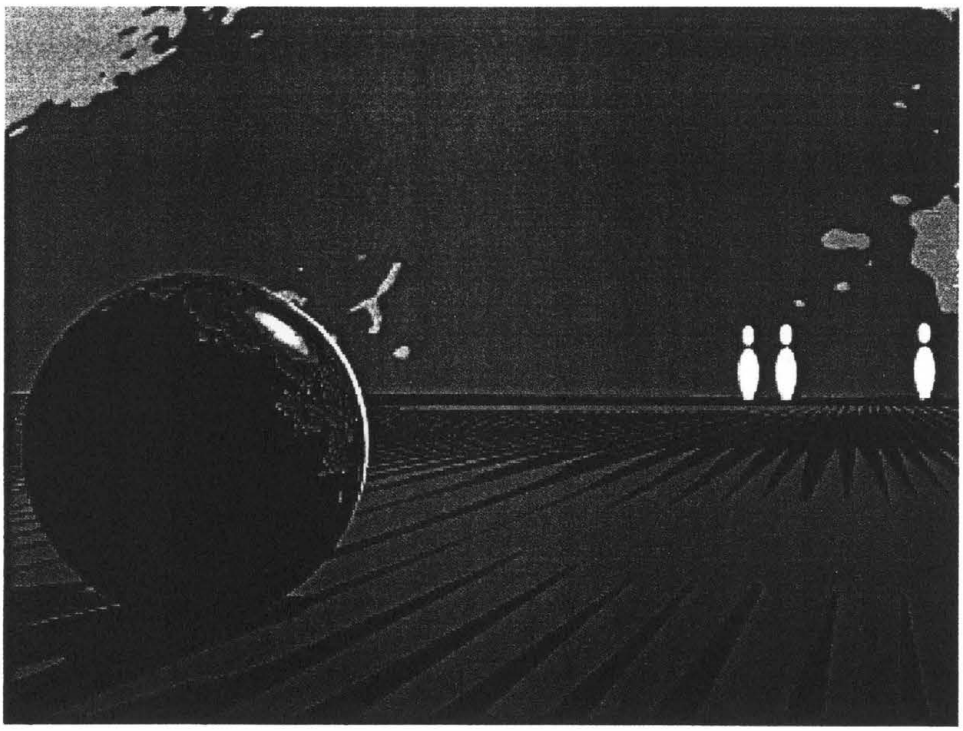

Figure 8.7—1:1 Zoom of 432 ppi 2×2 BS "Bowl" Image

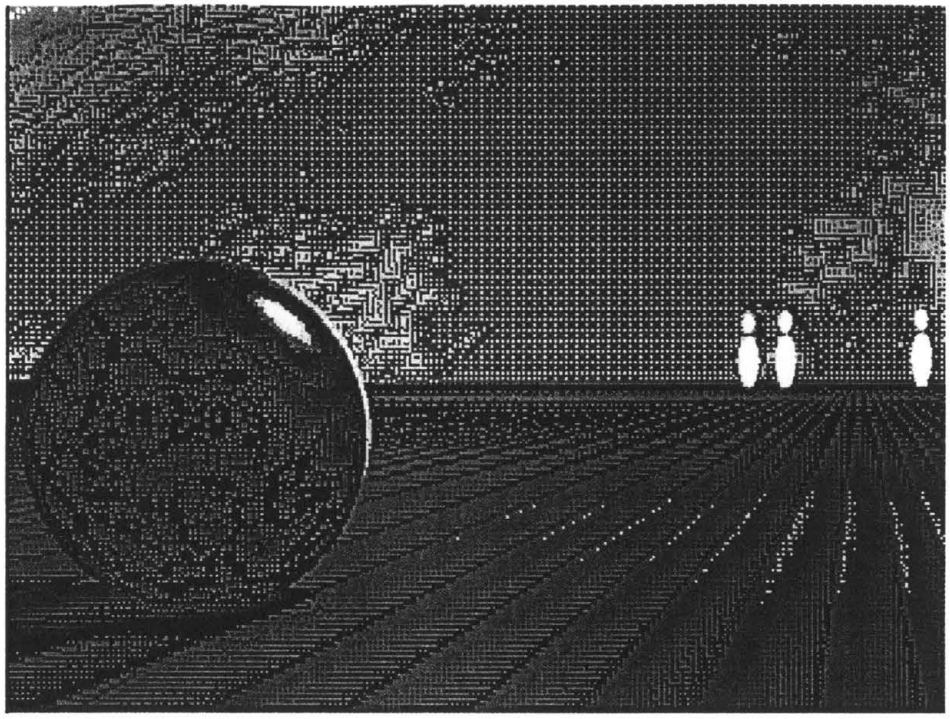

Figure 8.8—1:1 Zoom of 144 ppi 8×8 RD (on ordered dither)
"Bowl" Image

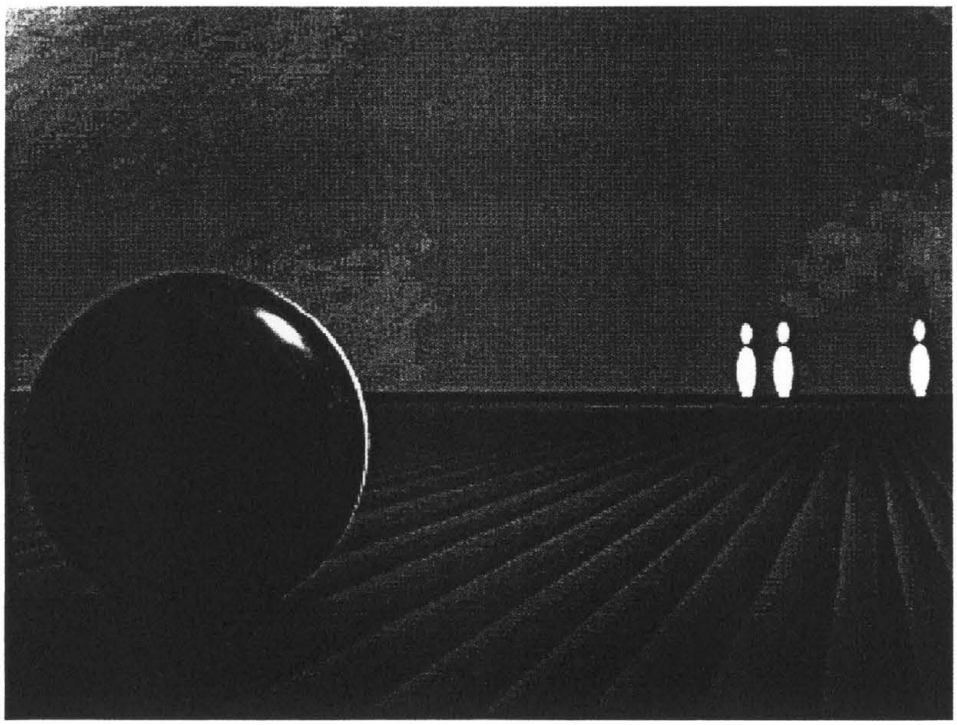

Figure 8.9—1:1 Zoom of 288 ppi 8×8 RD "Bowl" Image

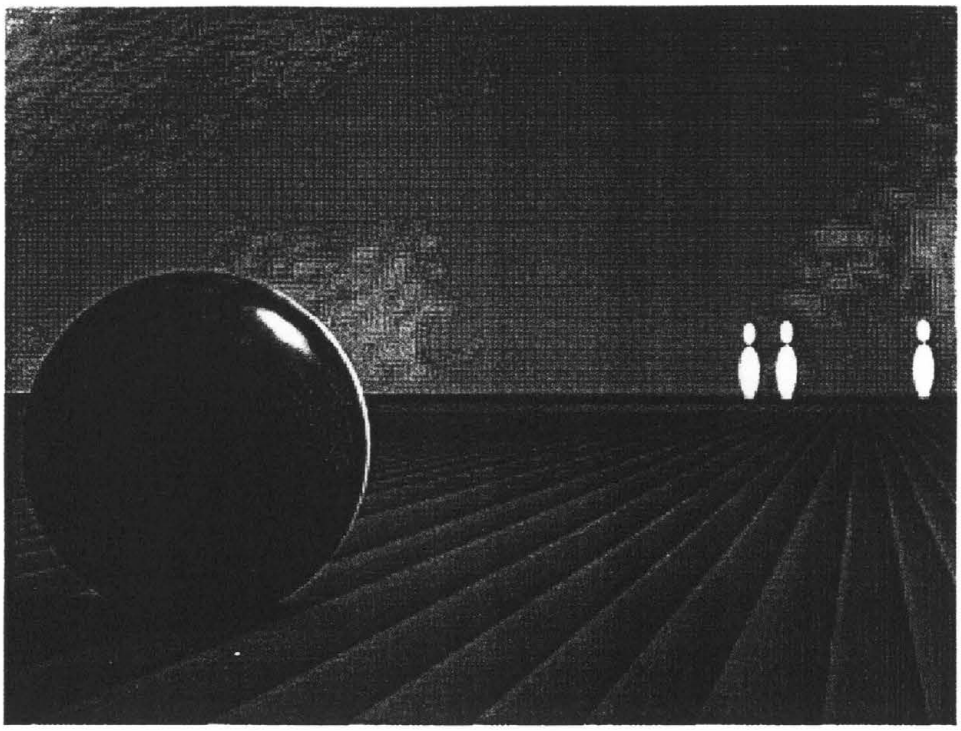

Figure 8.10—1:1 Zoom of 432 ppi 8×8 RD "Bowl" Image

Dither Comparison

For this test, Case's BS and RD dithers are compared to halftoned images produced via white noise (random) dither, clustered dither, ordered dispersed dither, and Floyd-Steinberg error diffusion. Using the precedent set by Ulichney (1987), these images are compared at 72 ppi. The LCD and print subjects were surveyed in a manner similar to the previous test.

Figure 8.11—Figure 8.16 display the "Cats" test image as an example of what the test subjects viewed.

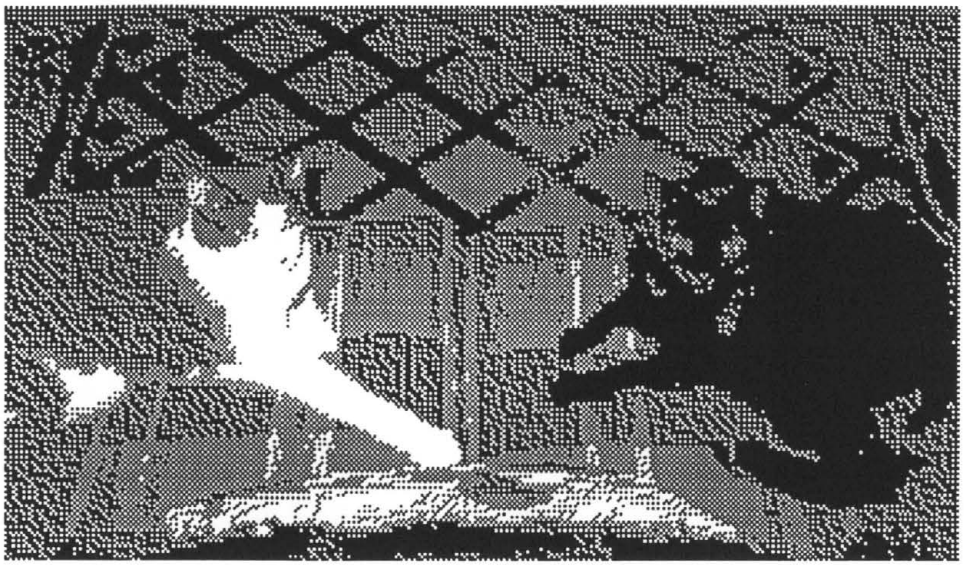

Figure 8.11—2×2 BS "Cats" Image at 72 ppi

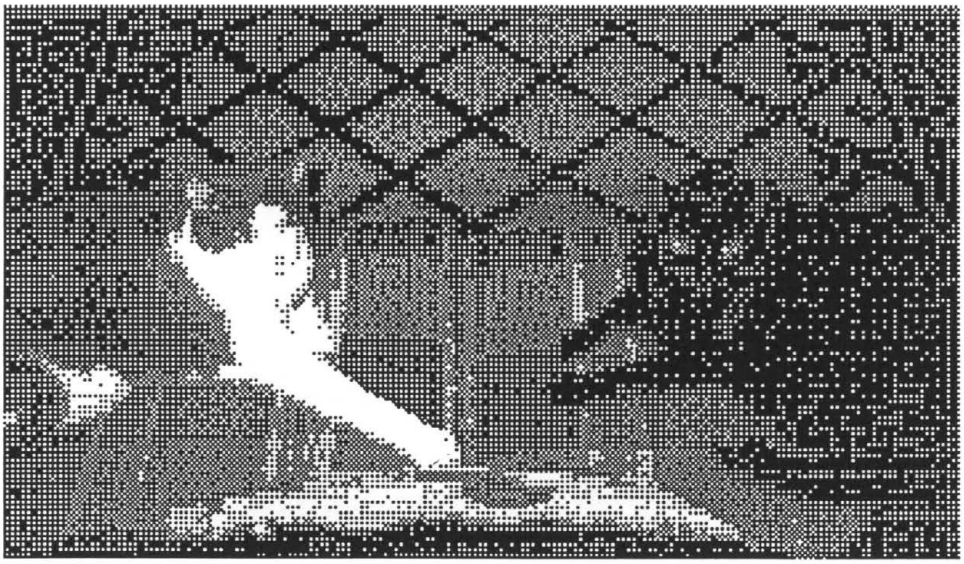

Figure 8.12—8×8 RD (on ordered dither) "Cats" Image at 72 ppi

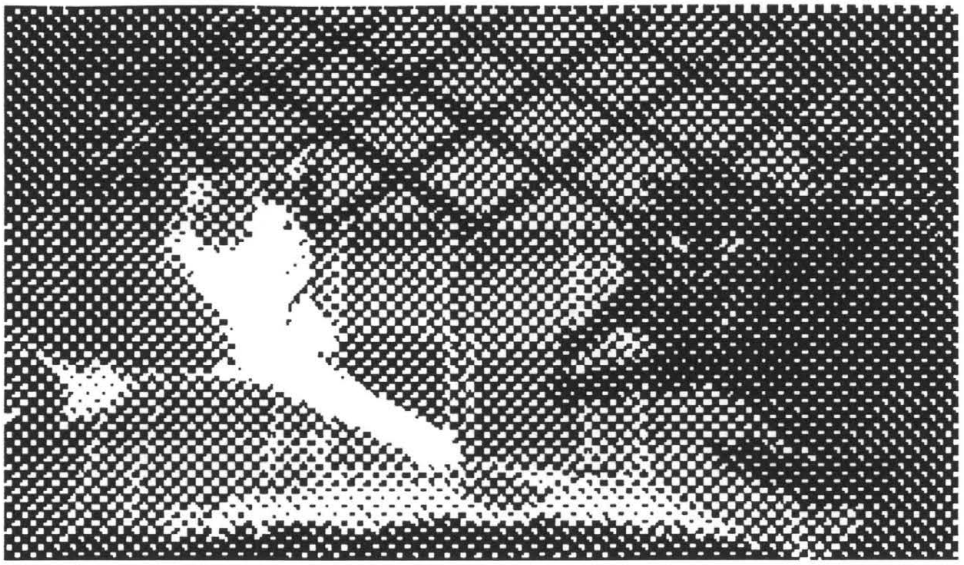

Figure 8.13—Clustered Dither "Cats" Image at 72 ppi

Figure 8.14—Ordered Dither "Cats" Image at 72 ppi

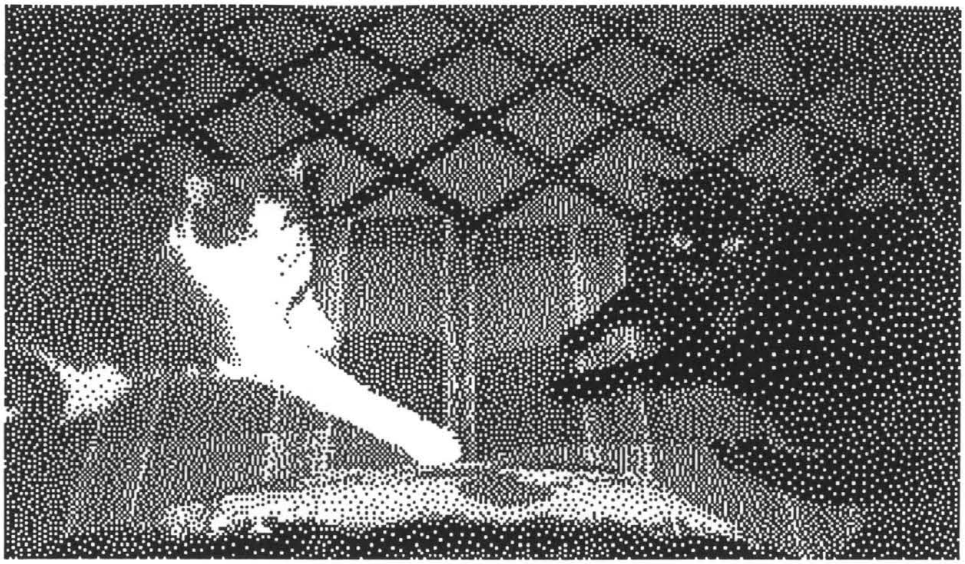

Figure 8.15—Floyd-Steinberg "Cats" Image at 72 ppi

Figure 8.16—White Noise "Cats" Image at 72 ppi

Figure 8.17-Figure 8.22 display the same images at 432 ppi:

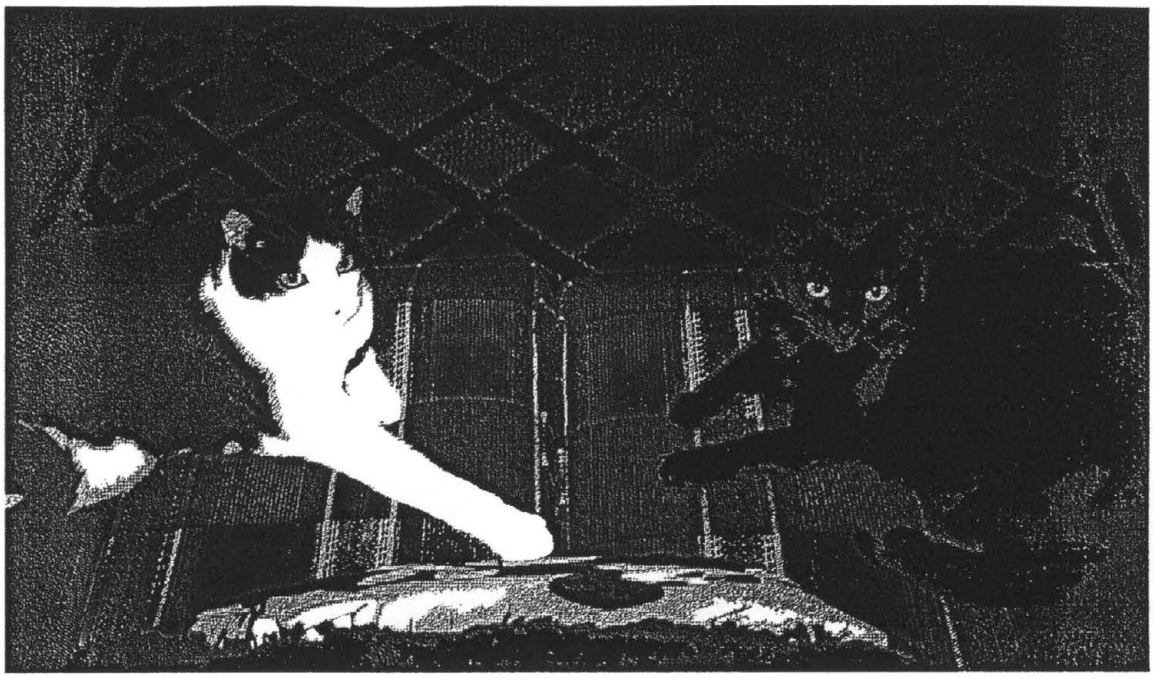

Figure 8.17— BS "Cats" Image at 432 ppi

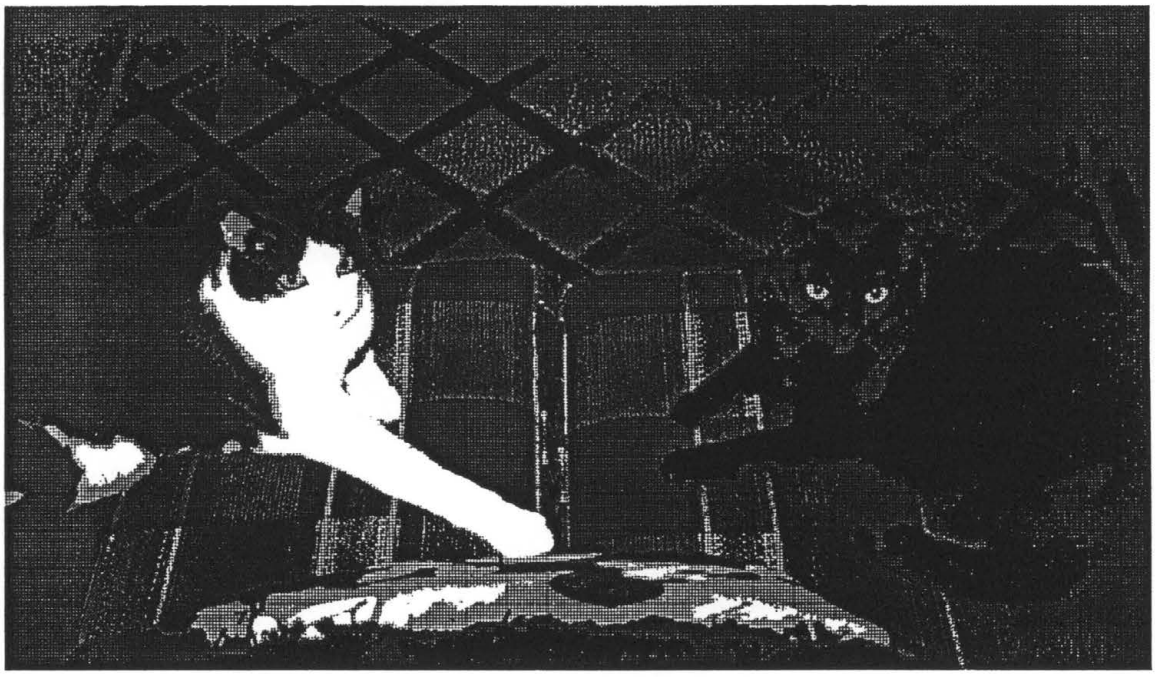

Figure 8.18—RD "Cats" Image at 432 ppi

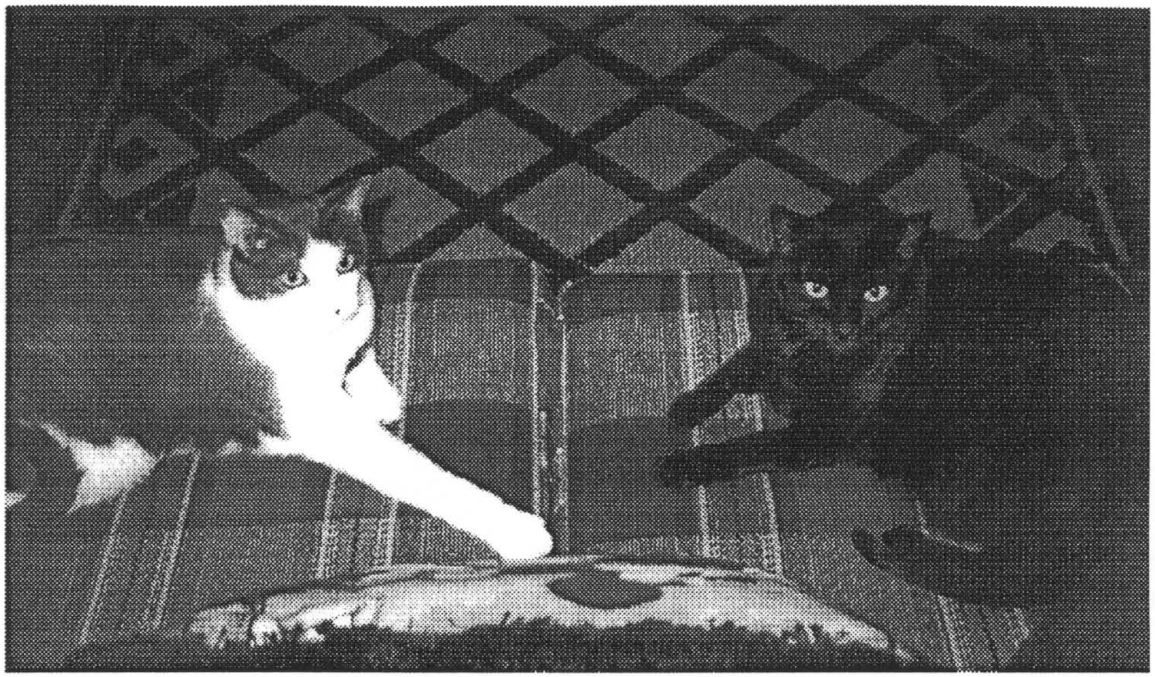

Figure 8.19—Clustered Dither "Cats" Image at 432 ppi

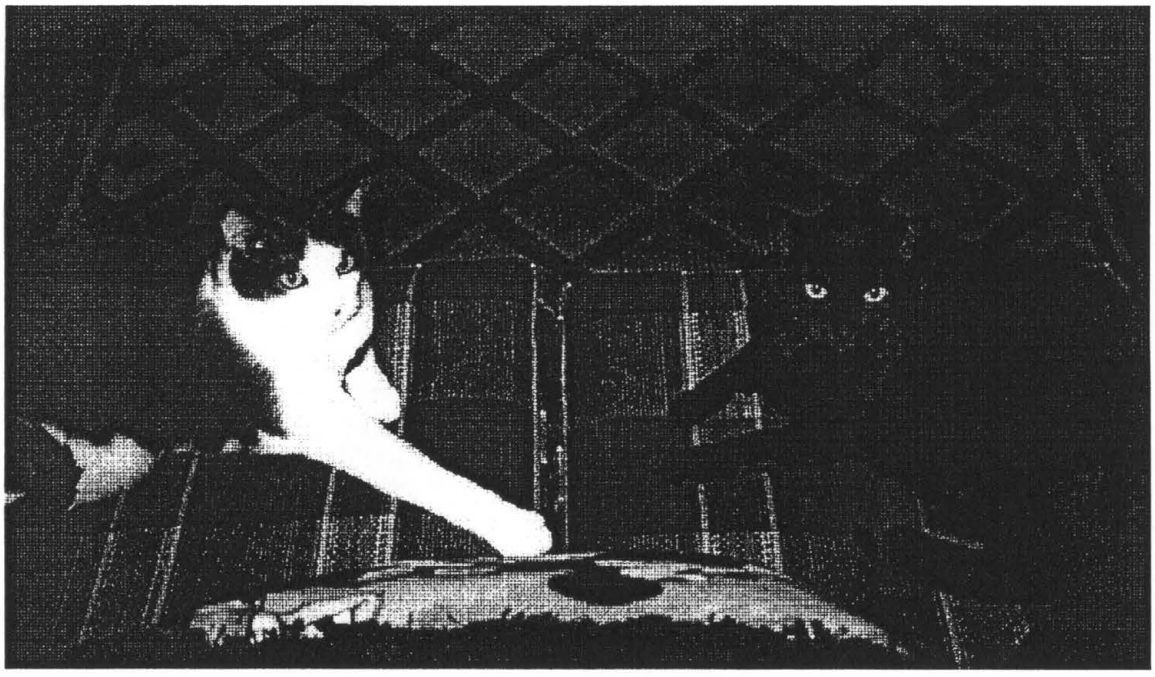

Figure 8.20—Ordered Dither "Cats" Image at 432 ppi

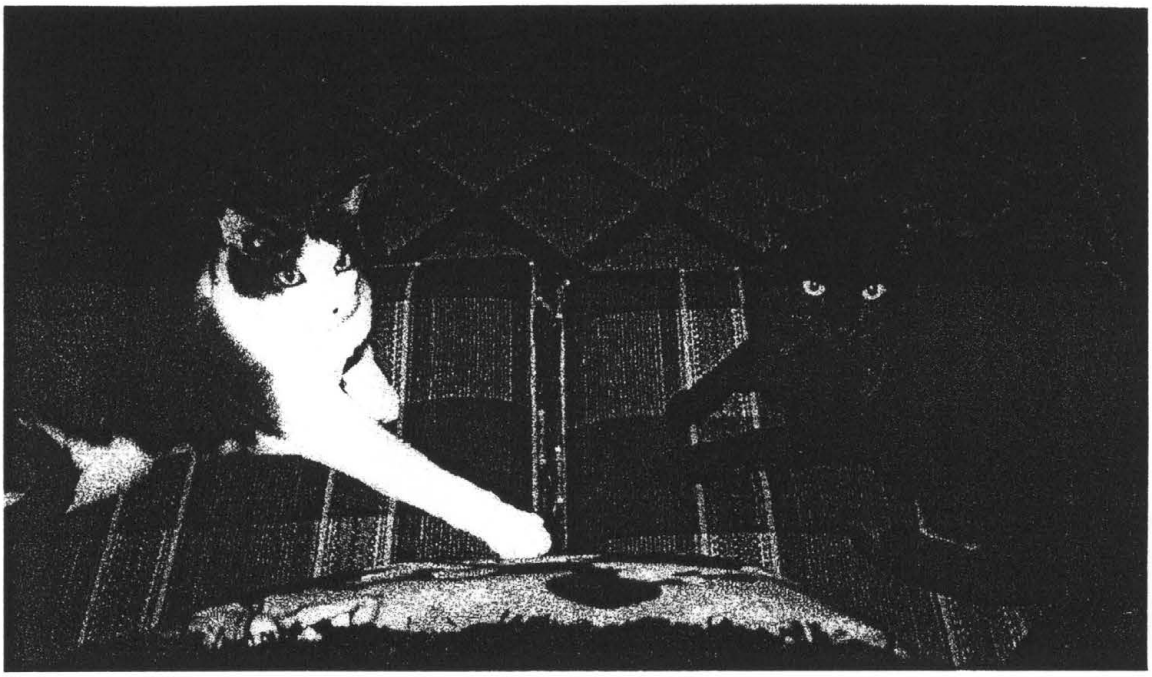

Figure 8.21—Floyd-Steinberg "Cats" Image at 432 ppi

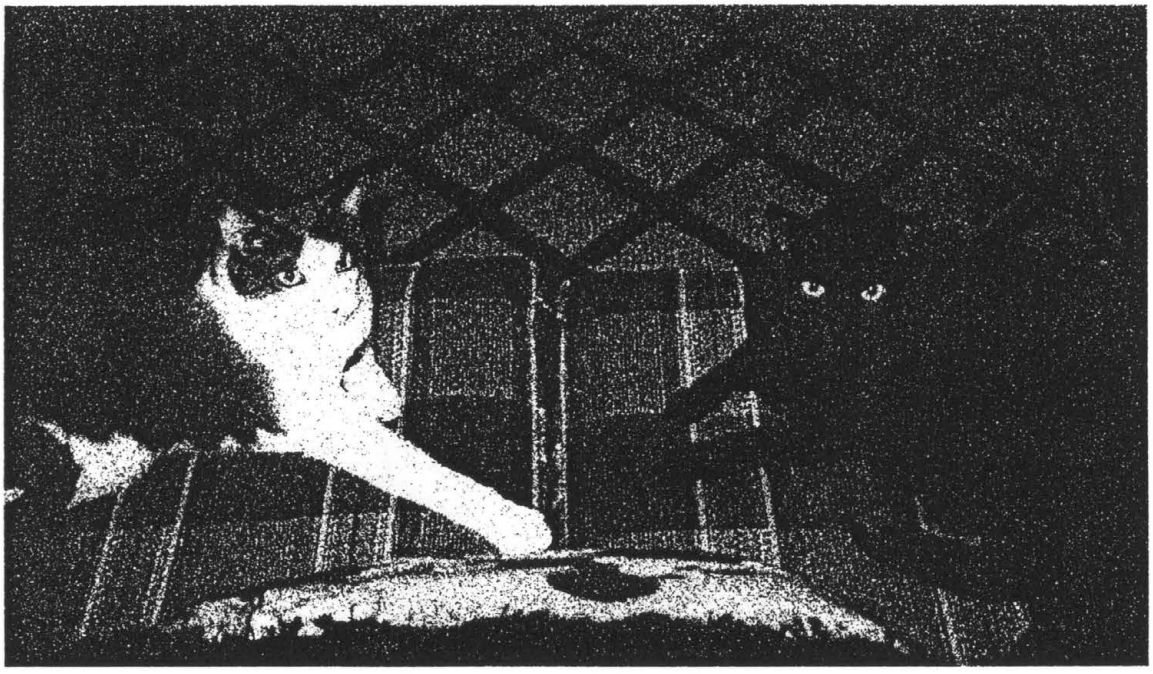

Figure 8.22—White Noise "Cats" Image at 432 ppi

BS Image Enhancements

According to Ulichney (1987), an image enhancement is any process that when applied to an image that produces a new image with better quality (as determined by some criteria). For this test, BS algorithm was applied to 432 ppi versions of the test images. The bi-level pixels of the resulting images were then averaged down to form a 1:1 zoom of the image at 72 ppi (similar to those of the "Threshold Resolution" test). Four image transforms were then applied to each of these zoom images in an attempt to produce image enhancements. These image transforms include selective Gaussian blur, median filtering, low pass Fourier filtering, and histogram equalization. In addition, an image sharpened before halftoning is included in the survey.

These modified images were then shown along with the original image to the LCD and print subjects in a manner similar to the previous tests. However, for this experiment, the subjects had no knowledge as to which image was the original. Figure 8.23—Figure 8.28 shows the six transforms applied to the 1:1 zoom "Sinus" image.

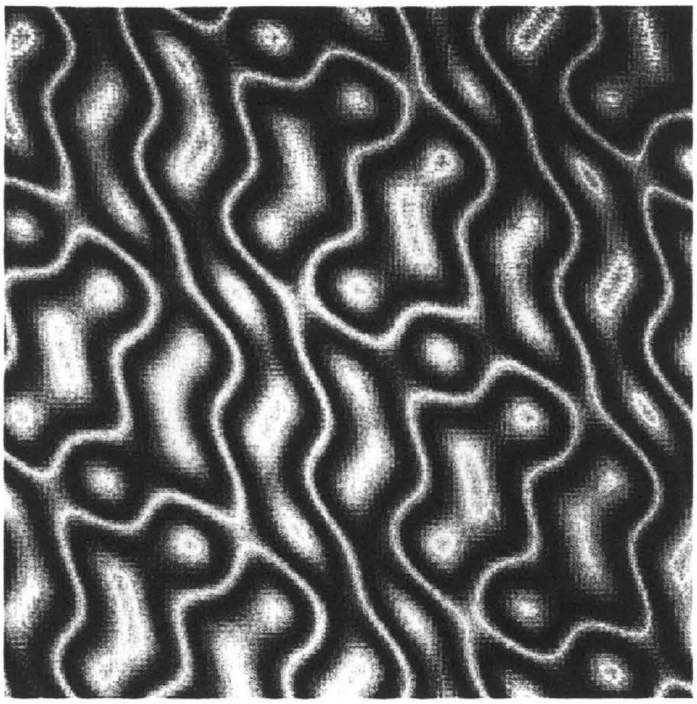

Figure 8.23— 1:1 BS zoom "Sinus" Image (at 72 ppi)

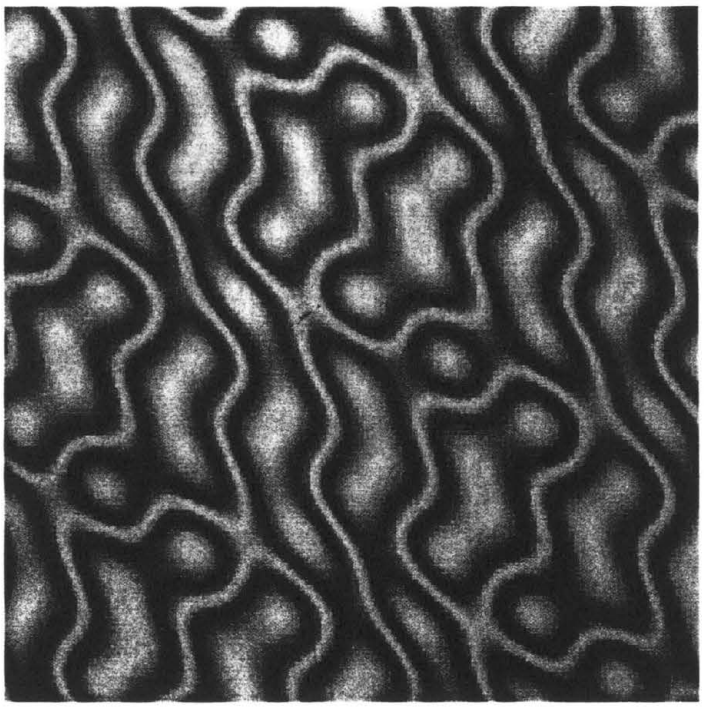

Figure 8.24— Guassian Blur Applied to the "Sinus" BS zoom

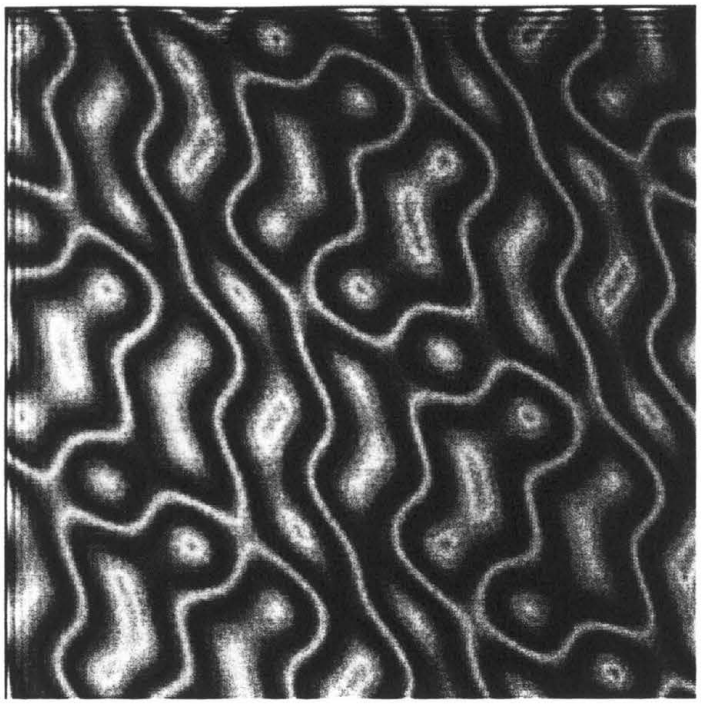

Figure 8.25— Fourier Low Pass Filtering of "Sinus" BS zoom

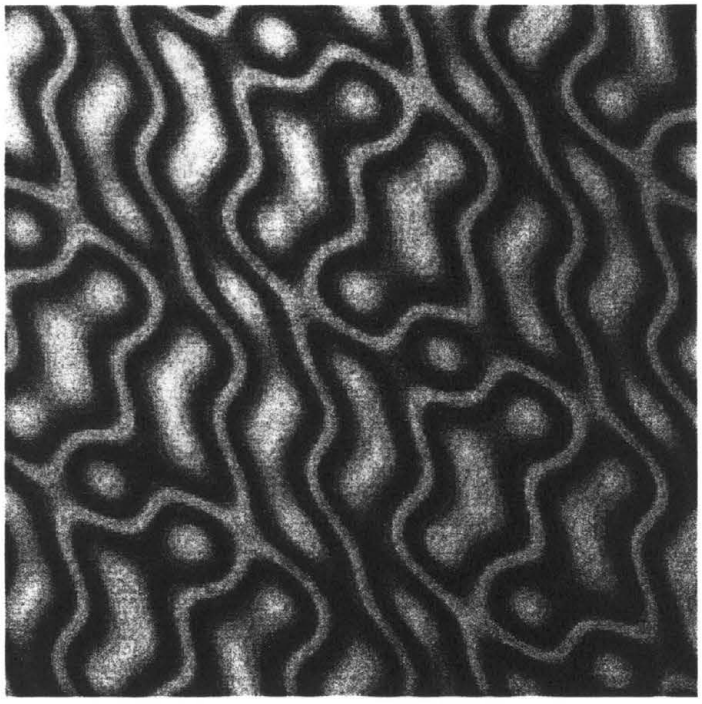

Figure 8.26—Median Filtering Applied to the "Sinus" BS Zoom

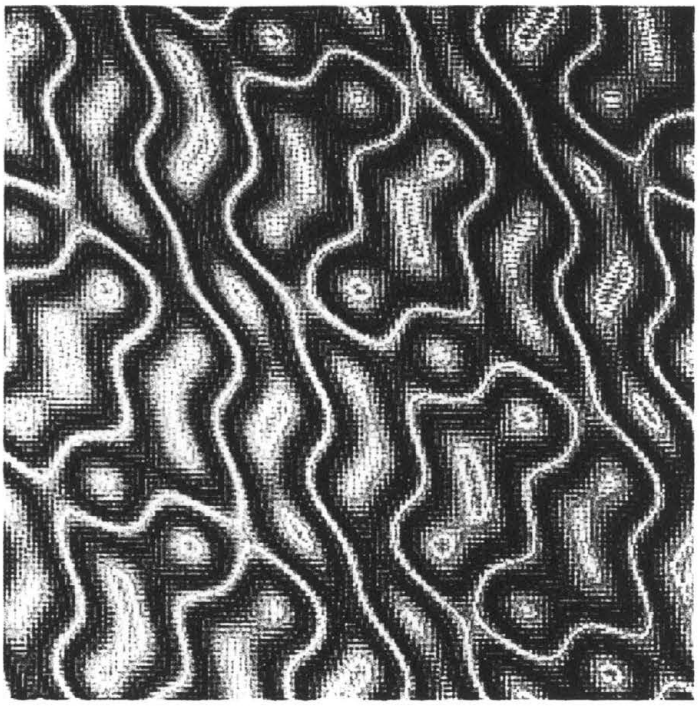

Figure 8.27— Post-Sharpening Applied to the "Sinus" BS Zoom

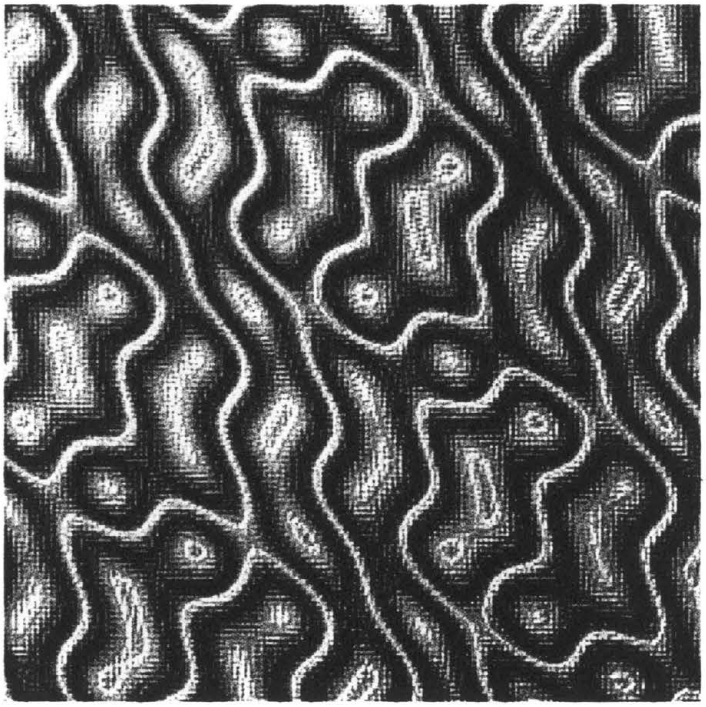

Figure 8.28— Histogram Equalization Applied to the "Sinus" BS Zoom

RD Image Enhancements

This test is the same as the "BS Image Enhancements" test where the RD algorithm is applied in place of BS. Figure 8.29—Figure 8.34

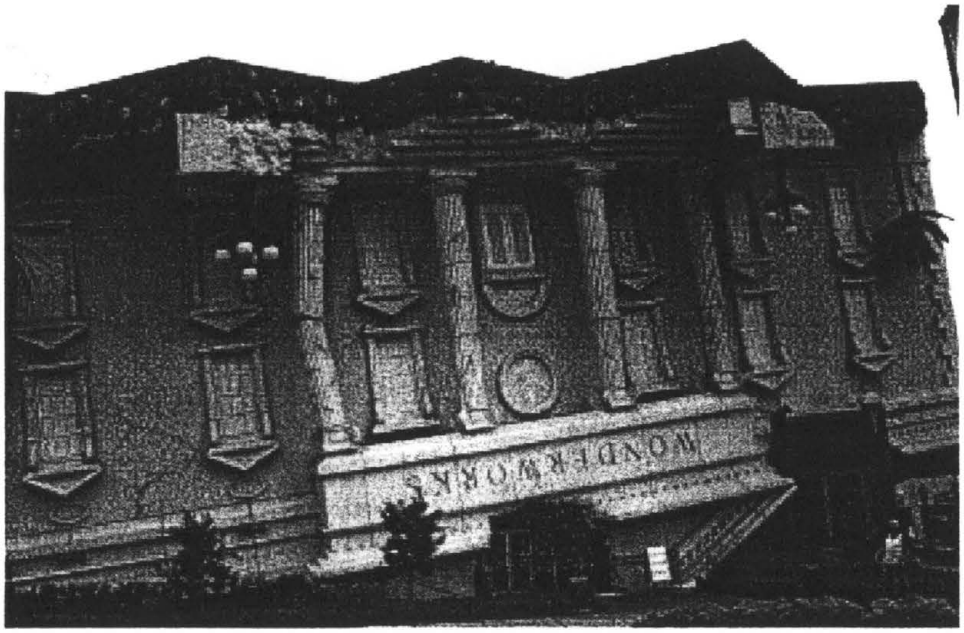

Figure 8.29— Applied to the RD zoom "Building" Image (at 72 ppi)

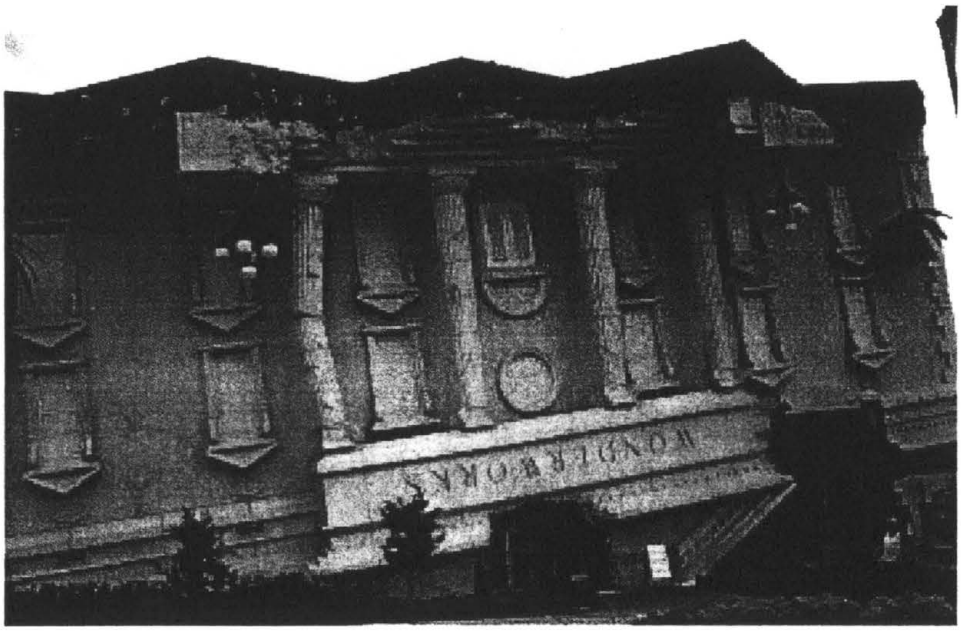

Figure 8.30— Applied to the RD zoom "Building" Image (at 72 ppi)

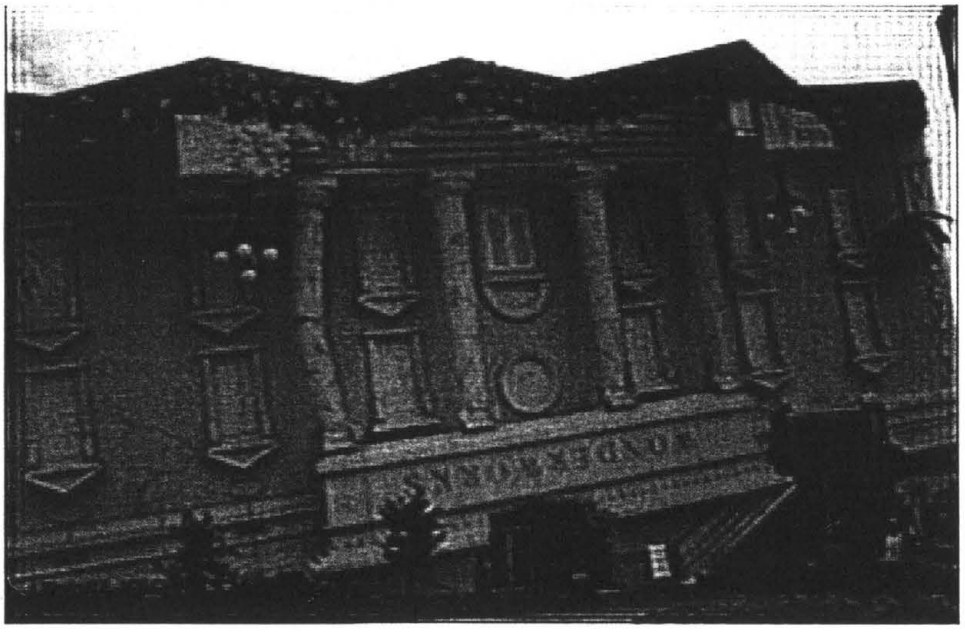

Figure 8.31— Applied to the RD zoom "Building" Image (at 72 ppi)

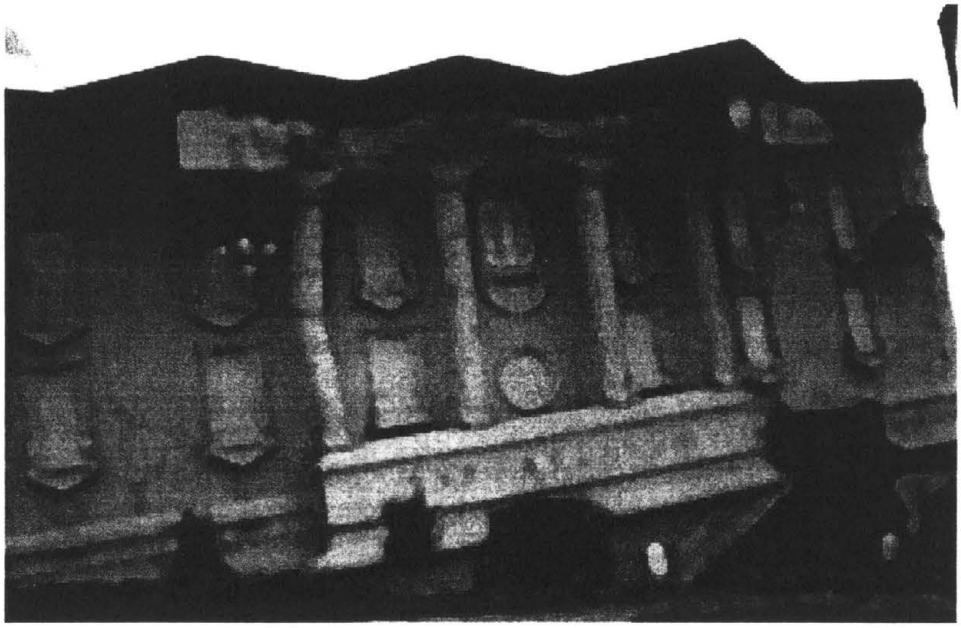

Figure 8.32— Applied to the RD zoom "Building" Image (at 72 ppi)

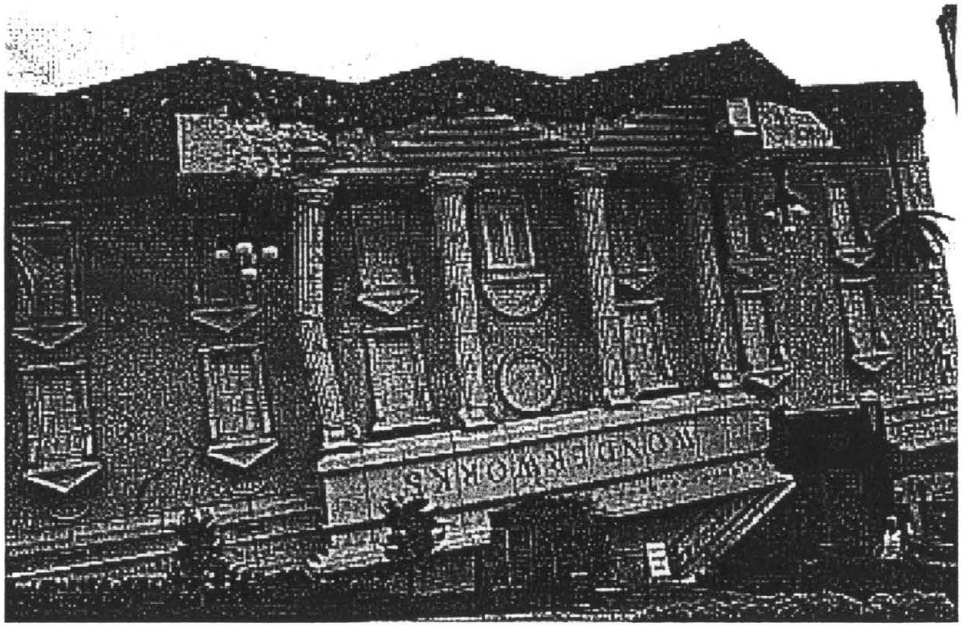

Figure 8.33— Applied to the RD zoom "Building" Image (at 72 ppi)

Figure 8.34— Applied to the RD zoom "Building" Image (at 72 ppi)

Text Survey

This survey gathers data on pure text images. The same 25 subjects of the image survey also participated in this survey. For this test, subjects looked at the text images composed of completely random lines of serif and sans serif fonts shown with and without antialiasing. The following fonts sizes were displayed: 72, 48, 36, 28, 24, 16, 14, 12, 10, and 8. Both the dithered and 1:1 zoom forms of the images were presented. For each of these images, the test subjects were asked to write out the characters of each line of text. The readability of the text is determined by noting the number of correctly identified characters for each line. The data is presented in charts plotting the readability percentage versus the font size.

Difference Perception

This experiment is inspired by the work of Daly (1997), Taylor, et al (1998, 1998), Lubin (1997), and other scientists who have used psychophysical theories to develop objective models for predicting the human ability to perceive differences between an original image and a processed counterpart. These models have been developed in response to the poor performance of statistically based quality/fidelity metrics.

Daly's method presents its predictions in the form of "difference maps" (1997) that highlight where a person with normal vision will perceive differences. This experiment attempts to generate an imitation of these difference maps subjectively. Eight human subjects with normal or corrected vision looked at printed versions of one of the original test images and its corresponding BS and RD 1:1 zooms (from 432 ppi bi-level to 72 ppi deep-level). These subjects were then asked to highlight where they could perceive differences using a gold pen. A 4×4 grid covered both the original image and the zoomed images to facilitate the comparisons. The fidelity of the processed images is assessed by approximating the highlight percentage of each grid cell. It is predicted that this data will be highly correlated to the subjective ratings of the surveys. Thus, these percentages are plotted against the subjective ratings. Figure 8.35 and Figure 8.36 are examples of the highlighted images produced by the test subjects.

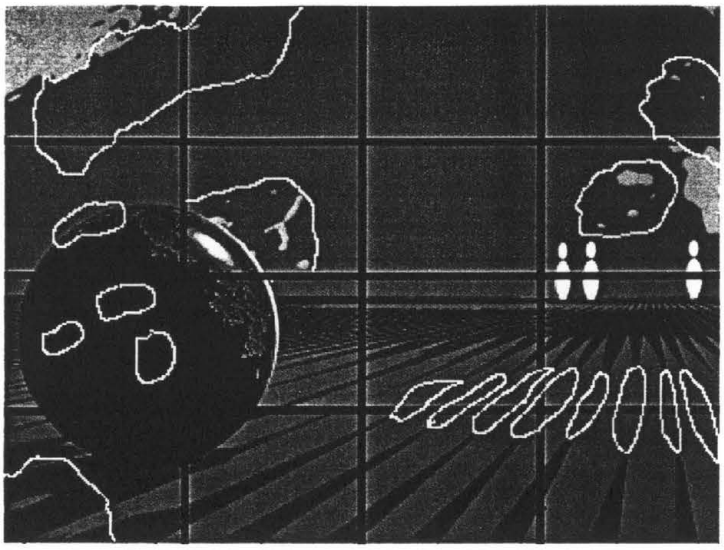

Figure 8.35—Highlighted BS Zoom of "Bowl" Image

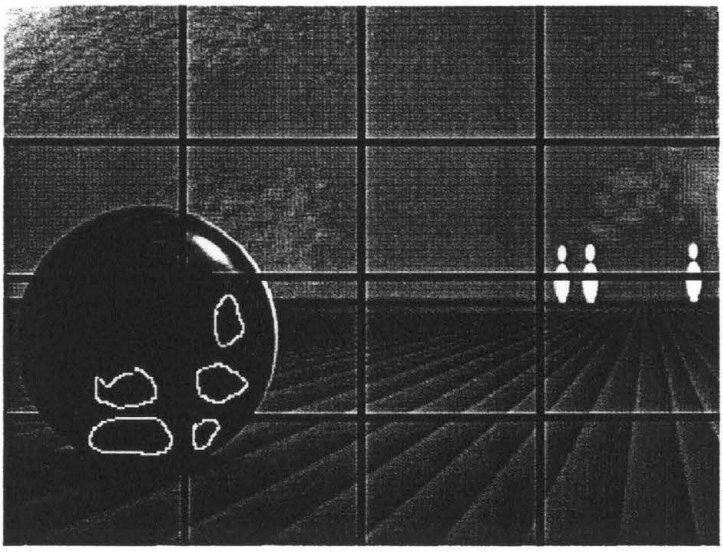

Figure 8.36—Highlighted RD (on Ordered Dither) Zoom of "Bowl" Image

Compression Analysis

Case's NOTA algorithm and the RBE algorithm of this study are analyzed in two ways. First, the general properties of these algorithms are determined. This analysis is followed by an assessment of the compressibility achieved for halftoned vector, natural, and pure text images. The following sections describe the data, tests, and experiments used in this assessment.

Image RLE Comparison

For this experiment, nine different sized versions of the test images were produced. The BS and RD algorithms were then applied to each of the resulting images producing 18 bi-level dithered images.

Data for this test was generated by applying the implementations of NOTA and RBE to the aforementioned input images. For the sake of comparison, two other RLE compression methods were also applied: Packbits and SimpleRLE (Nelson, 1998). This data is presented in Chapter 9 with charts plotting empirical compression ratios versus the size of the input data (in units of bytes).

Secondary Image Compressions

The NOTA and RBE files produced by the previous experiment are used as the inputs of this experiment. To each of these files, a second compression technique is applied. These secondary compressions include: Huffman coding (8-bit alphabet for

NOTA, 3-bit alphabet for RBE), dictionary encoder (gzip) (Nelson, 1998), and an arithmetic encoder (Nelson, 1998). Again, this data is presented in Chapter 9 with charts plotting empirical compression ratios versus the size of the input data (in units of bytes).

Image Format Comparison

In this test, the synergy of RD with NOTA and RBE is compared to other popular image formats (BMP, GIF, JPG, PNM, PNG). In order to compare against images of similar quality, the RD images are stored at 432 ppi and the other formats are stored at 72 ppi. The RD images would subsequently be averaged down to 72 ppi to produce images similar to the other formats. The data is presented in Chapter 9 with charts plotting file sizes versus the number of image pixels. This test along with the subjective quality data is meant to show the viability of such combinations for use as actual image formats.

Text Compression

The three above tests are also carried out on six different sized versions of a text image. The format comparison is slightly different, however. The RD-NOTA and RD-RBE text images are compared to the postscript and PDF formats, rather than the typical image formats.

CHAPTER 9

DATA ANALYSIS

This chapter presents the data gathered by the methods of Chapter 8. Since the smallest details of an experiment or test often offer the most information, this chapter describes, lists, and analyzes the data on a section-by-section basis. The collective interpretation of the entire data set is deferred until Chapter 10 ("Conclusions").

Halftone Analysis

As stated in Chapter 8, this study analyzes Case's BS and RD algorithms in two ways. First, the general properties of these algorithms are determined along with the properties of the images that they produce. This analysis is followed by a subjective and objective assessment of quality for images produced by the dithers.

Properties

The following six facets of Case's halftoning algorithms are subsequently explored:

- Dither Patterns
- Moirés

- Boundary Pixels
- Time and Space Complexity

Dither Patterns

As previously mentioned, the BS algorithm produces a distinct set of dither patterns. A given input cell size, δ, will produce $2^{\frac{\delta^2}{2}+1} - 1$ different patterns. Equation 9.1 shows that the number of graylevel intensities simulated by BS depends directly on the size of δ.

$$number\ of\ \ graylevels = \delta^2 + 1$$

Equation 9.1—Relation Between Simulated Graylevels and Dither Pattern Size

Figure 9.1 shows the patterns and intensities produced by BS with $\delta = 2$.

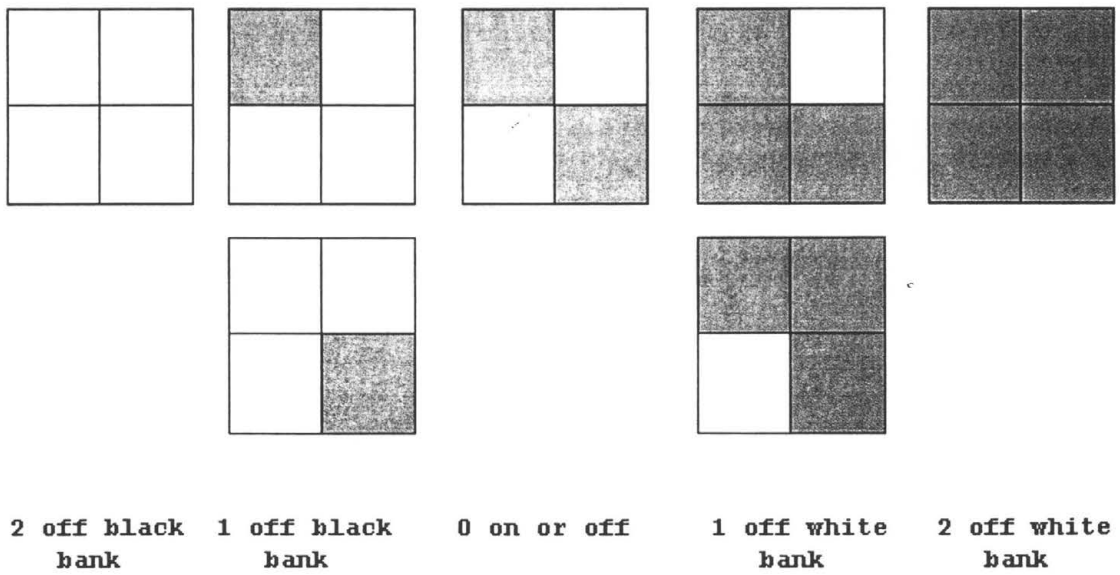

Figure 9.1—2×2 BS Patterns and Intensities

For practical applications, δ has an upper bound. This bound is imposed by the amount of resolution lost for each size of δ. For normal images, the pixels composing the image are the primitive display units. However, for dithered images, these primitives correspond to the dither cells used to halftone the deep pixel input. Figure 9.2 illustrates this loss of resolution using four BS images where δ equal 2, 4, 8, and 16. Note that as δ increases, so does the amount of lost detail in the image.

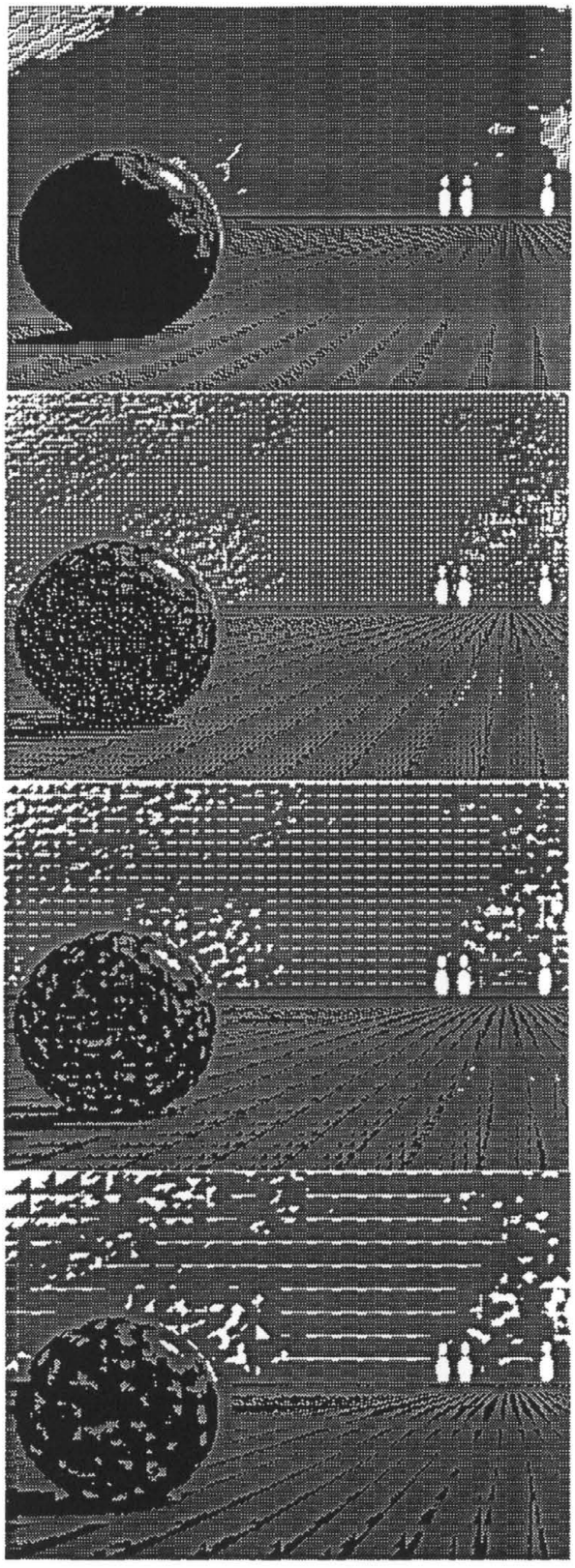

Figure 9.2—Resolution Loss and δ

RD is a method for halftoning deep pixel images, but, strictly speaking, it is not a dithering method. As described in Chapter 6, RD seeks to improve the performance of existing dithering methods by considering neighborhoods of $\delta \times \delta$ cells. As a result, the patterns produced by RD are dictated by the dither method on which it is applied.

For most of this study, RD operates on an ordered dispersed-dot dither. Patterns for these dithers can be derived from dither matrices. Equation 6.6 provides a means for recursively specifying a dither matrix for a given δ. Figure 9.3 illustrates the five patterns and grayscale intensities produced by the ordered dither used in this study.

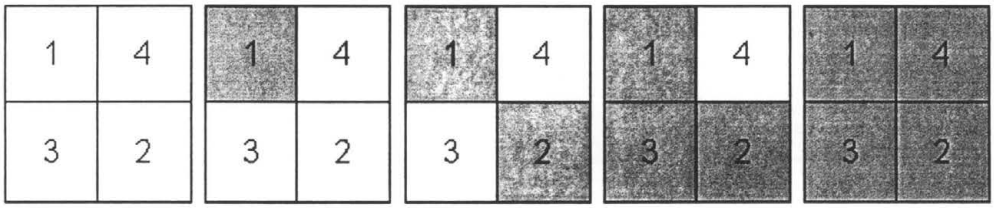

Figure 9.3—Five Patterns of 2×2 Ordered Dither

In general, a given δ produces $\delta^2 + 1$ distinct patterns and $\delta^2 + 1$ simulated graylevels. Similar to the BS algorithm, δ is bound from above for practical purposes due to lost resolution.

Observations made during the course of this study support the notion that δ's greater than four produce unacceptable bi-level images for both the BS and ordered dithers.

Moirés

The moirés produced by Case's halftoning methods are assessed by observing the moirés of gradient images, the test images of this study, and the animation of simple geometric shapes. Figure 9.4-Figure 9.6 show grayscale gradients halftoned with the 2×2 invocation of the BS algorithm, 8×8 RD applied over 2×2 ordered dither, and 8×8 RD applied over 2×2 BS.

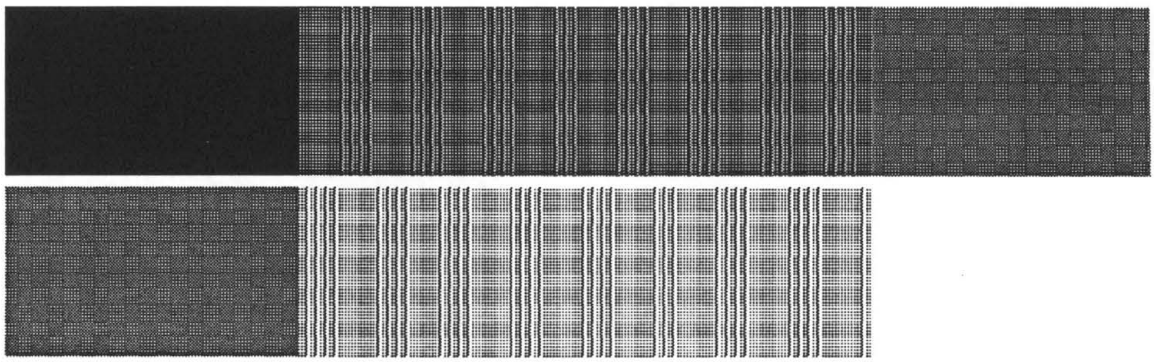

Figure 9.4—Moirés in BS Grayscale Gradients (at 144 ppi)

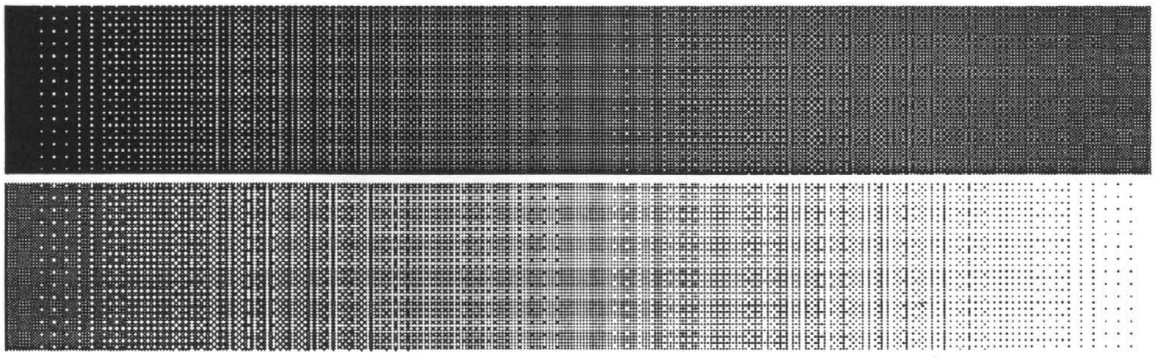

Figure 9.5—Moirés in Grayscale Gradients RD on Ordered Dither (at 144 ppi)

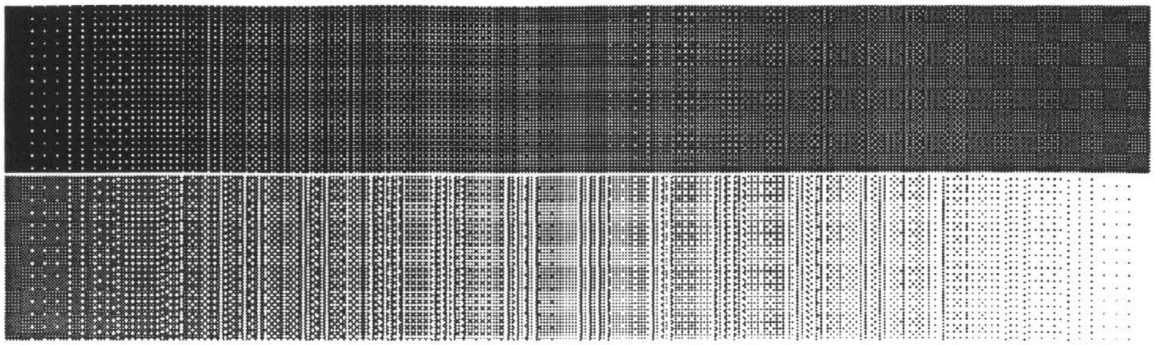

Figure 9.6—Moirés in Grayscale Gradients RD on BS (at 144 ppi)

These figures show two things. Firstly, notice how the 2×2 BS algorithm only reproduces five intensities in the grayscale image. The 8×8 RD algorithms reproduces 64 intensities. An important property of the RD images is that these 64 colors can be reproduced without losing an excessive amount of resolution. Refer to the third image of Figure 9.2 for a demonstration of how an application of 8×8 BS sacrifices detail in the image. Secondly, these images show the periodic vertical and horizontal artifacts present because of these schemes.

Figure 9.7-Figure 9.9 are 72 ppi versions of the "Bowl" image halftoned with the same methods as the gradients.

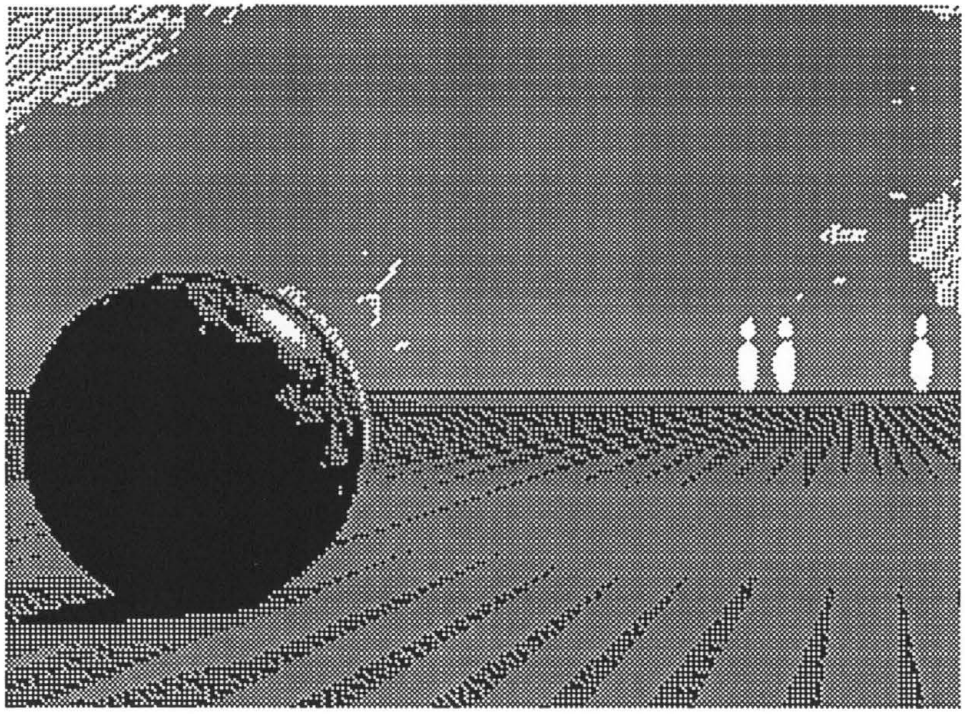

Figure 9.7—2×2 BS "Bowl" Image for Moiré Observation (at 72 ppi)

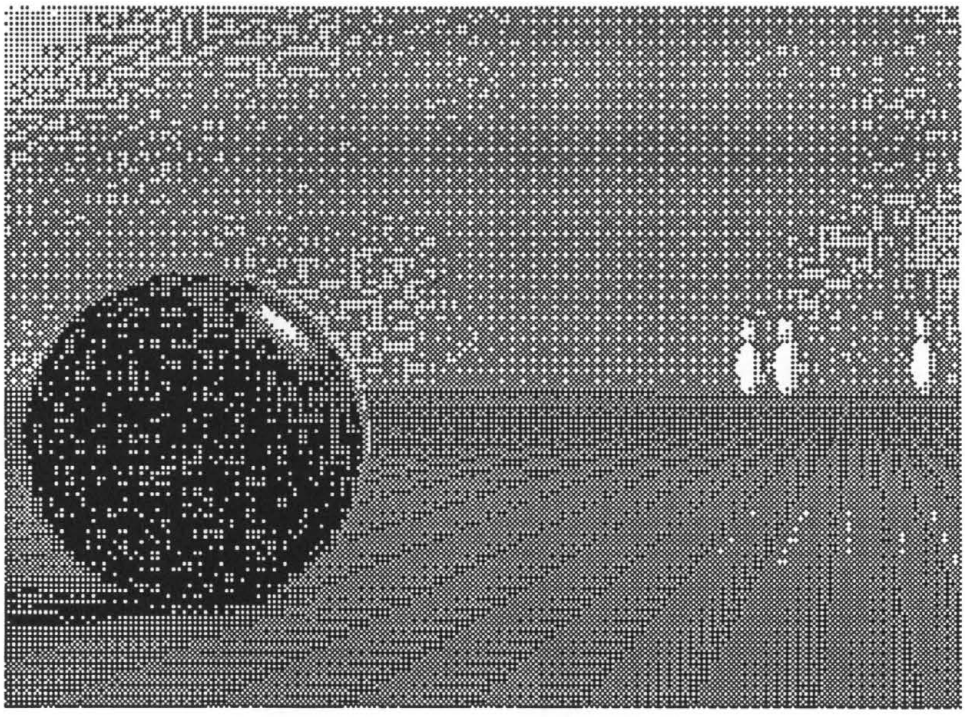

Figure 9.8—8×8 RD on Ordered Dither "Bowl" Image for Moiré Observation (at 72 ppi)

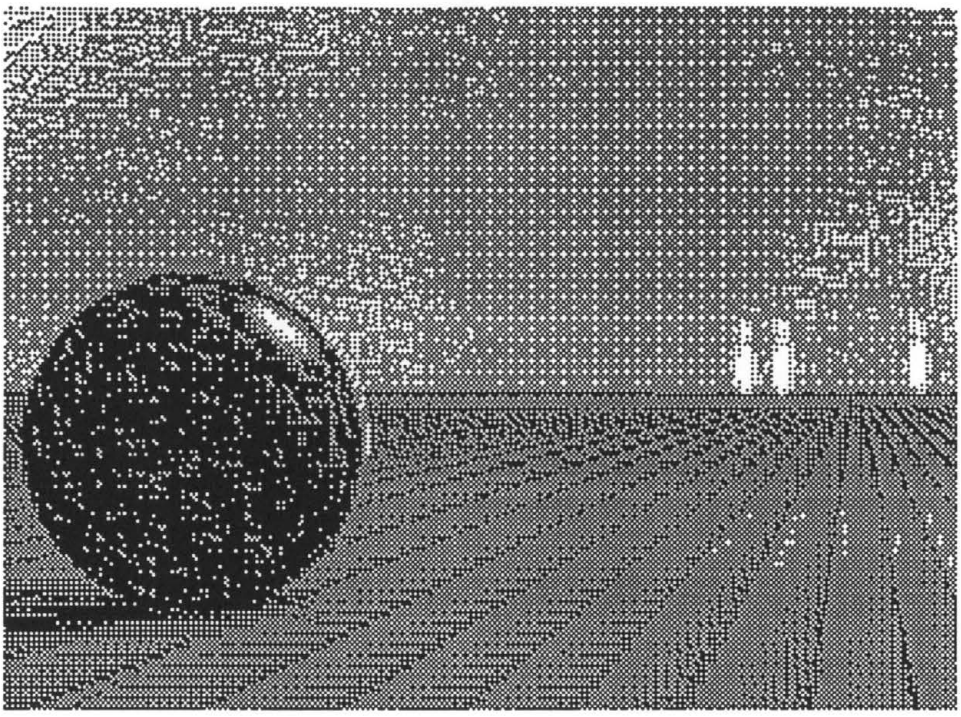

Figure 9.9—8×8 RD on BS "Bowl" Image for Moiré Observation
(at 72 ppi)

These images illustrate the moirés that these methods produce on the "Bowl" image. Moirés in the BS image are not very apparent; however, as previously stated, this image does not reproduce many intensities. The RD images, on the other hand, reproduce several graylevels, but also contain several vertical and horizontal moirés (at 72 ppi). These moirés are most apparent in the sky portion of the image. Closer inspection of the RD images reveals that the moirés in the ordered dither image are more frequent and larger than those of the BS image. This fact can be observed by looking at the top-left and bottom-right corners of the images. The RD on BS tends to breakup and reorient the vertical and horizontal artifacts present in the RD on ordered dither image.

Animation of dithered images also can produce undesirable moiré artifacts. This section uses animated GIFs to observe the moiré effects produced when rotating the

dithered images of simple geometric figures. Figure 9.10 shows five triangles with different grayscale intensities. Figure 9.11 and Figure 9.12 show non-antialiased and antialiased version of lines joined at a common center in a circular fashion. Each of these images was rotated through 360 degrees in one degree increments. Each of these increments became a one-tenth second frame in the subsequent animation.

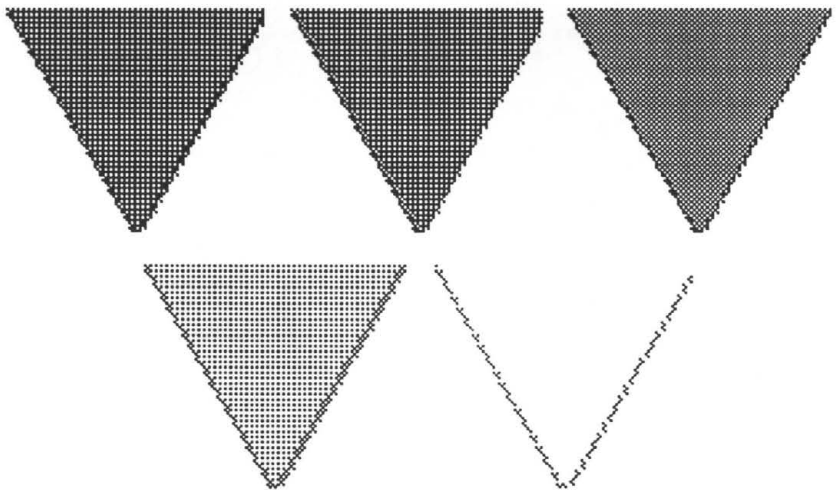

Figure 9.10—Triangle at 12.5% gray, 25% gray, 50% gray, 62.5% gray, 75% gray (Dithered at 72 ppi)

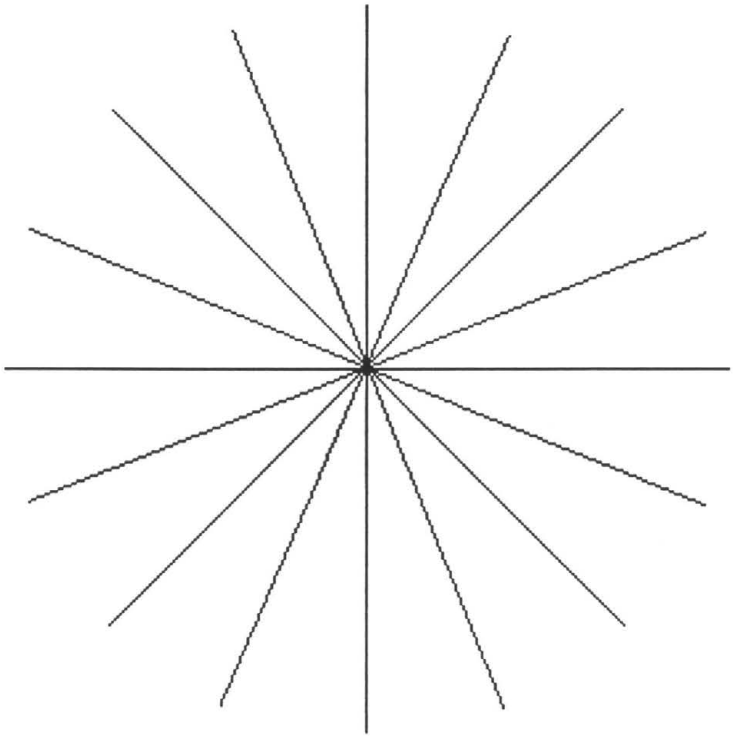

Figure 9.11—Non-antialiased Lines (Dithered at 72 ppi)

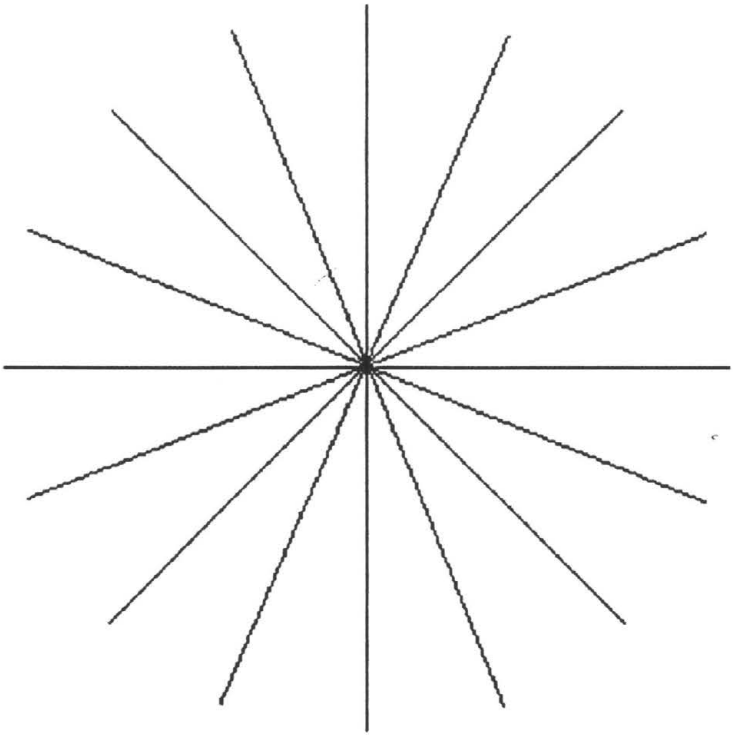

Figure 9.12—Antialiased Lines (Dithered at 72 ppi)

Rotation of each triangle in Figure 9.10 produced moirés that resembled the spinning of pinwheels. These artifacts became most apparent as the triangles rotated through angles that are multiples of 45°. The rotations of Figure 9.11 and Figure 9.12 produced artifacts in which waves seemed to radiate from the center of the lines. These lines are less visually disturbing than the moirés of the triangles only because of the smaller area occupied by the individual lines. Again, the artifacts became most apparent as the images rotated through angles that are multiples of 45°.

Boundary Pixels

The current realizations of the BS and RD algorithms convert images on a cell-by-cell basis while scanning cells in normal English reading order (left to right, top to bottom). They implicitly assume that an image contains an integer number of $N \times N$ or $\delta \times \delta$ cells. In general, this is not the case. As a result, problems arise when the width and/or height of the image is not evenly divisible by the cell size of the method.

Truncation of along the edges of the image could serve as a simple solution for the problematic pixels. However, a significant amount of information would be lost. A simple test was run with five test subjects to determine the noticeability of these truncated pixels. The subjects were shown an original image along with six additional images with 2, 4, 8, 16, 32, and 64 pixels truncated. The subjects were then asked to pick the images with noticeable truncations. Four of the five subjects chose the truncation of 16 pixels as the noticeability threshold.

Since truncation leads to degradation of the original image, a better solution would be to halftone the "partial" cells along the edges of the image with smaller and smaller cells until an entire row has been halftoned. If the width and height of the cell are even then this method can convert all of the cell using cell sizes of $\frac{(cell\ size)}{2^m}$ where m is some integer less than $\log_2(cell\ size)$. When the width or height is odd, then this method can convert all of the pixels except those of the final row or column. These remaining pixels can be halftoned using pixel thresholding. Since a smaller scale of the same scheme is used to convert these boundary pixels, it is predicted that this method will produce images without artifacts specific to the edges of the bi-level image.

Time and Space Complexity

According to Rosen (1995) and Cormen, et al (2000), if $f_1(x)$ and $f_2(x)$ are $O(g_1(x))$ and $O(g_2(x))$, respectively, then $(f_1(x)+f_2)(x)$ is $O(max(g_1(x), g_2(x))$. Thus, the time and space complexity of the BS and RD algorithms is analyzed by first decomposing the processes into a set of primitive linearly ordered subprocedures. The average case complexities of the algorithms are then taken to be the same as those of the most complex (also average case) subprocedure.

Table 9.1and Table 9.2 summarize the time complexity analysis for the BS and RD algorithms.

	Subprocedure	*Average Time Complexity*	*Justification*
a.	Read a $\delta \times \delta$ cell of deep pixel graylevels	$O(\delta^2)$	One assignment statement is required for each element of the $\delta \times \delta$ cell.
b.	Calculate the sum of the $\delta \times \delta$ graylevels.	$O(\delta^2)$	Each element of the $\delta \times \delta$ cell must be added to a running total before the sum is calculated.
c.	Determine the number of bi-level pixels, t, to turn "on" or "off" from the checkerboard.	$O(1)$	This subprocedure is realized by a few scalar integer operations.
d.	Sort the proper bi-level bank of pixels on their deep pixel graylevels and change i of them.	$O(\delta \lg(\delta))$	Several existing sorting routines run with this average time complexity. The implementation in the appendix uses a heap sort.

Table 9 1—BS Time Complexity Analysis

	Subprocedure	*Average Time Complexity*	*Justification*
a.	Read a $N \times N$ cell of deep pixel graylevels	$O(N^2)$	One assignment statement is required for each element of the $N \times N$ cell.
b.	Calculate all of the subcell graylevels sums.	$O(N^2 lg(N))$	Each element of the $N \times N$ cell is an input to $lg(N)$ addition operations for calculating each sum.
c.	Calculate the number of "on" pixels for the $N \times N$ cell, $W_{3,0}$.	$O(1)$	This subprocedure is realized by a few scalar integer operations.
d.	Calculate all of the subcell whole and partial pixels, $W_{r,\omega}$ and $P_{r,\omega}$, respectively.	$O(N^2 lg(N))$	At most, N^2 operations are performed through $lg(N)$ levels of recursion.
e.	Adjust for shortfalls, σ.	$O(N^2 lg(N))$	At most, N^2 operations are performed through $lg(N)$ levels of recursion.
f.	Turn on $W_{1,\omega}+\sigma_{1,\omega}$ pixels for each adjusted subcell.	$O(N^2)$	For the entire cell, N^2 pixels are turned "on" at most.

Table 9.2—RD Time Complexity Analysis

In Table 9.1, steps *a* and *b* require the most time. Thus, since an arbitrary image contains a constant number of $\delta \times \delta$ cells, the BS algorithm can halftone the image in $O(\delta^2)$ time. Similarly, steps *b*, *d*, and *e* require the most time in Table 9.2. Thus, an arbitrary image can be halftoned with RD in $O(N^2 lg(N))$ time.

Table 9.3 and Table 9.4 summarize the space complexity analysis for the BS and RD algorithms.

	Subprocedure	*Average Space Complexity (in bytes)*	*Justification*
a.	Read a $\delta \times \delta$ cell of deep pixel graylevels	$O(\delta^2)$	At least one byte is required to store each element of the $\delta \times \delta$ cell.
b.	Calculate the sum of the $\delta \times \delta$ graylevels.	$O(1)$	A single integer variable can be used to both store the running total and the final summation of the $\delta \times \delta$ graylevels.
c.	Determine the number of bi-level pixels, i, to turn "on" or "off" from the checkerboard.	$O(1)$	This subprocedure is realized by a few scalar integer operations each requiring a constant number of integer variables.
d.	Sort the proper bi-level bank of pixels on their deep pixel graylevels and change i of them.	$O(\delta^2)$	While this depends on the sorting mechanism used, it takes at least the δ^2 bytes needed to store the bi-level pixels. The implementation in the appendix uses a heap sort. As such, it adds no additional space complexity. Using quicksort (as the first implementation did) requires much more space due to the recursive function calls that it places on the program stack.

Table 9.3—BS Space Complexity Analysis

	Subprocedure	*Average Time Complexity*	*Justification*
a.	Read a $N \times N$ cell of deep pixel graylevels	$O(N^2)$	At least one byte of storage is required for each element of the $N \times N$ cell.
b.	Calculate all of the subcell graylevels sums.	$O(N^2 lg(N))$	At most, $\frac{N^2}{4}$ storage elements are required at each of the $lg(N)$ recursion levels.
c.	Calculate the number of "on" pixels for the $N \times N$ cell, $W_{3,0}$.	$O(1)$	A constant number of storage elements are need to calculate and store this value.
d.	Calculate all of the subcell whole and partial pixels, $W_{r,\omega}$ and $P_{r,\omega}$, respectively.	$O(N^2 lg(N))$	At most, $2*\frac{N^2}{4}$ storage elements are required at each of the $lg(N)$ recursion levels.
e.	Adjust for shortfalls, σ.	$O(1)$	The storage elements of d are updated by σ. No extra storage is need for this computation.
f.	Turn on $W_{1,\omega}+\sigma_{1,\omega}$ bi-level pixels for each adjusted subcell.	$O(N^2)$	For the entire cell, N^2 pixels are turned "on" at most.

Table 9.4—RD Space Complexity Analysis

In Table 9.3, steps *a* and *d* require the most space. Again, since image generally contains a constant number of $\delta \times \delta$ cells, the BS algorithm can halftone the image with $O(\delta^2)$ bytes of memory. Similarly, for Table 9.4, steps *b* and *d* require the most space. Thus, RD can halftone an arbitrary image with $O(N^2 lg(N))$ bytes.

Image Quality Analysis

For this section, the results of the image survey data are presented first. The correlation between this data and the objective metric data (where available) is subsequently presented to ascertain the validity of these objective metrics.

Threshold Resolution

Chart 9.1-Chart 9.4 summarize the results for this portion of the image survey.

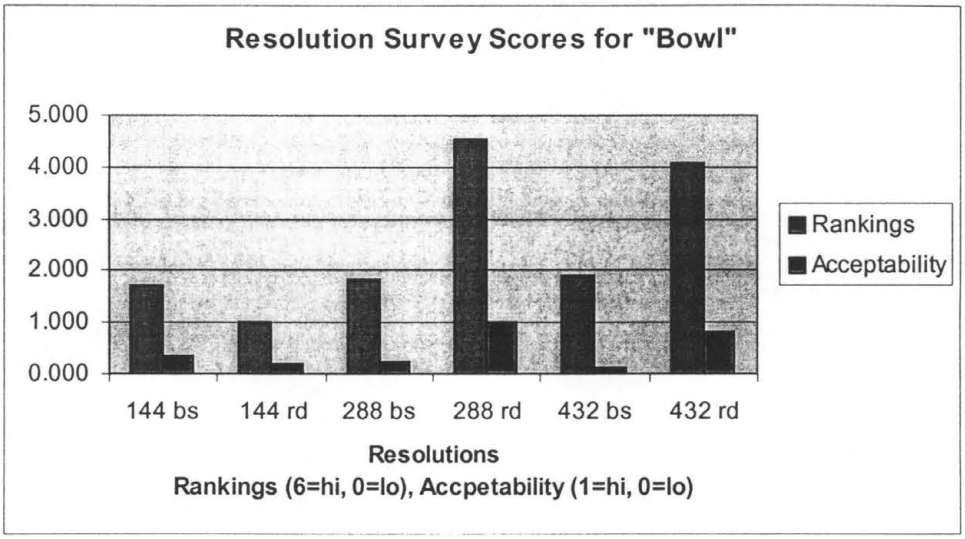

Chart 9.1—Resolution Survey Results for "Bowl"

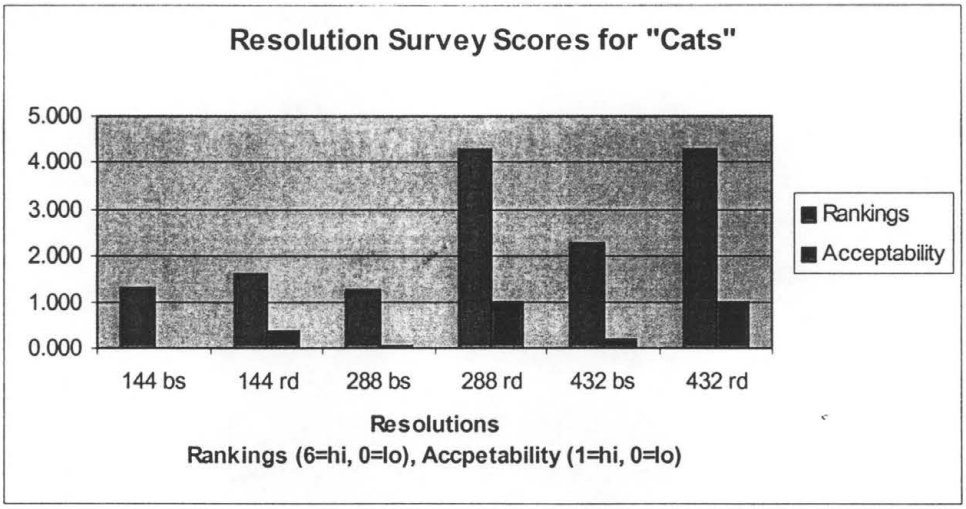

Chart 9.2—Resolution Survey Results for "Cats"

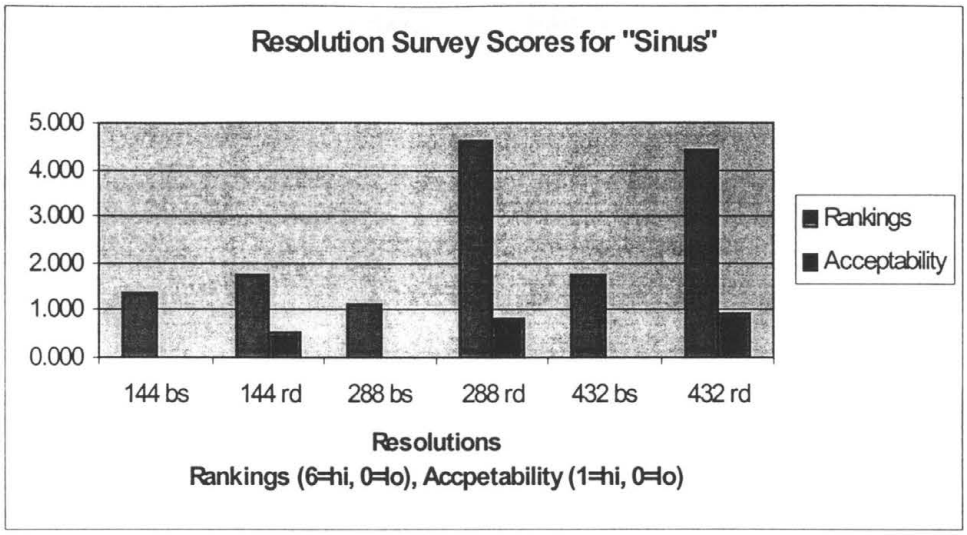

Chart 9.3—Resolution Survey Results for "Sinus"

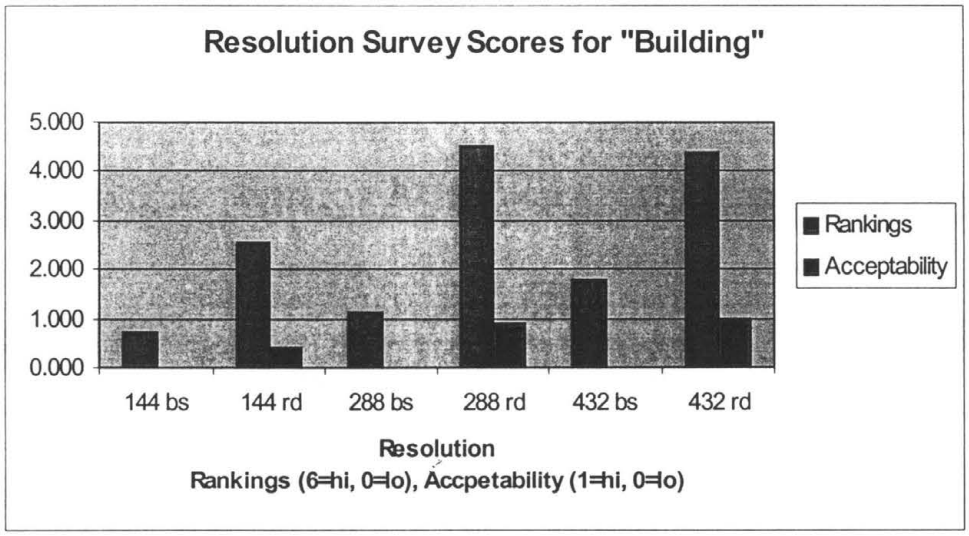

Chart 9.4—Resolution Survey Results for "Building"

This data shows that test subjects overwhelmingly chose the images produced by RD on ordered dither. The acceptability scores of these images show a resolution threshold for of 288 ppi. Interestingly, the RD 288 ppi zooms rank slightly ahead of their 432 ppi counterparts. In other words, the subjects chose a 4-bit grayscale image (the 288 ppi zoom) over an image with more intensities (36 for the 432 ppi zoom).

As previously mentioned, the ordered dither RD zooms of vector images contain a periodic box-like moiré that are noticeable when *scale factors* are not equal to some power of two[3]. These moirés are very apparent in the 432 ppi "Bowl" zooms. In addition, these moiré are not noticeable in the digitized natural images (see Figure 8.10). These results suggest that the moiré may be caused by the scaling methods used by NetPBM and GIMP. However, more importantly, it suggests that this particular moiré is more significant to test subjects than the number of image graylevels when gauging the quality of these images.

Halftone Comparison

Chart 9.5—Chart 9.8 and Table 9.5—Table 9.6 summarize the results for this portion of the image survey.

[3] The scale factor is the factor by which the height and width of the bi-level image is divided to produce the 72 ppi zoom.

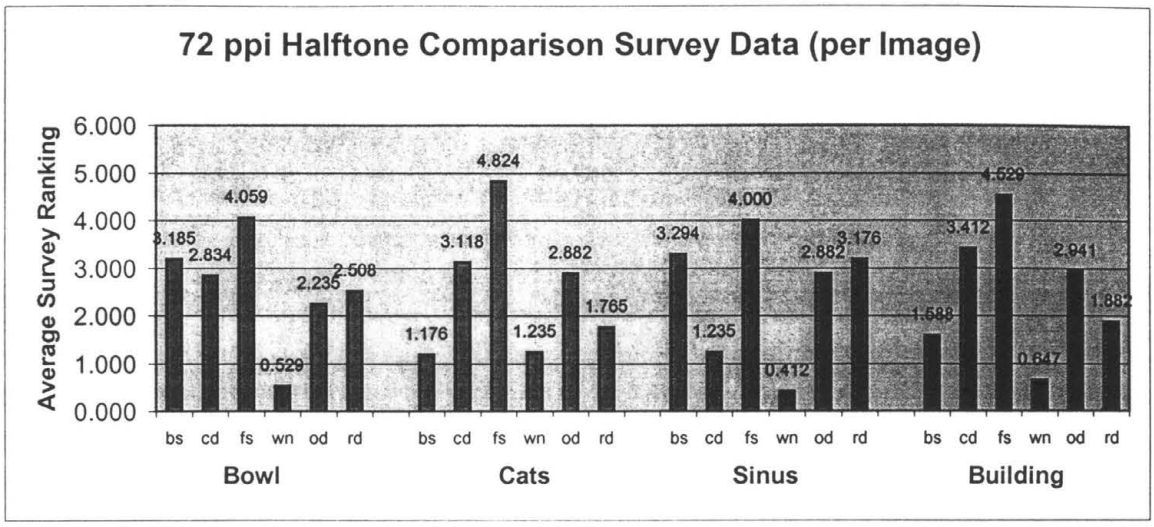

Chart 9.5—72 ppi Halftone Comparison Data

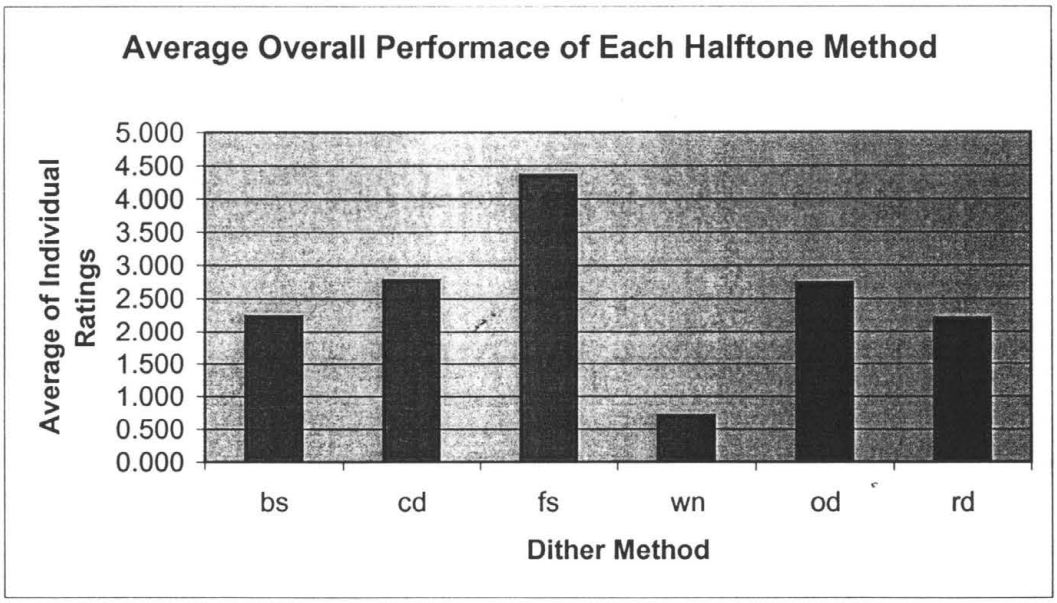

Chart 9.6—Four-Image Average of Individual Halftone Ratings

Image	*Halftone Method*	*Average Subjective Ranking*	*Universal Quality Index*	*Root Mean Square*
Bowl	Fs	4.06	0.017104	118.14
	Cd	3.29	0.014913	119.43
	Bs	2.88	0.018996	117.17
	Od	2.24	0.152370	118.47
	Rd	2.00	0.016587	118.20
	Wn	0.53	0.018186	115.33
Cats	Fs	4.82	0.043414	99.41
	Cd	3.12	0.041127	104.12
	Od	2.88	0.042429	100.36
	Rd	1.76	0.042950	99.74
	Wn	1.24	0.049028	89.92
	Bs	1.18	0.054653	96.86
Sinus	Fs	4.00	0.042834	119.51
	Bs	3.29	0.649618	119.15
	Rd	3.18	0.044099	119.23
	Od	2.88	0.042423	119.55
	Cd	1.24	0.043129	118.32
	Wn	0.41	0.063387	113.27
Building	Fs	4.53	0.032911	112.14
	Cd	3.41	0.079807	112.11
	Od	2.94	0.031204	122.53
	Rd	1.88	0.030021	112.52
	Bs	1.59	0.077669	112.17
	Wn	0.65	0.050843	106.13

Table 9.5—Subjective and Objective Quality Data for Halftoning Method Comparison

Image	*R between Survey Data and UQI*	*R between Survey Data and RMS*
Bowl	-0.115	0.710
Cats	-0.642	0.551
Sinus	0.260	0.839
Building	-0.186	0.416

Table 9.6—Correlation Coefficients Between Subjective and Objective Data per Image

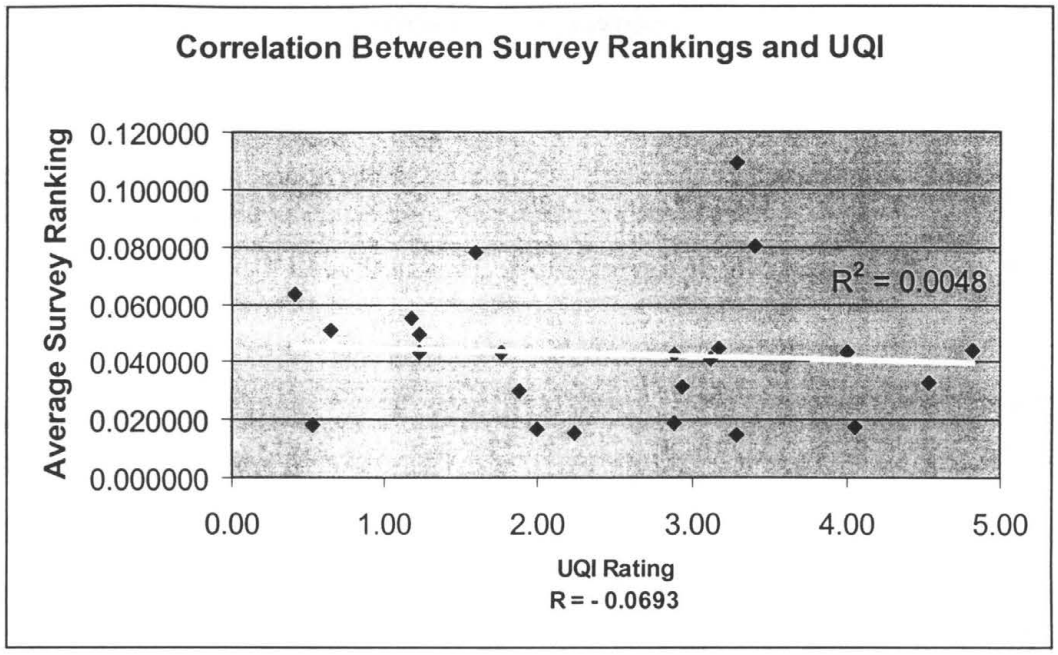

Chart 9.7—Correlation Between Subjective Rankings and
Universal Quality Index for All Halftone Comparison Data

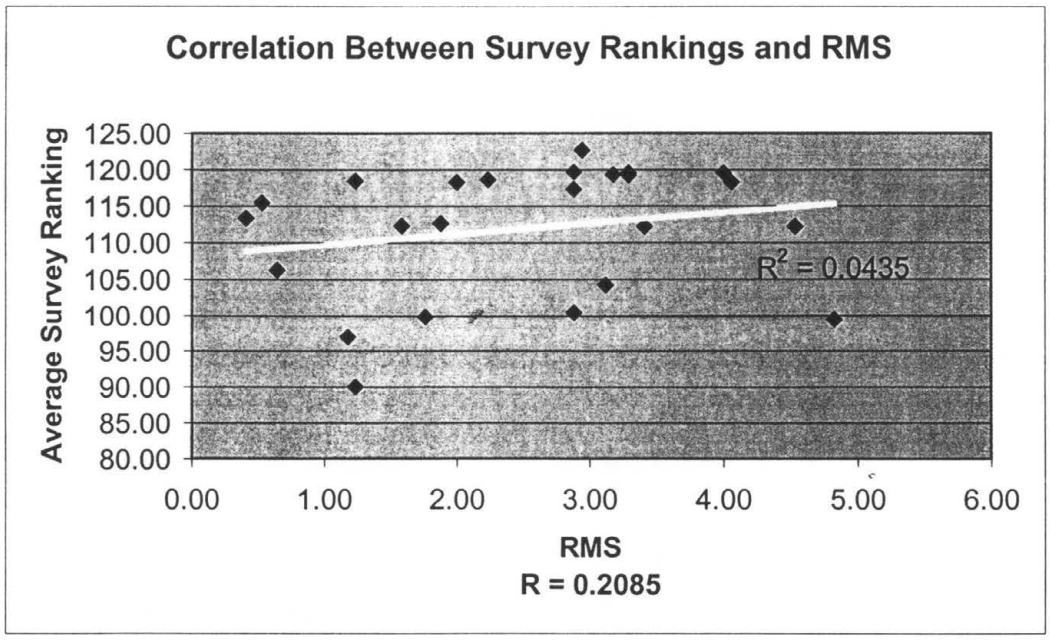

Chart 9.8—Correlation Between Subjective Rankings and RMS
for All Halftone Comparison Data

Chart 9.5 and Chart 9.6 pertain to the subjective assessment of the halftoning methods at 72 ppi. These charts show that at 72 ppi, test subjects prefer Floyd-Steinberg error diffusion (FS) by nearly 2:1 on average. White noise thresholding performed the worst in this survey as expected. The other methods performed relatively similar on average. However, inspection of the individual image data shows that Case's method slightly out ranked the ordered dithers (clustered and dispersed) for vector images. Conversely, the ordered dithers out performed Case's images for natural images, but with a wider gap.

Typically, the ordered dithers produced images with noticeable vertical and horizontal patterns that emerge periodically across the image. Case's halftoning methods contain similar patterns; however, their frequencies of emergence are more random. FS error diffusion contains what many researchers call "serpentine" moirés (Ulichney, 1987; Jahne, 1993). These moirés emerge at diagonal orientations and with little periodicity. According to Floyd & Steinberg (1975), Jarvis (1976), Ulichney (1987), and Jahne (1993) these diagonal patterns are less perceivable by the human visual system; hence, the high average rating of FS.

These results suggest that at 72 ppi (a very low resolution for bi-level images), the orientation and periodicity of a moiré contributes most to the determination of quality. For the diagonally oriented serpentines of FS error diffusion, periodicity seems to matter little. However, when patterns appear along vertical or horizontal orientations, test subjects prefer the more homogeneous moirés of the ordered dithers to the more sporadic patterns of Case's methods. In addition, the fact that the clustered dither outperforms

Case's methods suggests that the number of simulated graylevels produced by a dither is less significant to the perceived quality of the image at 72 ppi.

Table 9.5—Table 9.6 and Chart 9.7—Chart 9.8 present the subjective ratings of the halftoned images along side the respective objective data. Particularly, attention is focused on the correlations (or lack of) between these data sets. The discussion of statistical image quality metrics on page 19 describes the relevance of the correlation between data sets.

Table 9.5 is an exhaustive display of all subjective and objective image quality data. Table 9.6 shows the individual correlations between the survey data and Zhou's and Bovik's *UQI* metric and the *RMS* error metric. This table seems to indicate that a relation may exist between the subjective rankings and the *RMS* value for the halftoned vector images. Admittedly, more data needs to be taken before this conclusion can be made especially because of the information shown by Chart 9.7 and Chart 9.8. These charts indicate that no *general* correlation exists between the subjective data and objective data for the halftoned images.

BS Image Enhancements

Chart 9.9-Chart 9.12 and Table 9.7-Table 9.8 summarize the results for this BS portion of the image enhancements survey.

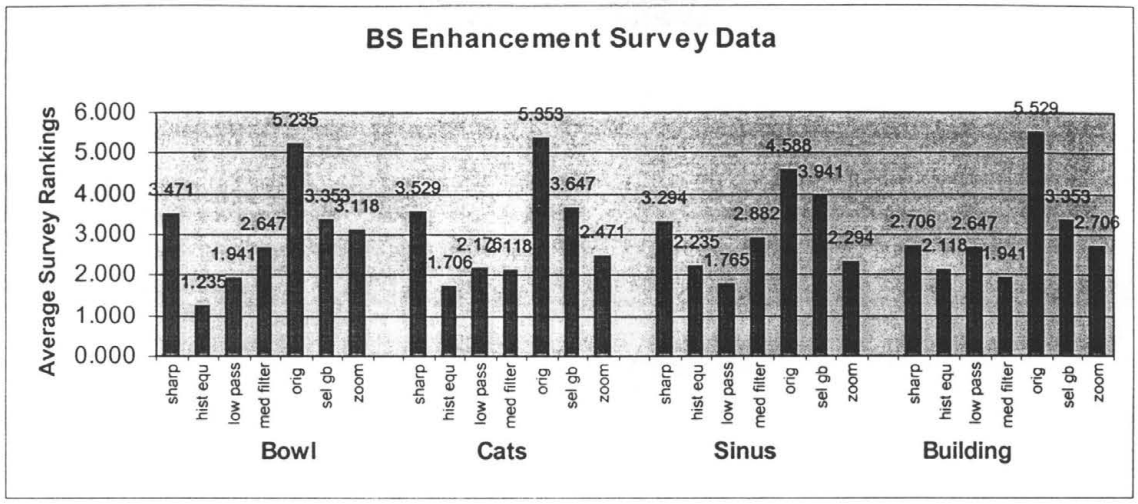

Chart 9.9—Enhancement Survey Data for BS 1:1 Zoom Images

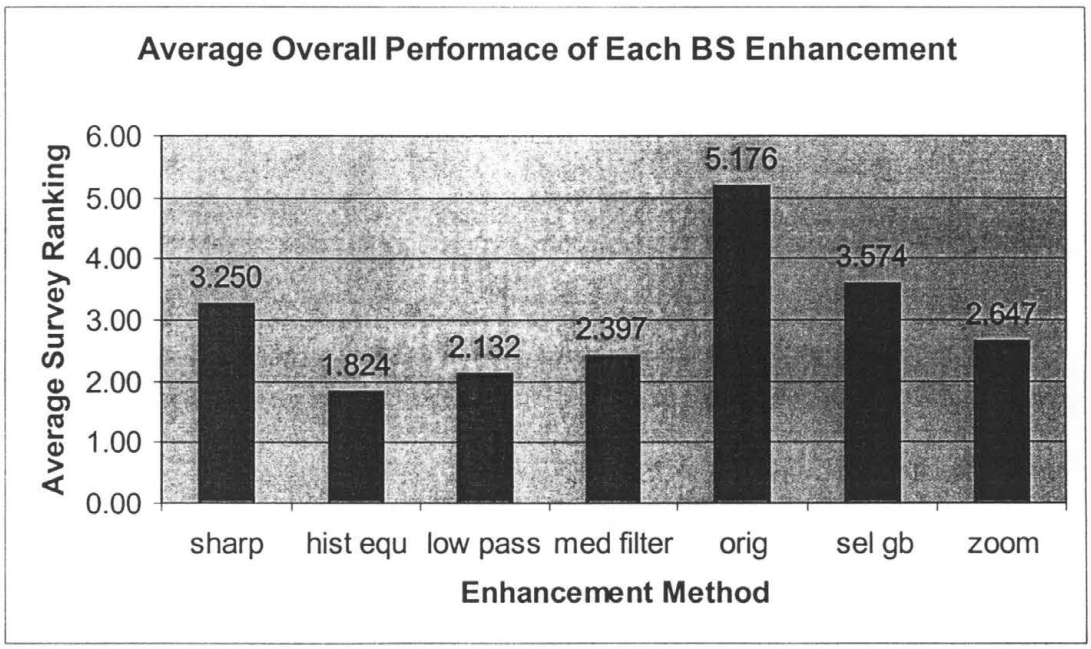

Chart 9.10— Four Image Average of Subjective BS Enhancement Data

Image	*Enhance Method*	*Average Subjective Ranking*	*Universal Quality Index*	*Root Mean Square*
Bowl	Orig	5.24	1.000000	0.00
	Sharp	3.47	0.323150	22.06
	sel gb	3.35	0.356710	19.25
	Zoom	3 12	0.344440	20.34
	med filter	2.65	0.356520	19.66
	low pass	1 94	0.156060	27 28
	hist equ	1.24	0.319060	40.12
Cats	Orig	5.35	1.000000	0.00
	sel gb	3.65	0.466850	15.76
	sharp	3.53	0.278590	24.43
	zoom	2.47	0.354690	19.34
	low pass	2.18	0.443840	17.11
	med filter	2.12	0.426120	16.83
	hist equ	1.71	0.286270	41.36
Sinus	orig	4.59	1.000000	0.00
	sel gb	3.94	-0.007825	56.31
	sharp	3.29	-0.007328	59.76
	med filter	2.88	-0.008046	57.09
	zoom	2.29	0.540780	20.49
	hist equ	2.24	-0.007099	72.67
	low pass	1.76	-0.009164	58.29
Building	orig	5.53	1.000000	0.00
	sel gb	3.35	0.315230	23.65
	sharp	2.71	0.207040	29.53
	Zoom	2.71	0.256420	25.91
	low pass	2.65	0.254890	28.39
	hist equ	2.12	0.234480	32.17
	med filter	1.94	0.280320	24.37

Table 9.7—Subjective and Objective Quality Data for BS 1:1 Zoom Enhancements

Image	*R between Survey Data and UQI*	*R between Survey Data and RMS*
Bowl	0.824	-0.957
Cats	0.799	-0.743
Sinus	0.498	-0.497
Building	0.935	-0.928

Table 9.8—Correlation Coefficients Between Subjective and Objective Quality Data

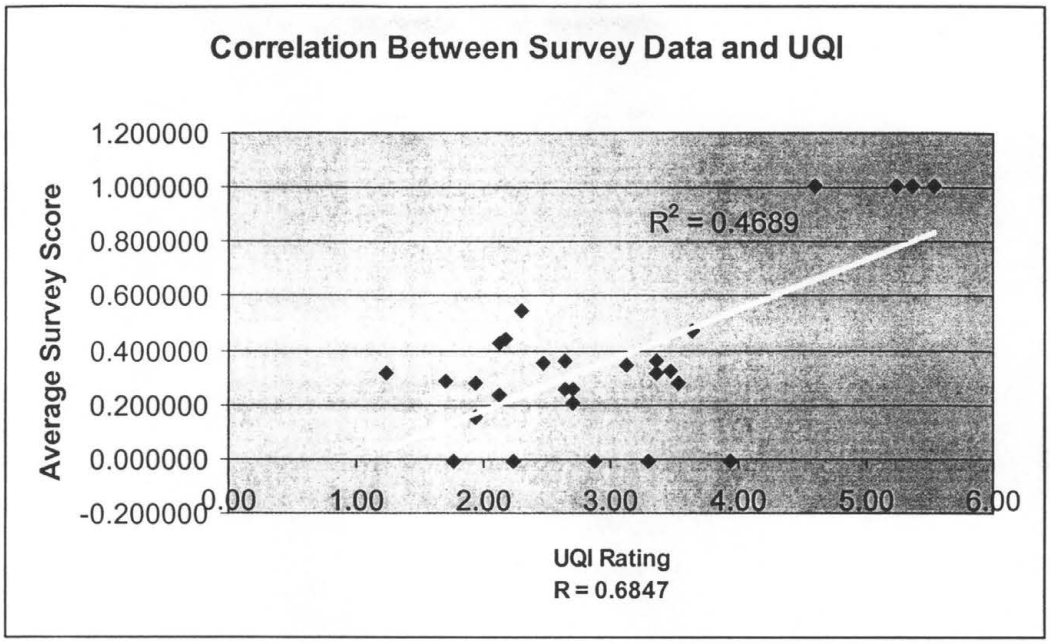

Chart 9.11—Correlation Between Survey Rankings and Univeral Quality Index for All BS Enhancement Data

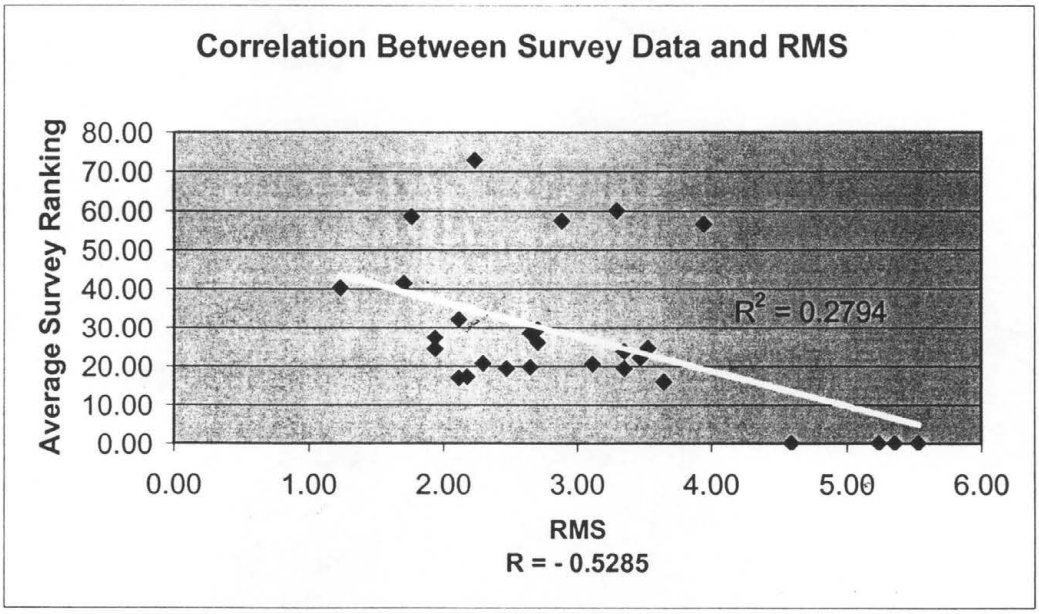

Chart 9.12—Correlation Between Survey Rankings and RMS for All BS Enhancement Data

As expected, Chart 9.10 shows that the original 72 ppi image scored the highest subjective rating.

Two images scored higher than the zoom image: the pre-sharpened image and the post-selective Gaussian blur image. The positive effect of pre-sharpening supports the commonly held notion that halftoning leads to some degree of blurring. Strangely, the data also shows that blurring (in a controlled manner) can lead to visually pleasing images. The selective Gaussian blur process is a "smart" blurring process (van Os, 1998) that adds the following restriction to normal Gaussian blur: pixels neighboring the current pixel with a value outside some delta are not blurred. This restriction allows the blur to correct the halftone-caused moirés previously described without losing an excessive amount of the image's original detail.

Three enhancement methods ranked lower that the 1:1 432 ppi zoom image: the median filtered image, the low-pass Fourier filtered image, and the histogram equalized image. Median filtering of an image is preformed in order to remove "specks" from an image. The term speck refers to any relatively small and random discontinuity in an image. Despite lacking color depth, the BS zooms had little to no specks; thus, the enhancement produced no visual improvement.

Low-pass Fourier filtering modifies an image by operating in the image frequency (Fourier transform) domain. A low-pass filter generally attempts to preserve an image's overall content while blurring unnoticeable details. These details often translate into the high frequency section of the image's Fourier transform. However, moirés exist in the low frequency section of the transform; thus, they pass through such filters. Hence, the transform received a lower subjective rating. A band-pass filter tuned the observed

moirés would probably rate better; however, producing such a filter is itself a process of experimentation.

Histogram equalization is a contrast enhancement technique that produces a uniform histogram for a low contrast image. The method actually fails on the BS zooms because the BS algorithm does not produce enough depth in the available image intensities for the process to work correctly.

Similar to the last section, the objective quality data is compared to the subjective data. Table 9.8 indicates that the subjective data for three of the BS zooms images correlate highly with both objective metrics. The data for the "Sinus" image does not correlate as well. This fact supports the notion generally shared by the subjects that the "Sinus" images are difficult to judge.

Chart 9.11 and Chart 9.12 show that the general correlation for the all of the data pairs is better than for the "Halftone Comparison" section. However, the correlation may be stronger than these charts indicate since the "Sinus" image is causing a detrimental skew.

RD Image Enhancements

Chart 9.13-Chart 9.16 and Table 9.9-Table 9.10 summarize the results for the RD portion of the image enhancement survey.

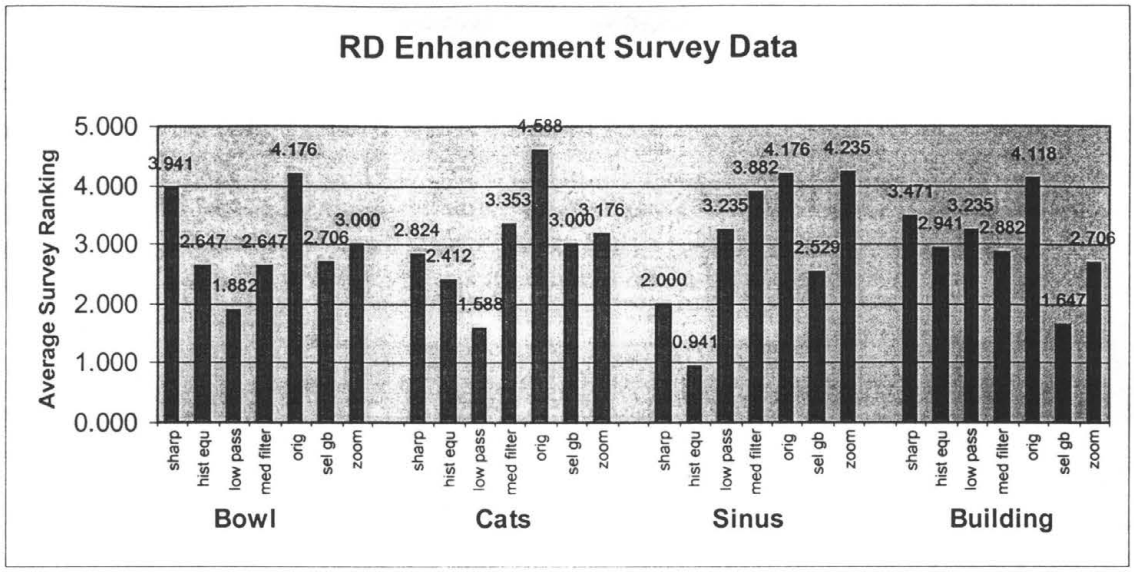

Chart 9.13—Enhancement Survey Data for RD 1:1 Zoom Images

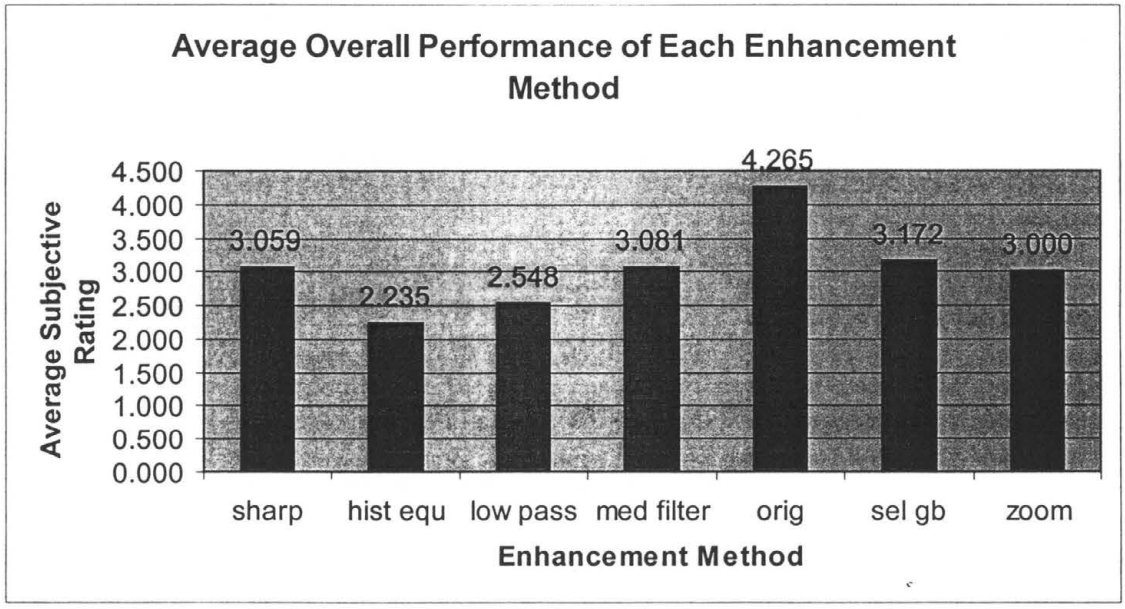

Chart 9.14—Four Image Average of Subjective RD Enhancement Data

Image	*Enhance Method*	*Average Subjective Ranking*	*Universal Quality Index*	*Root Mean Square*
Bowl	Orig	4.18	1.000000	0.00
	Sharp	3.94	0.173040	16.05
	low pass	3.00	0.431970	15.60
	sel gb	2.71	0.735660	9.98
	hist equ	2.65	0.197790	38.49
	med filter	2.65	0.391350	10.38
	Zoom	1.88	0.255390	12.38
Cats	Orig	4.59	1.000000	0.00
	med filter	3.35	0.659620	9.29
	Zoom	3.18	0.490090	12.54
	sel gb	3.00	0.731530	9.22
	Sharp	2.82	0.368700	17.95
	hist equ	2.41	0.384430	38.82
	low pass	1.59	0.664640	11.48
Sinus	Zoom	4.24	-0.002408	56.48
	Orig	4.18	1.000000	0.00
	med filter	3.88	-0.002185	55.58
	low pass	3.24	-0.003234	55.92
	sel gb	2.53	-0.000903	54.02
	Sharp	2.00	-0.003252	57.53
	hist equ	0.94	0.000336	80.86
Building	Orig	4.12	1.000000	0.00
	sharp	3.47	0.265590	23.43
	low pass	3.24	0.451750	24.05
	hist equ	2.94	0.326630	24.40
	med filter	2.88	0.492580	17.80
	Zoom	2.71	0.358540	19.80
	sel gb	1.65	0.624900	17.31

Table 9.9—Subjective and Objective Quality Data for RD 1:1 Zoom Enhancements

Image	*R between Survey Data and UQI*	*R between Survey Data and RMS*
Bowl	0.426	-0.334
Cats	0.585	-0.538
Sinus	0.417	-0.644
Building	0.282	-0.415

Table 9.10—Correlation Coefficients Between Subject and Objective Data

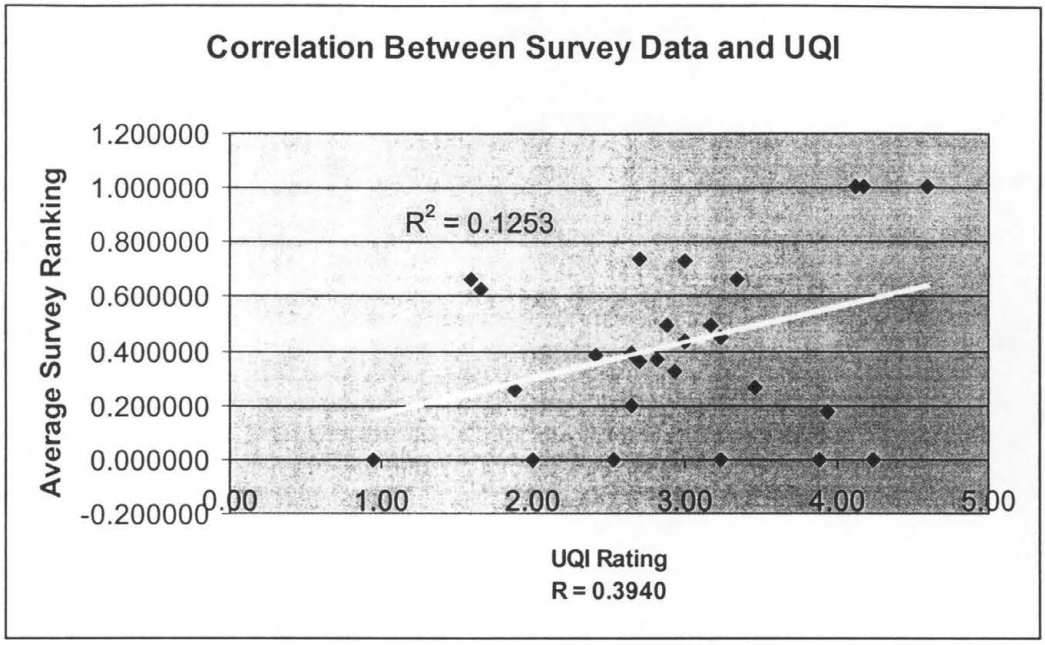

Chart 9.15—Correlation Between Survey Rankings and the Universal Quality Index for All RD Enhancement Data

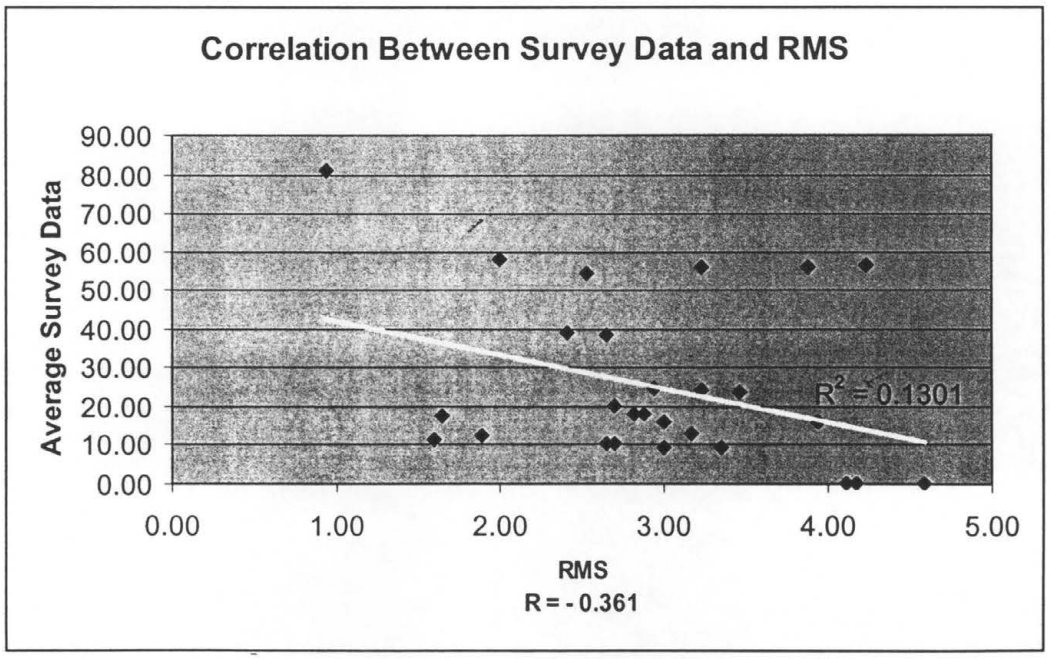

Chart 9.16—Correlation Between Survey Rankings and RMS for All RD Enhancement Data

The subjective ratings for the enhanced RD images are nearly the same as those for the BS with one notable exception: the median filter. As has been established, the RD zooms of this study suffer from periodic box-like moirés. The improved performance (over the BS data) of the median filter for the RD zooms is attributed to its ability to help nullify these moirés.

Another interesting observation can be made by considering the individual image data. For the "Sinus" image, the 1:1 zoom slightly outranks the original image. Test subjects generally spent more time on all of the "Sinus" images than on any of the other images in this survey. Several of the subjects even commented verbally (and frequently) that the image was difficult to judge. These observations of the test subjects may help explain the nature of this result.

Chart 9.15-Chart 9.16 and Table 9.9-Table 9.10 show the correlation data for the subjective and objective quality metrics. Unlike the previous section, the correlation between the data sets is not as strong. This result makes the result of the "BS Image Enhancements" section appear suspect. Thus, it is concluded that more research is necessary to determine if a relation truly exists between the subjective rankings of both sets of enhanced zoom images and the objective data.

Difference Perception

All measurements of the highlights in the images are rounded to the nearest eighth of an inch. Table 9.11 summarizes this information in terms of percentages.

Image	*Percent Highlighted*
BS "Bowl"	24.2%
BS "Cats"	15.6%
BS "Sinus"	14.1%
BS "Building"	28.9%
RD "Bowl"	3.1%
RD "Cats"	2.3%
RD "Sinus"	0.0%
RD "Building"	10.9%

Table 9 11—Highlighted Percentages per Image

These percentages show that test subjects do perceive the differences caused by insufficient graylevels in the BS zooms. Chart 9.17 plots these percentages against the image survey data and shows that a relatively strong inverse correlation exists between this data and that of the image survey. This result gives a preliminary indication that this method may be a usable way to gather quality data; however, the method needs to be tested on more subjects to verify the validity of this apparent relation.

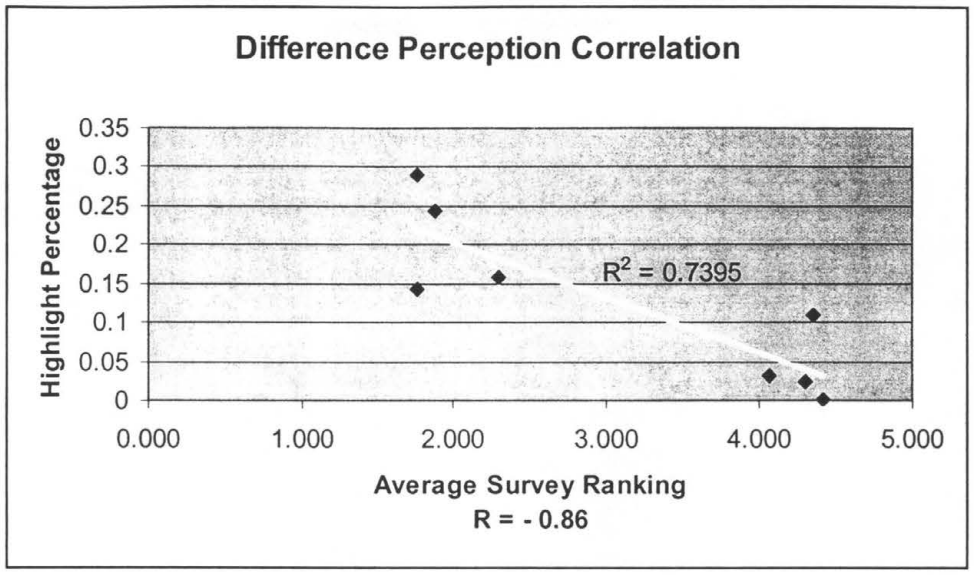

Chart 9.17—Difference Perception versus Survey Rankings

Text Images

Chart 9.18-Chart 9.21 display the results of the image text survey. These charts plot the *readability percentage* of a line of text against the font size of the text. The readability percentage of a line is the percentage of characters correctly identified by a test subject for a line of text. The abbreviations in the chart legends are as follows: "D" implies the dithered version of the image, "Z" implies the zoom; "A" corresponds to antialiased text, while "NA" is non-antialiased text; "S" is a serif font ("Times New Roman") and "SS" is a sans serif font ("Arial").

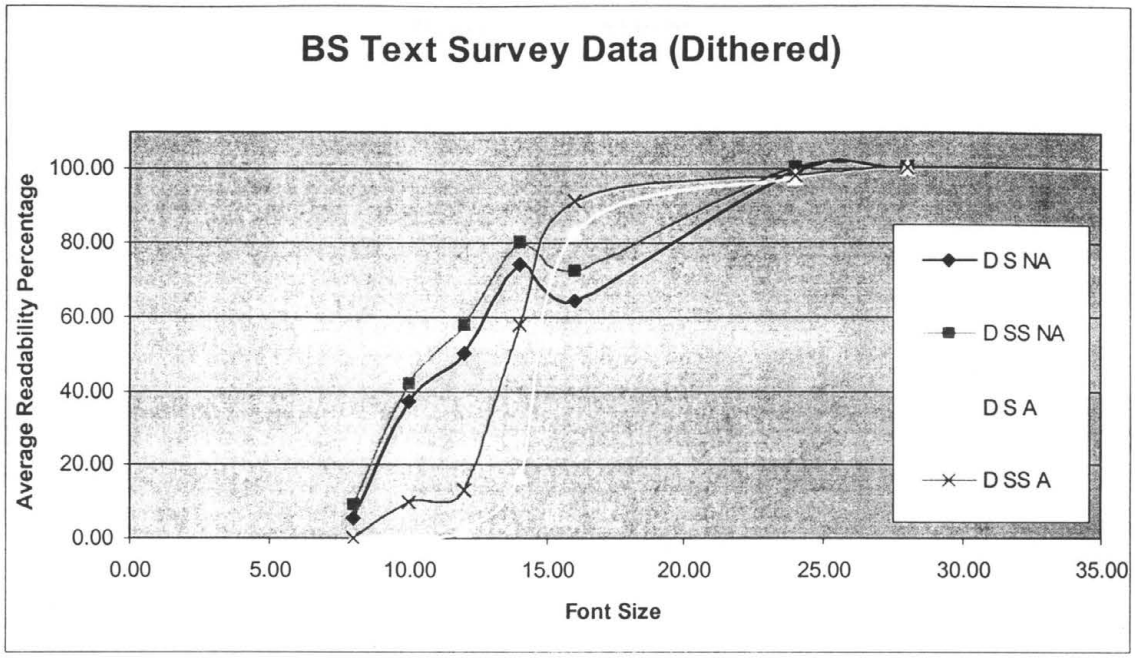

Chart 9.18—BS Text Survey Data (Dithered)

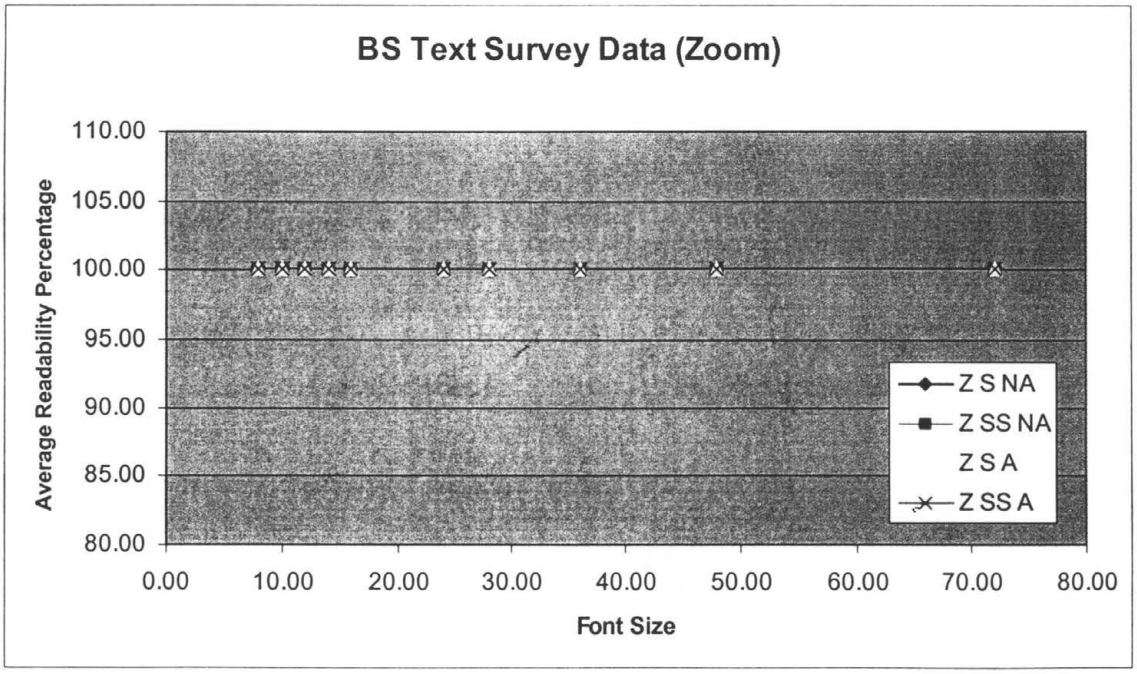

Chart 9.19—BS Text Survey Data (Zoom)

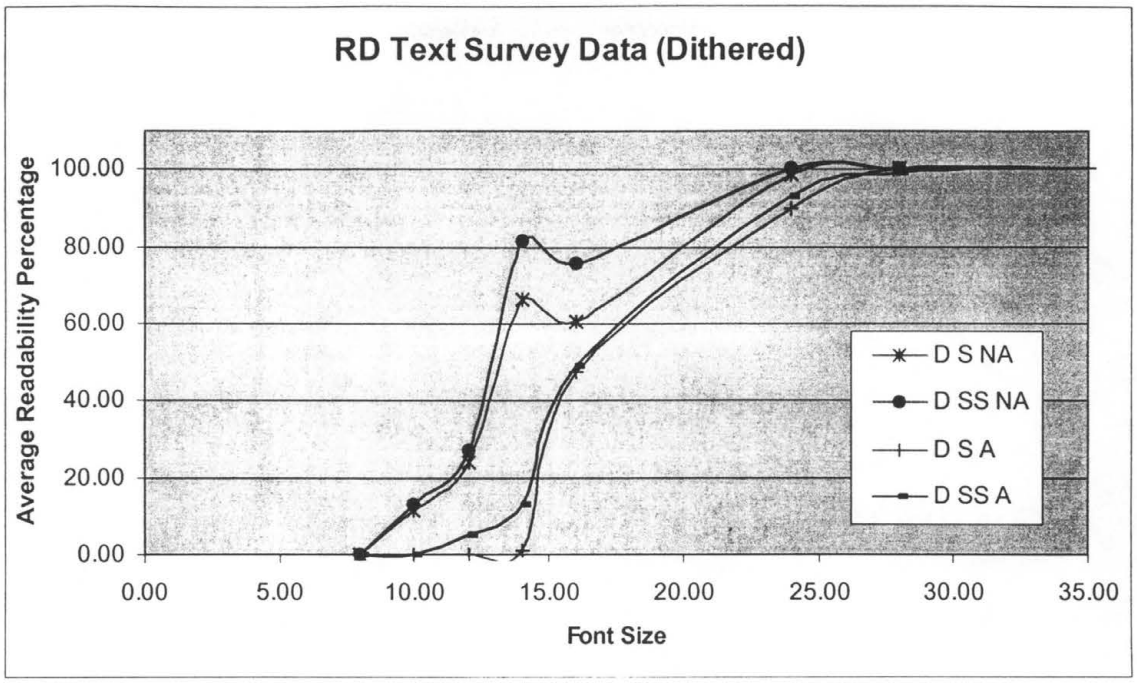

Chart 9.20—RD Text Survey Data (Dithered)

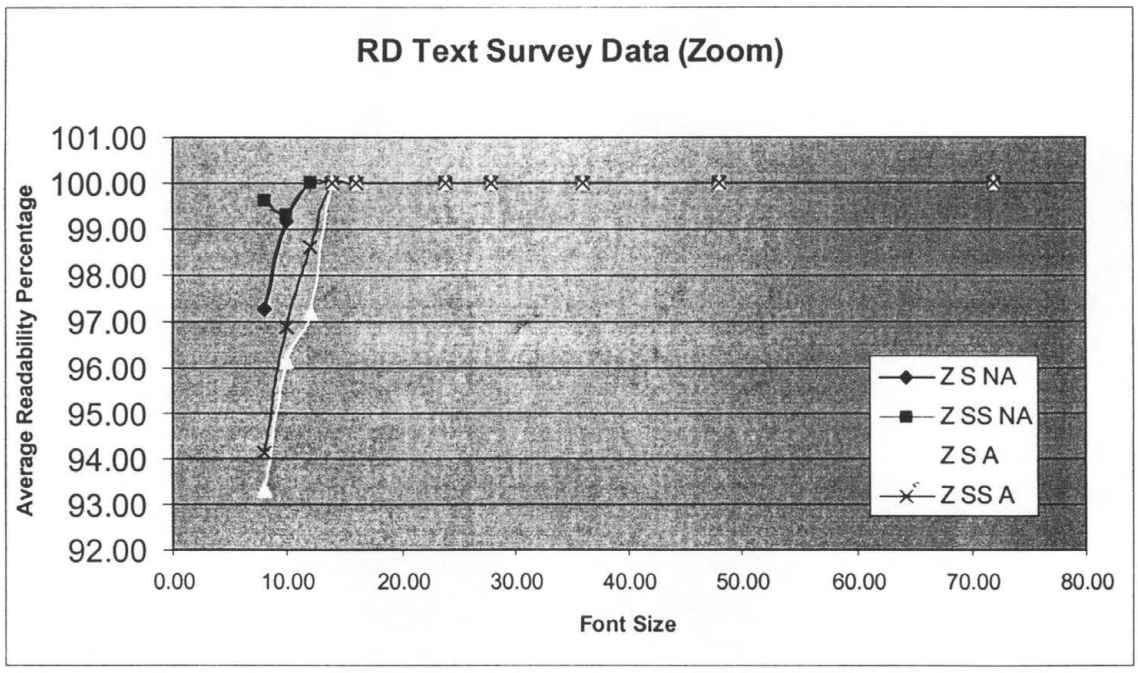

Chart 9.21—RD Text Survey Data (Zoom)

Chart 9.19 and Chart 9.21 show that practically every character of the zoom images is readable by the subjects. Chart 9.18 and Chart 9.20 convey the data for halftoned text images at 72 ppi. They are actually truncated to fonts sizes less than 35 points since, for both charts, all of the fonts greater than 35 have readability percentages of 100%. These charts show that text readability drops greatly once the font size is less than 15 points.

Closer inspection of the dithered non-antialiased fonts shows a slight anomaly between the readability of the 14 and 16 point fonts. The data shows a slight readability increase for the 14 point fonts. This result is explained by the nature of the characters that composed these lines. The 14 point line of text predominantly contained alphanumeric characters. The 16 point line, on the other hand, contained many punctuation characters dispersed throughout the alphanumeric characters. Evidently, this data indicates that such characters are more difficult to read in the 72 ppi dithered images than alphanumeric characters. This interpretation coincides with the fact that punctuation marks typically have few identifying cues due to their size and simplicity.

Compression Analysis

The NOTA and RBE compression methods are analyzed by first studying the properties of the algorithms. Data is then presented on vector images, natural images, and pure text images.

Properties

The properties of the NOTA and RBE algorithms are assessed in four ways. First the time and space complexity of the methods are analyzed. Then the best, worst, and average cases for code size are presented. The effects of pixel rearrangements on NOTA are shown. Finally, the importance of image homogeneity to the RBE method is illustrated.

Time and Space Complexities

Similar to the "Halftone Analysis," the time and space complexity of the NOTA and RBE algorithms is analyzed by first decomposing the processes into a set of primitive linearly ordered subprocedures. The average case complexities of the algorithms are then taken to be the same as those of the most complex (also average case) subprocedure.

Table 9.12 and Table 9.13 describe the time complexities of the NOTA and RBE compression schemes.

	Subprocedure	*Average Time Complexity*	*Justification*
a.	Count Run	$O(M \times N)$	At most, a run could be the length of all of the image's pixels. Counting this maximum run would require $M \times N$ operations. The average case actually requires a fraction of this time.
b.	Encode Run	$O(lg(M \times N))$	Encoding a run requires $lg(M \times N)$ comparison operations for the maximum possible run size.

Table 9.12—Time Complexity Analysis of NOTA Algorithm

	Subprocedure	*Average Time Complexity*	*Justification*
a.	Get $n \times n$ Cell	$O(n^2)$	
b.	Encode the Cell	$O(n^2 \lg(n))$	For each of the $lg(n)$ levels of recursion, at most n^2 operations are necessary.
c.	Output the Code	$O(n^2)$	At worst, some constant number of prefix bits are output along side no more that 75% of the original number of pixel bits (specific for this implementation).

Table 9 13—Time Complexity Analysis of RBE Algorithm

For the NOTA algorithm the counting of a run requires the most time; thus, the NOTA algorithm is $O(M \times N)$. Most of the RBE method time is spent encoding a cell; thus, its time complexity is $O(n^2 \lg(n))$.

	Subprocedure	*Average Time Complexity*	*Justification*
a.	Count Run	$O(1)$	Only one integer variable is required to determine the size of a run.
b.	Encode Run	$O(lg(M \times N))$	Encoding a run requires $lg(M \times N)$ bits at most for the largest possible run size.

Table 9.14—Space Complexity Analysis of NOTA Algorithm

	Subprocedure	*Average Time Complexity*	*Justification*
a.	Get $n \times n$ Cell	$O(n^2)$	Getting a cell requires n^2 bits to store the pixels of the cell.
b.	Encode the Cell	$O(n^2 \lg(n))$	For each of the $\lg(n)$ levels of recursion, at most n^2 bytes of storage are placed on the program stack
c.	Output the Code	$O(n^2)$	At worst, some constant number of prefix bits are output along side no more that 75% of the original number of pixel bits (specific for this implementation).

Table 9.15—Space Complexity Analysis of RBE Algorithm

The space complexity of the NOTA algorithm is only $\lg(M \times N)$ bytes since the encoding step requires the most space. Similar to its time complexity, the RBE algorithm requires $O(n^2 \lg(n))$ space for its encoding step.

Case Analysis

The NOTA algorithm can compress large runs with relatively few bits. Its best case compression is limited only by the manner of its implementation and the amount of memory contained in a computer. For instance, the GNU C compiler limits long integers to a size of maximum value of 2^{32}. Suppose a bi-level image contained this number of black pixels. The NOTA algorithm can compress these 2^{32} bits to 88 bits for a compression ratio of over 48 million to 1.

The average number of NOTA bits per pixel run has been empirically determined by dividing the total number of encoding bits by the number of distinct runs on a per file basis. Without any rearrangement of the input pixels, the average number of bits per run

is 2.05. This number implies that runs of one, two, three, and four pixels are prevalent in such images. For this case, file sizes are typically shrunk by 25%-30%. Using Case's quad parse rearrangement of pixels (subsequently discussed) increases the length of the runs and increases the number of bits per pixel run to 11.94 on average. This number implies that the runs have grown to an average size of approximately 4,000 ($\sim 2^{11.94}$) bits.

At first glance, it may seem that an image composed entirely a bi-level checkerboard are the worst case for the NOTA algorithm. However, this seemingly worst case produces a bit stream of all "1s" and a second application of NOTA on this bit stream quickly produces the best case NOTA compression.

Worst than the checkerboard case is the case in which an image is composed of nothing but runs of two, five, six, or seven. Each of these pixels runs causes *data expansion*. Data expansions represent scenarios in which the number of bits used to encode a run is more than that required to represent the run originally. All runs of two require three encoding bits. This remaining expansion runs require eight encoding bits. These expansion run can cause image files to grow up to 1.5 times the original size. A second application of NOTA to an expanded file is likely to cause addition expansion since the codes for the expansion runs contain many runs of two themselves.

RBE best case is limited by the size of the cell used for encoding. This best case occurs when an $N \times N$ cell of pixels is completely composed of a single dither pattern. For this situation, an RBE code can represent the original N^2 bits by the number of prefix bits plus the size of the one fixed length code used to represent the dither pattern. For a

32×32 encoding cell parsing through 7 patterns using a 3 bit prefix code, the maximum compression is 171 bits to 1.

The average case size of an RBE code is represented by codes containing at least two levels of nested recursion delimiters. Using the same 32×32 cell example, the compression for this average case ranges between 101 bits to 1 and 16 bits to 1.

The RBE has no data expansion cases. Its worst case occurs when a cell of $N \times N$ pixels contains no equal and adjacent dither patterns. The above 32×32 cell example produces a worst case compression of 1.32 bits to 1. Compression for this worst case RBE scenario is attributed to its use of fixed length dither cell codes.

Pixel Rearrangements

NOTA can densely compress large runs of pixels and pixel rearrangements can be used to increase the runs present in a bi-level image without affecting the accuracy of the information. One particular method of rearrangement uses the pixels of the bi-level dither patterns to produce δ^2 smaller versions of the image. FIG shows an example of this idea using an image composed of 2×2 ordered dither patterns.

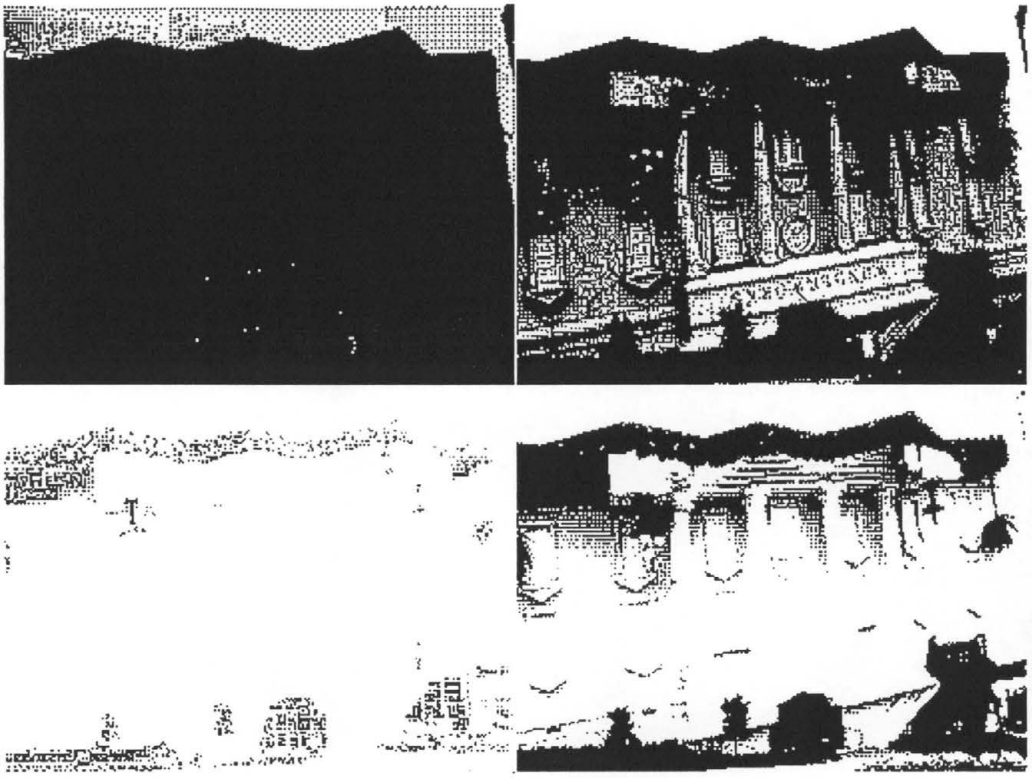

Figure 9.13—Pixel Rearrangements

For this image, four quarter sized subimages are derived from the original bi-level image. The runs for each of the subimages are larger that those of the original. These quarter image run lengths correspond to the probability with which a 2×2 order dither cell pixel is likely to be filled. According to Case (1998), the upper left and upper right pixels of the 2×2 are black and white 80% of the time, respectively. The lower left and lower right pixels are black and white 60% of the time, respectively.

Case calls this method of producing large run quarter images *quad parsing*. Chart 9.22 illustrates the effects that this pixel rearrangement has on the NOTA compression ratios. This chart plots the average observed compression ratios against the file size. Nine different sizes of the four test images contributed to the averages below.

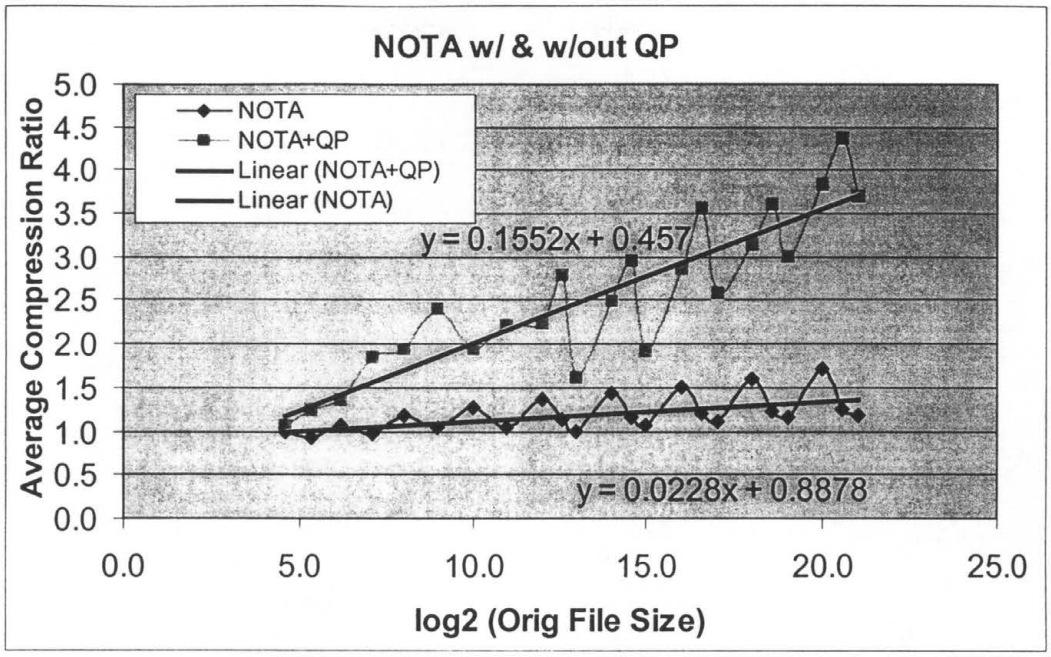

Chart 9.22—Effect of Applying Quad Parsing Before NOTA

As the chart illustrates, the NOTA-QP compression ratios grow at a faster rate than the non QP files. In fact, the slope of the NOTA-QP linear trend line is seven times that of the NOTA linear trend line. The saw tooth shape of both of the curves is due to the variable sized nature of NOTA's codes. Each local maximum occurs as the average run lengths approach the maximum size that can be represented by the current (average) variable sized NOTA code. The local minima correspond to the point at which the run lengths cause the encoder to begin using the next variable sized NOTA code (on average). Because of the improved performance of NOTA due to quad parsing, all subsequent NOTA data has been taken after quad parsing. As such, the term "NOTA" will subsequently imply "NOTA+QP."

Cell Homogeneity

Case's halftoning methods strive for smoothness by relying on checkerboards as their basis. These methods produce bi-level images that can potentially contain large areas of homogeneous dither patterns. The RBE method thrives on such homogeneity.

Halftoning a deep pixel image with a 2×2 BS dither produces a bi-level image with much homogeneity. However, the small size of the dither cells prevents the method from simulating enough graylevels to be practical. A similar situation occurs when using a 2×2 ordered dither matrix to halftone an image.

Applying the RD algorithm over these dithers alleviates this graylevel problem while removing much of this homogeneity in the bi-level image. Figure 9.14 is a qualitative demonstration of this idea. These images are 8×8 cells of pixels extracted from the exact same position in the "Bowl" image.

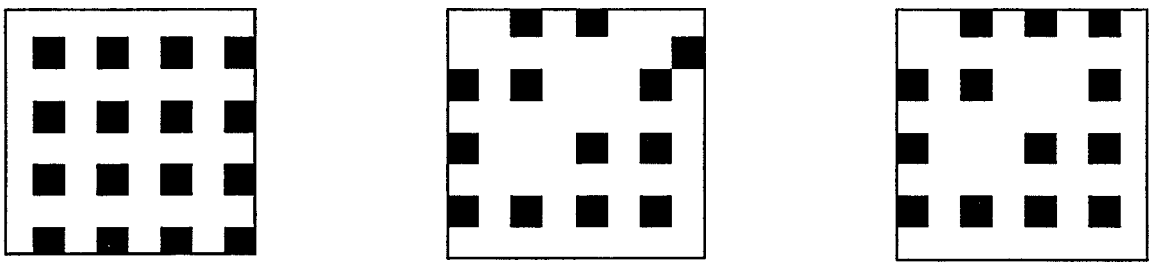

Figure 9.14—Homogeneity Demonstration

The image on the left is very homogeneous in terms of the 2×2 dither patters that compose it. Compared to the other images, this image contains less overall information and can be represented in as little as 3 bits with the RBE method. As shown, the RD

algorithm removes much of this homogeneity while improving the placement of the pixels. This process helps to retain positional and intensity details of the original image. However, as will be shown in the next sections, RBE does not currently perform well in these situations.

Comparisons

The following sections present and analyze the compression data taken for this study. The first section primarily illustrates how NOTA and RBE perform against each other. The next section shows how NOTA and RBE files respond to secondary compressions. The next section compares how mock image formats based on NOTA and RBE perform against other image formats. The final section applies the above test to text images.

Image RLE Comparison

Chart 9.23-Chart 9.30 present how NOTA and RBE perform against each other, the Packbits compressor, and the SimpleRLE compressor.

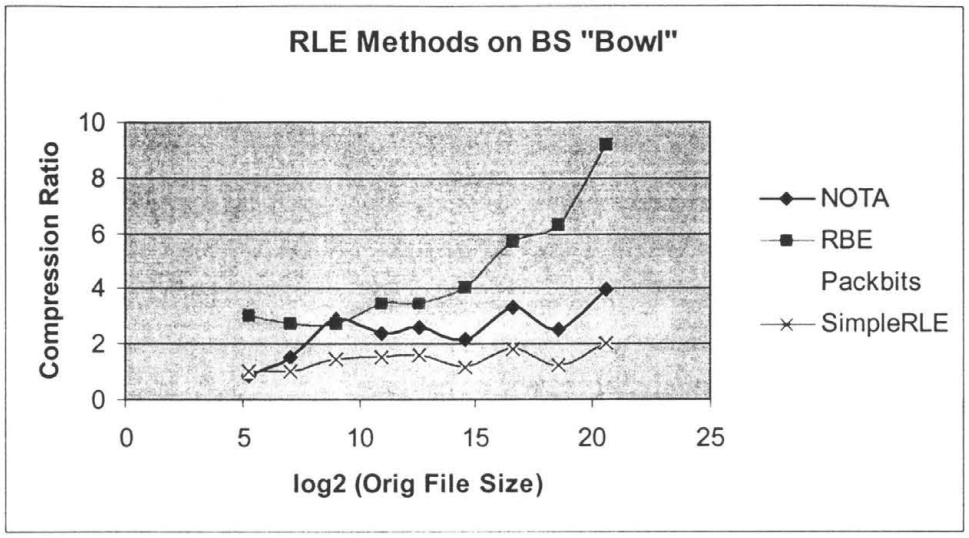

Chart 9.23—RLE Methods on BS "Bowl"

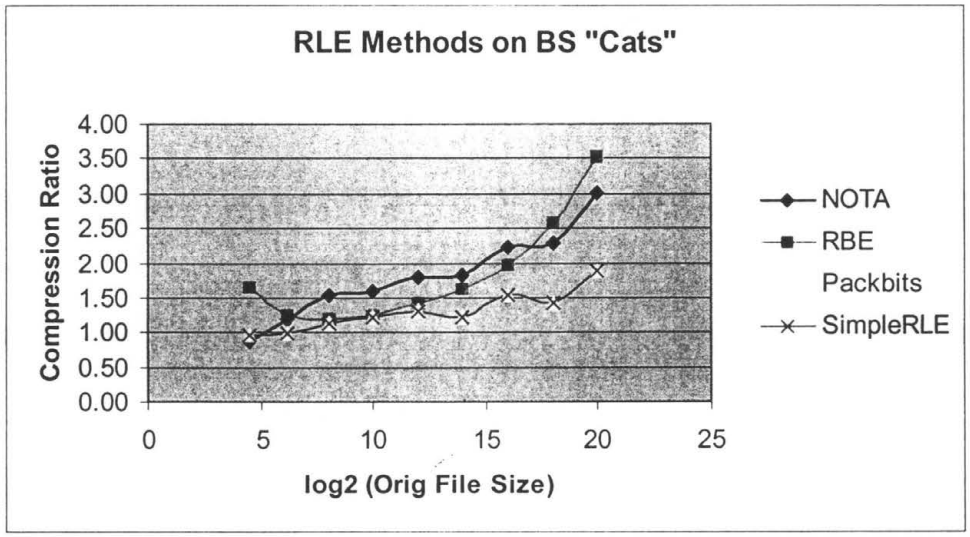

Chart 9.24—RLE Methods on BS "Cats"

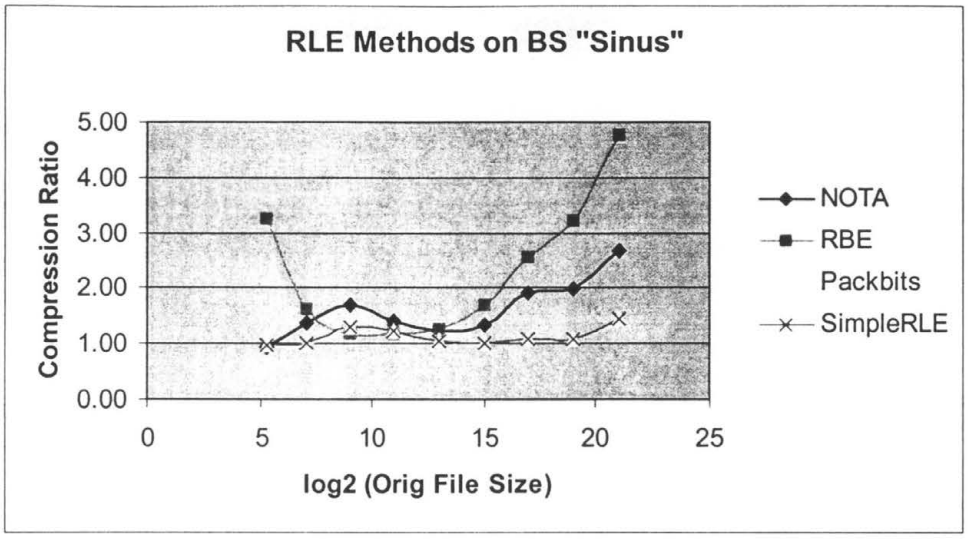

Chart 9.25—RLE Methods on BS "Sinus"

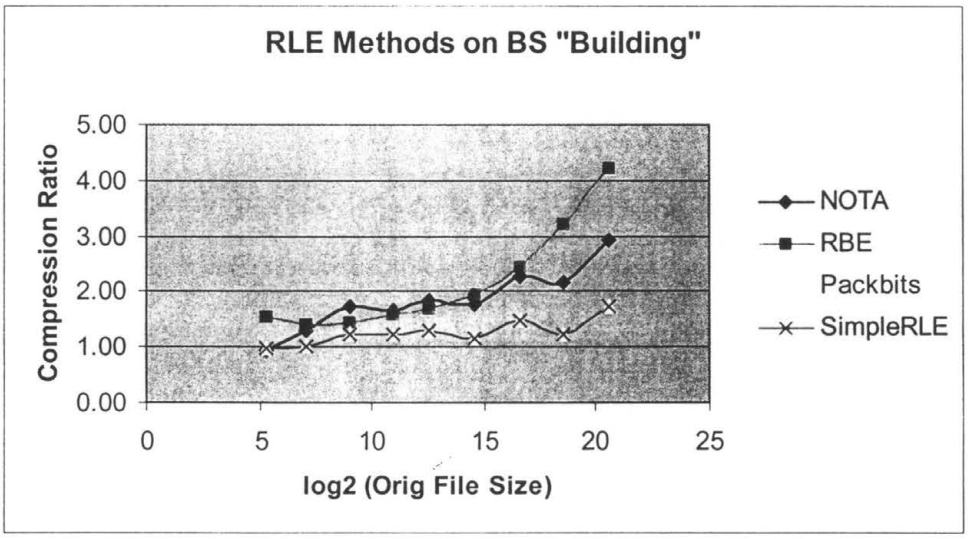

Chart 9.26—RLE Methods on BS "Building"

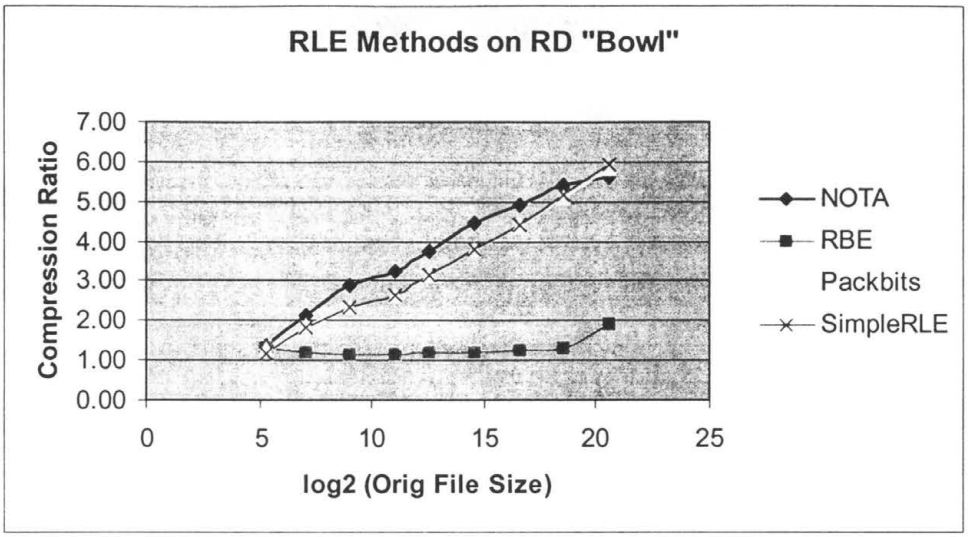

Chart 9.27—RLE Methods on RD "Bowl"

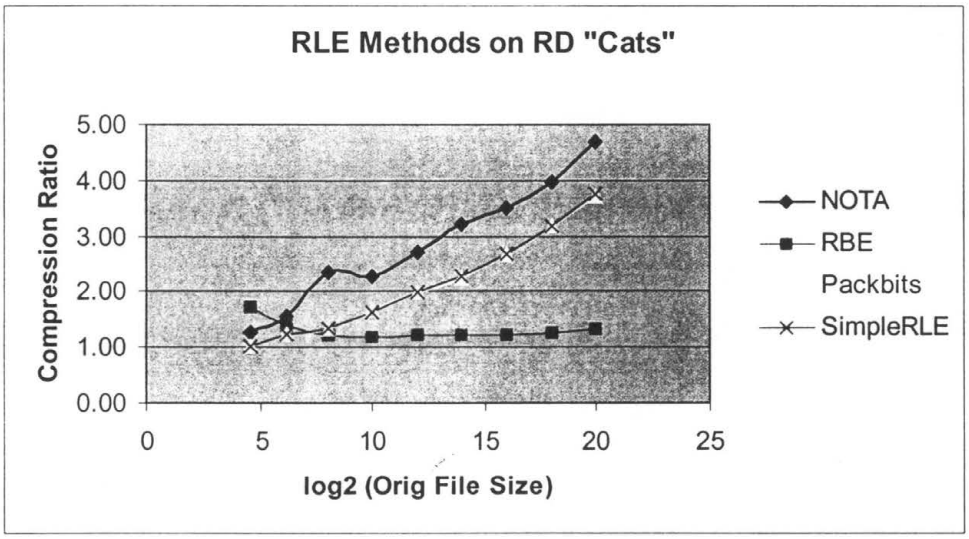

Chart 9.28—RLE Methods on RD "Cats"

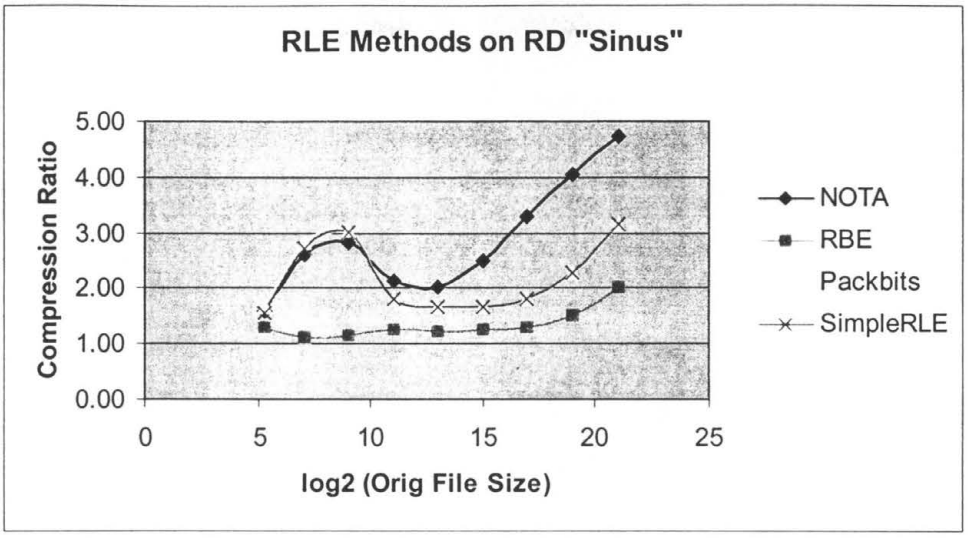

Chart 9.29—RLE Methods on RD "Sinus"

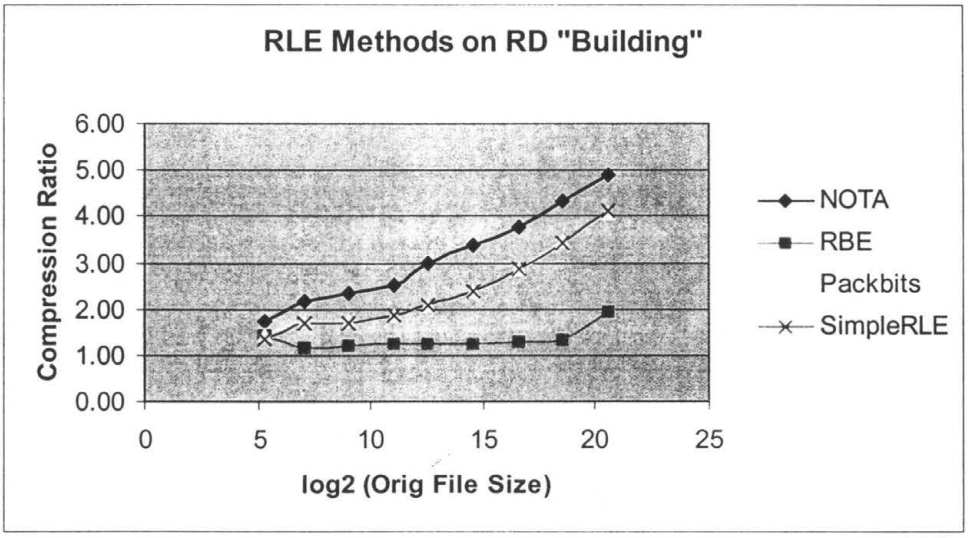

Chart 9.30—RLE Methods on RD "Building"

Firstly, these charts show that Packbits and SimpleRLE are the same algorithms implemented by two different individuals. In every chart, their respective graphs overlap each other.

For the BS charts, RBE generally achieves higher compression ratios than NOTA. For the "Bowl" image, it achieves a compression ratio of 9.12 for a file containing about 1.5 MB of pixels. This is the highest compression achieved for all of the RLE image data. The data also shows that the RBE algorithm performs better on vector images than on natural images. This result for RBE supports the fact that the 2×2 invocation of the BS algorithm produces images with a high degree of homogeneity.

For the RD charts, NOTA overwhelms RBE. In fact, for the RD images RBE consistently performs near its worst case. NOTA, on the other hand, seems to gain some added compression due to its use of the ordered dither. Apparently, the two extra 2×2 patterns output by the BS dither have a slightly negative effect on the bi-level runs in the quad parsed images. Further inspection of the quad parsed images shows that this indeed is the case. Figure 9.15 shows the upper left quarter image of two quad parsed images. The left image is that of RD applied over ordered dither. The right image is an image halftoned using RD over BS.

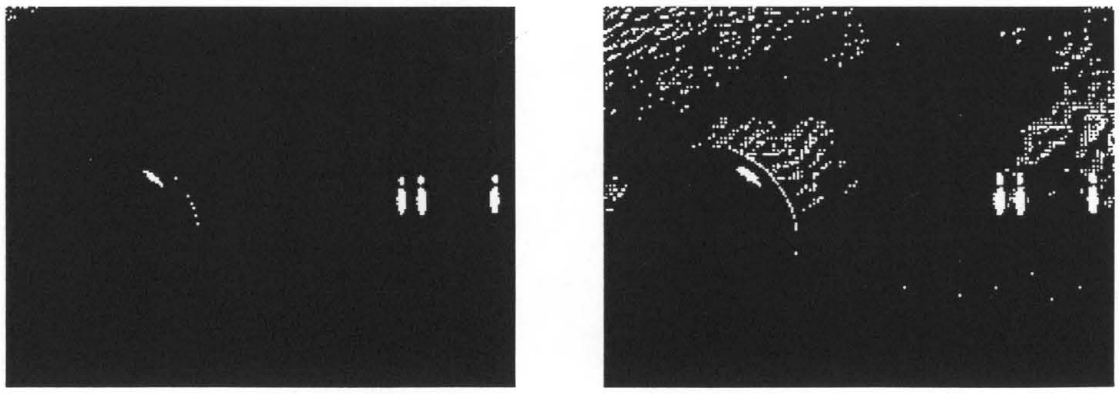

Figure 9.15—The Upper Left Quarter Image of an RD-OD and RD-BS image

Note how the black runs of the BS image are more frequently interrupted by white pixels than those of the ordered dither image.

Secondary Image Compressions

Chart 9.31-Chart 9.38 summarize the secondary compression methods applied over the NOTA and RBE files. The abbreviations for the charts are as follows: "A" implies the use of the arithmetic encoder; "D" implies the use of the dictionary encoder, and "H" implies that the Huffman encoder is used.

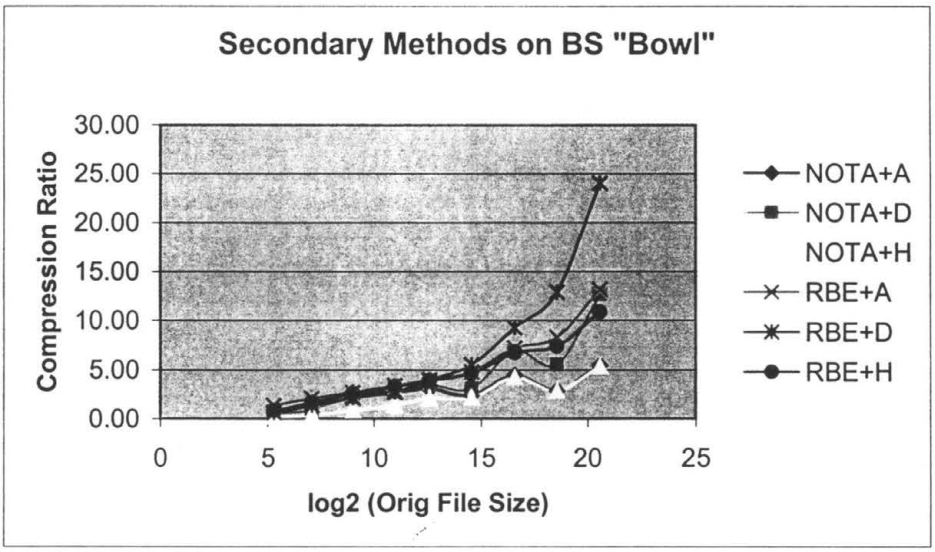

Chart 9.31—Secondary Methods on BS "Bowl"

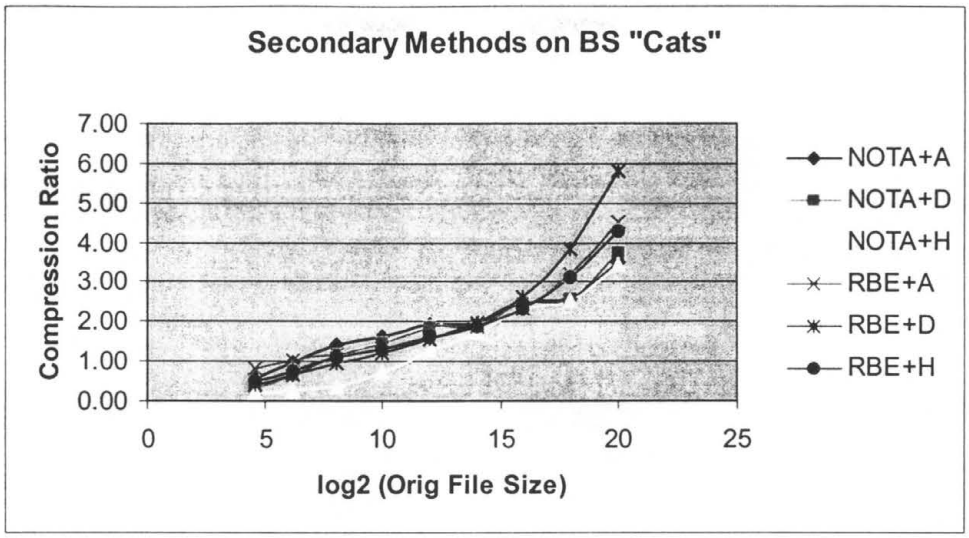

Chart 9.32—Secondary Methods on BS "Cats"

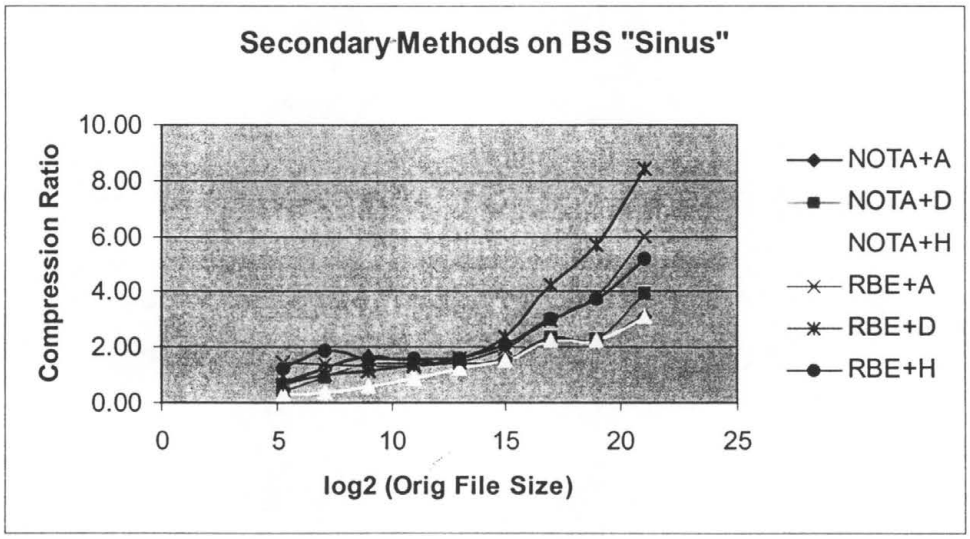

Chart 9.33—Secondary Methods on BS "Sinus"

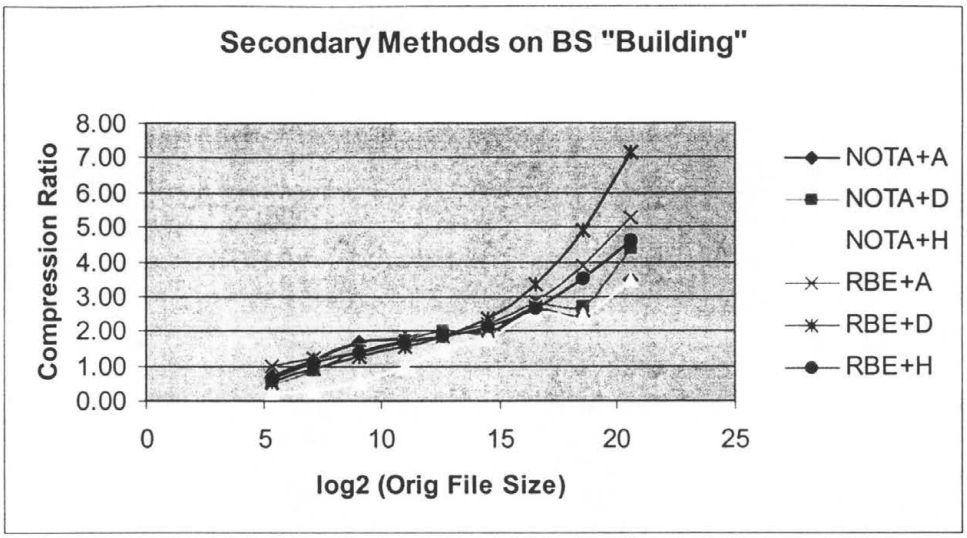

Chart 9.34—Secondary Methods on BS "Building"

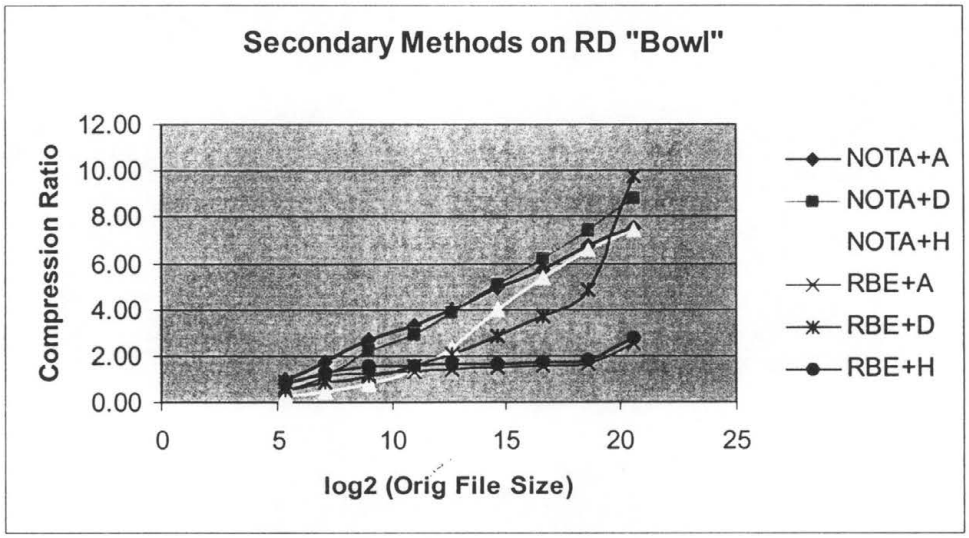

Chart 9.35—Secondary Methods on RD "Bowl"

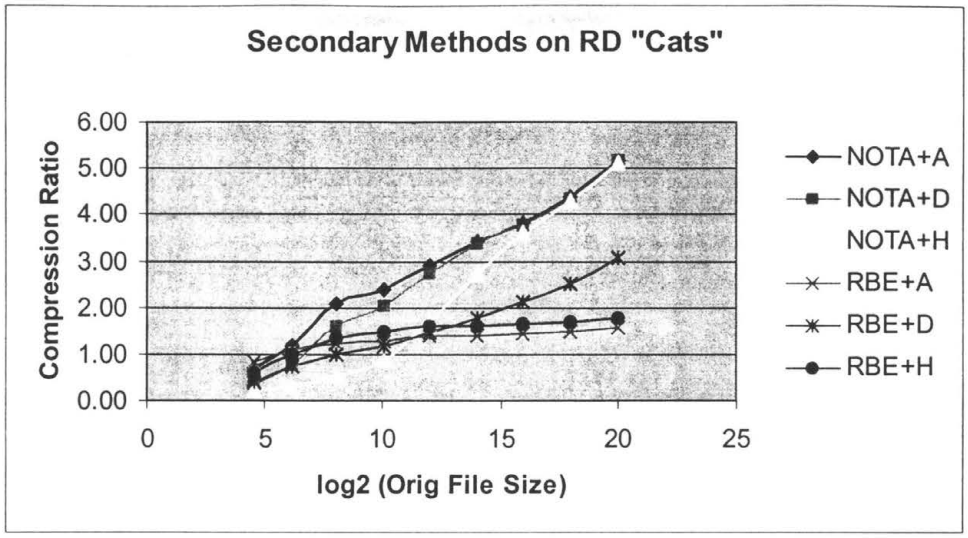

Chart 9.36—Secondary Methods on RD "Cats"

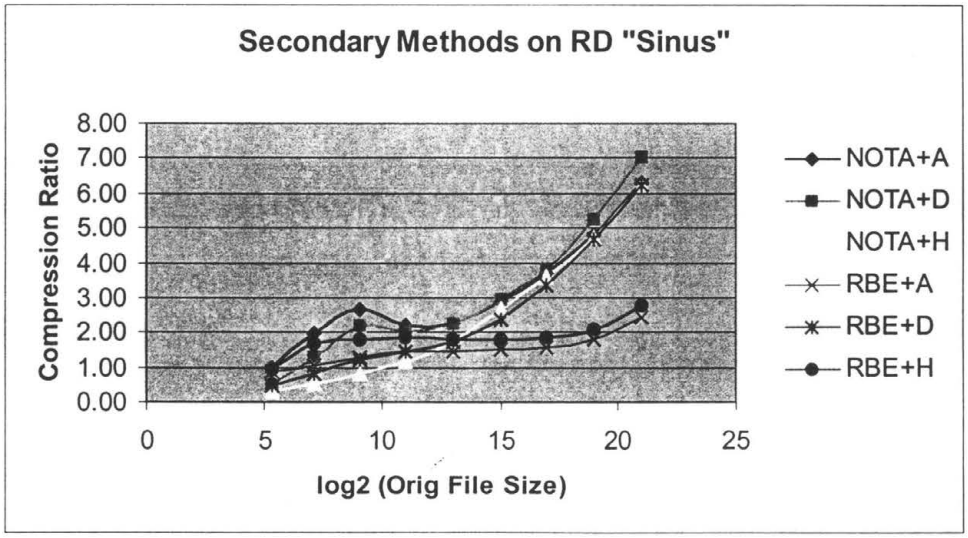

Chart 9.37—Secondary Methods on RD "Sinus"

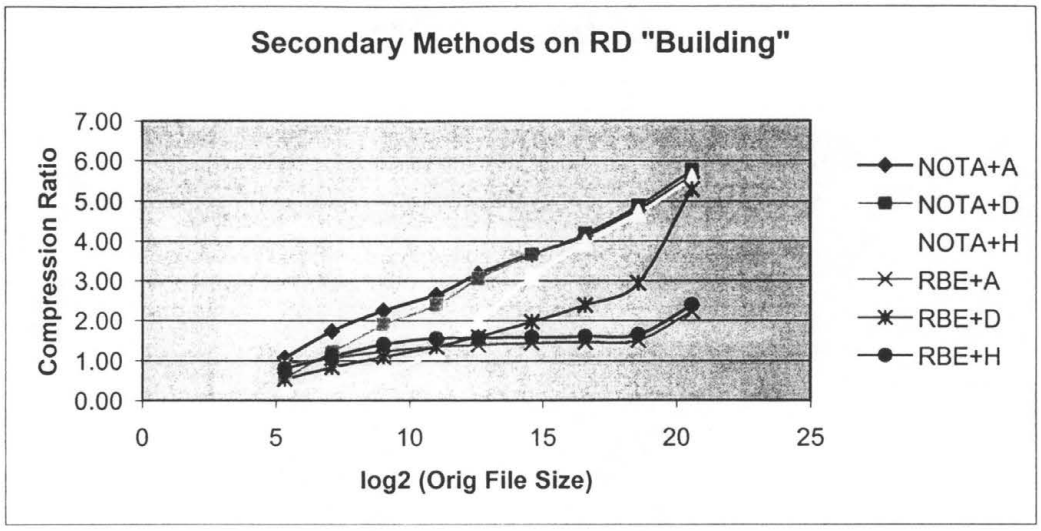

Chart 9.38—Secondary Methods on RD "Building"

Generally, these secondary compressions show an increased compression of ranging from a factor of two to a factor of four. For the most part, the trends of the previous section continue to hold for this data. Namely, the post-RBE performed better on the BS data while the RD files compressed better with the post-NOTA methods.

On the NOTA files, all of the secondary compressions performed about the same; however, gzip ("D") does slightly outperform the others. Gzip played a much more important role on the RBE files. Notice that for three of the four RD charts, the RBE+D data rises sharply once it passes a threshold file size. On the "Sinus" image, this sharp increase causes the RBE method to perform as well as NOTA.

In fact, for all of the input images, gzip applied over RBE produced better secondary compressions than it did on the NOTA files. This result seems to indicate that the 3-bit dither codes used by this implementation of RBE are more predictable than their NOTA counterparts are. Thus, RBE's codes must appear with a large amount of

periodicity in practice. This result seems counterintuitive and was not previously predicted especially because of the long strings of "0s" that NOTA provides in its larger codes.

This data also shows that NOTA's expansion runs cause less trouble than previously expected. This is attributed to the quad parse rearrangement of pixels. In the quarter sized QP images, runs of 2, 5, 6, or 7 are extremely rare.

Image Format Comparisons

Chart 9.39-Chart 9.42 summarize how a mock file format based on NOTA and RBE compares to other popular image formats.

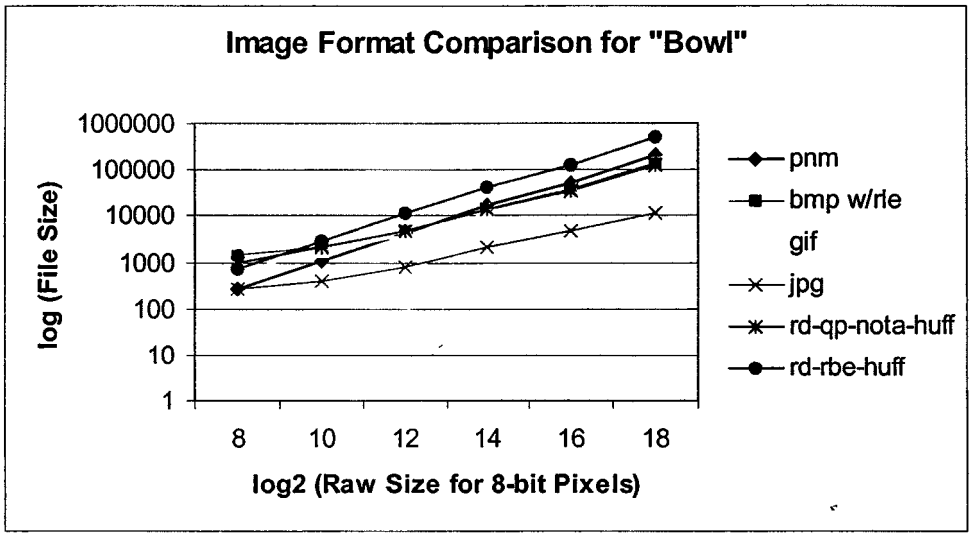

Chart 9.39—Comparison of NOTA and RBE with Other Image Formats for "Bowl"

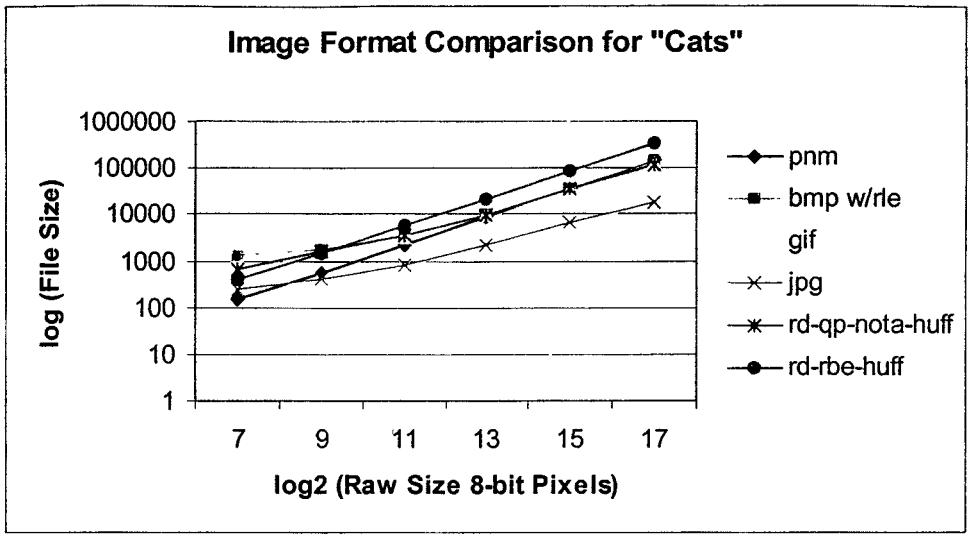

Chart 9.40—Comparison of NOTA and RBE with Other Image Formats for "Cats"

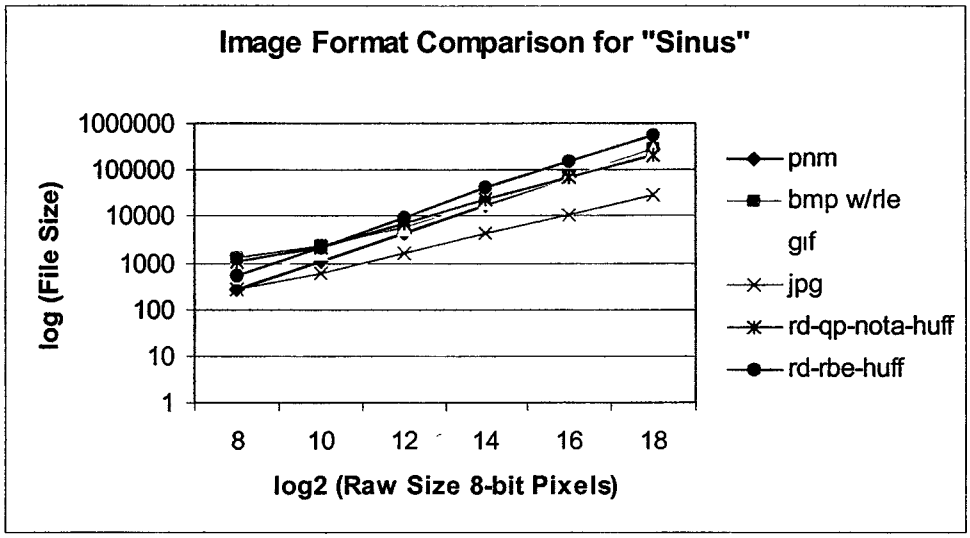

Chart 9.41—Comparison of NOTA and RBE with Other Image Formats for "Sinus"

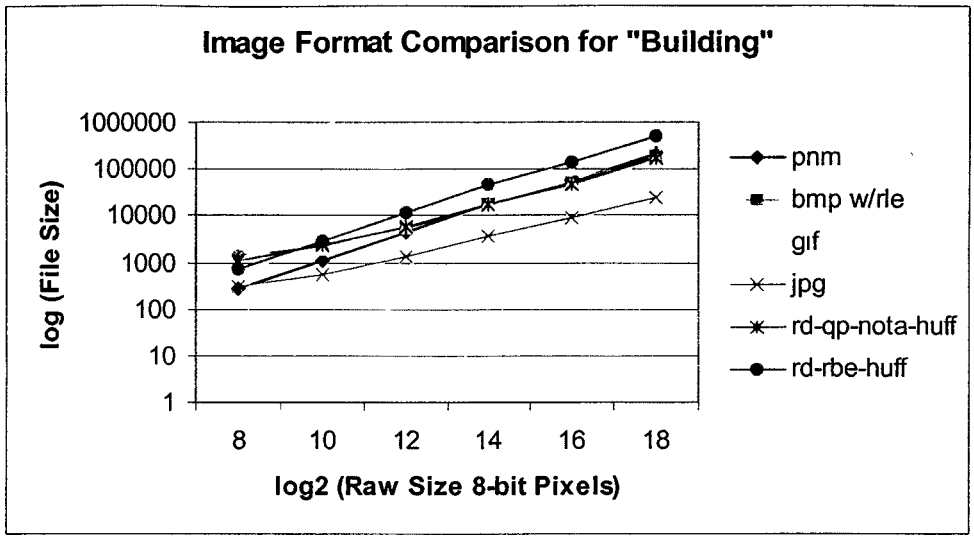

Chart 9 42—Comparison of NOTA and RBE with Other Image Formats for "Building"

As expected, the JPEG files achieve the highest degree of compression. The JPEG data especially shows the power of frequency domain image compressions. However, JPEG typically is best suited for storage of digitized natural images. The RBE mock format generally performs worst. This result coincides with the rest of the data taken for RBE run on an RD image. The rest of the formats perform at nearly the same level.

The mock NOTA format performs at the same level as GIF using a Huffman encoder as its secondary compression. The data from the previous sections indicates that the NOTA format could gain a slight edge over GIF if a dictionary method is used. GIF itself uses a form of Limpel-Ziv dictionary coding as its primary form of compression.

Text Image Compression

Chart 9.43-Chart 9.45 summarize the data taken for the compression of pure text images.

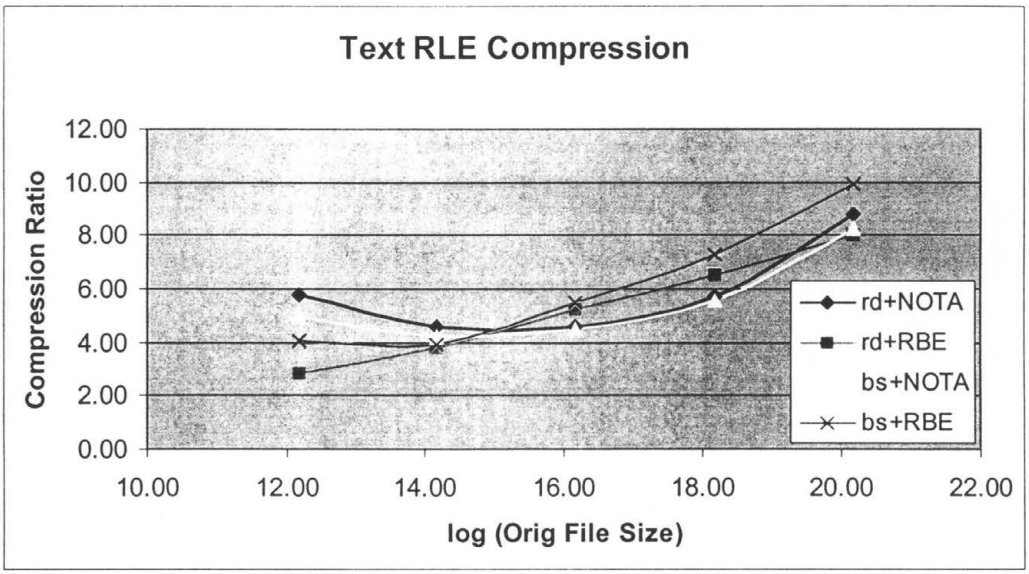

Chart 9.43—Comparison of NOTA and RBE on RD and BS Text

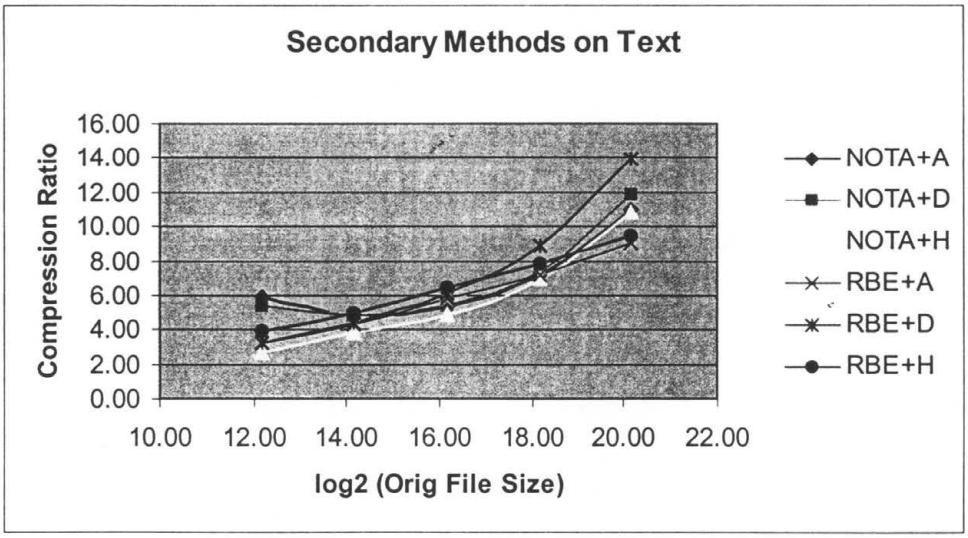

Chart 9.44—Secondary Compressions Applied to NOTA and RBE on RD Text

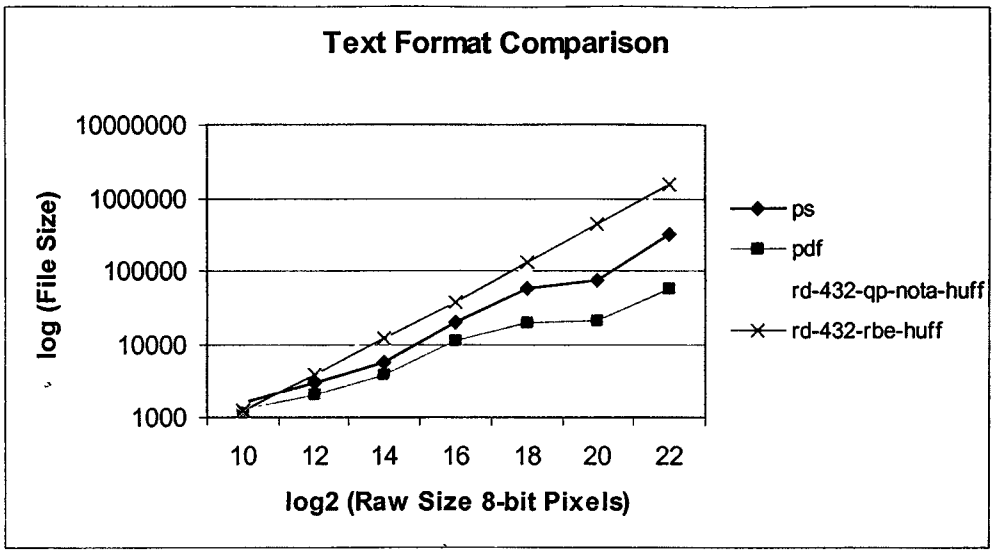

Chart 9.45—Comparison of NOTA and RBE with Other Text Formats

Chart 9.43 shows that these compression methods can achieve higher compression ratios on pure text images as opposed to the vector and natural image of the last section. This is due to the frequent long runs of white pixels in the text images. The RLE methods studied here (like all RLE methods) thrive on these long runs.

Overall, RBE applied to a BS image works best. Unlike for images, the five intensities that can be stored by 2×2 invocation of BS are enough to represent pure text images accurately. This fact is supported by the text survey given in the quality analysis section. However, the five intensities of this scheme cannot support antialiasing. For antialiasing, one of two things can be done. One, NOTA applied on an RD image can be used. Two, a separate antialiasing operation can be applied after the decoding of the image.

Chart 9.44, like the previous image data, shows that RBE responds well to a secondary dictionary compression. The trend for the NOTA data is also the same.

Typically, these secondary compressions achieve additional compression ranging from 120% to 140%.

Chart 9.45 shows that Postscript and Adobe PDF perform much better than the mock file formats of this study. The mock formats are raster style image formats, meaning they store information on every pixel in the image. Postscript and PDF are vector formats that can store data about the placement, size, and orientation of the text in the images. These formats do not store information on every pixel; instead, they derive an image from the rules they that store. Generally, vector formats can store images with less data than their raster counterparts can. This data supports this notion.

CHAPTER 10

CONCLUSIONS

This chapter summarizes the overall findings of this study. It is a collective interpretation of the data described in the previous chapter. The implications of the data are presented along with suggestions for additional issues to be researched.

Summarized Findings

This study considered the image quality of Case's halftoning methods using subjective assessments, objective metrics, and difference perception test. The findings of this portion of the study can be collectively summarized as follows:

- Survey subjects perceived image quality the same regardless of the devise used to display the images.
- No obvious correlation exists between the objective metrics and the subjective survey rankings.
- The frequency, orientation, and periodicity of moirés are significant factors affecting the subjective assessment of image quality.
- Moirés oriented at 0° and 90° are more significant to survey subjects than displayed intensities when assessing image quality.
- 8×8 RD applied to a dispersed-dot ordered dither produces superior zoom images when compared to those created via 2×2 BS.

- The moirés of 8×8 RD applied to BS are less disturbing than those produced when 8×8 RD is applied to a dispersed-dot ordered dither.
- Details are retained better in zoom images when RD is a applied to BS as opposed to when RD is applied to ordered dither.
- Pre-sharpening and post-selective Gaussian blur produce visually pleasing enhancements to the BS and RD zoom images.
- Case's halftoning methods preserve the readability of text as small as 8 points (and possibly smaller).
- 2×2 BS applied to pure text images produces superior results when compared to other halftoned images.

This study considers the compressibility of images produce by Case's halftoning methods in terms of the positional and statistical redundancies present in the pixels composing these images. The compressibility findings of this study can be summarized as follows:

- NOTA is simpler than RBE in terms of time and space complexity.
- NOTA performs better when pixels are rearranged in a manner that produces larger runs.
- NOTA works better on images produced via ordered dither as opposed to those produce by BS.
- Arithmetic, Huffman, and dictionary compressors produce similar secondary compressions when applied over a NOTA encoded file.
- The synergy of RD, NOTA, and Huffman coding produces a mock image storage format that rivals the GIF, BMP, PNG, and PNM formats in terms of file size.
- RBE works well for 2×2 dispersed-dot ordered dither and 2×2 BS; however, the added heterogeneity provided by 8×8 RD is detrimental.
- Due to RBE's dependence on homogeneity, it can possibly be used as a smoothness metric for dithered images.

- The small fixed length pattern codes of RBE respond very well to dictionary compression.

Future Work

These findings support the notion that Case's halftoning methods can be combined with the studied compression methods to produce an image storage format that rivals many of the already existing formats. However, more work would be necessary to produce such a format. This section outlines some of the unresolved issues currently inhibiting the development of such a format.

In order to become a viable image format, this technology needs to be adapted to handle color images. This can be achieved with relative ease by directly applying Case's methods to each of the RGB (red, green, blue) channels often used to store such images.

Neither the NOTA nor the RBE compression methods are perfect and both can benefit from improvements. Particularly, NOTA needs its expansion runs removed or nullified. This need may be realized indirectly through the study of alternative pixel rearrangements or directly through changes to the NOTA algorithm. Alternative pixel rearrangements may also help improve NOTA's performance when applied to BS images. RBE must be adapted to better handle the less homogeneous cells produced by the RD algorithm. These adaptations are likely to occur on the recursion delimiters used by the method.

The accuracy of the correlation charts of Chapter 9 can be improved by devising a subjective ranking system that promotes equality of the differences between the ranks. In

other words, such a method would ensure that the difference between ranks "1" and "2" is the same as the difference between "5" and "6". Statistics theory relies on this property; thus, the current correlation data can only serve as a decent approximation.

Finally, the programs developed for this study are merely prototypes. At the moment, they run slow and some of them contain small bugs. Once the other issues are resolved, these prototypes will be discarded and "production grade" implementations developed from scratch will replace them. These programs can benefit from threaded implementations since none of the algorithms prohibits parallel processing.

REFERENCES

Ahumada, A. J., & Null, C. H. (1993). Image Quality: A Multidimensional Problem. In Watson, A. B. (Ed), Digital Images and Human Vision (pp. 141-149). Cambridge, MS: The MIT Press.

Avcibaş, I., & Sankur B. (1999). Statistical Analysis of Image Quality Measures. Retrieved Oct. 20, 2001 from the World Wide Web

Baxes, G. A. (1984). Digital Image Processing: A Practical Primer. Englewood Cliffs, NJ: Prentice-Hall

Bayer, B. E. (1973). An Optimum Method for Two Level Rendition of Continuous-Tone Pictures. Proc. IEEE. (26) 11-15

Boehm, B. (1988), A Sprial Model for Software Development and Enhancement. Computer, 21, 5, 61-72.

Bourke, P. (1993). A Beginners Guide to Bitmaps. Retrieved June 22, 2001 from the World Wide Web: http://astronomy.swin.edu.au/pbourke/dataformats/bitmaps

Bowers, H., & Bowers, J. S. (1990). Process for Providing Digital Halftone Images with Random Error Diffusion. Retrieved May 18, 2001, from USPTO on-line database (US Pat. No. 5107346) on the World Wide Web: http://www.uspto.gov

Case, R. M. (1993). Method for Reproducing an Image. Retrieved May 14, 2001, from USPTO on-line database (US Pat. No. 6002493) on the World Wide Web: http://www.uspto.gov

Case, R. M. (1998). Youniverse Systems Website. (Last) Retrieved Jan 4, 2002, from http://www.youniverse.com

Case, R. M. (2001). Untitled Patent Regarding Reverse Diffusion.

Chellappa R., & Sawchuk A. A. (Eds.). (1985). Digital Image Processing and Analysis. (Vol. 3). Silver Spring: Computer Society Press

Clarke, R. J. (1995). Digital Compression of Still Images and Video. San Deigo, CA: Academic Press.

Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (2000). Introduction to Algorithms.

Cambridge, MA: MIT Press

Crinion, R. J., Jeng, & Yih-Chyun (1987). Method and Apparatus for Carrying Out a Dithering Operation. Retrieved May 18, 2001, from USPTO on-line database (US Pat. No. 4800443) on the World Wide Web: http://www.uspto.gov

Daly, S. (1997). The Visible Differences Prdeictor: An Algorithm for the Assessment of Image Fidelity. In Watson, A. B. (Ed), Digital Images and Human Vision (pp. 179-206). Cambridge, MS: The MIT Press.

Dasarathy, B. V. (1995). Image Data Compression. Los Alamitos, CA: IEEE Computer Society Press.

Davis, A., & Sitaram, P. (1994). A Concurrent Process Model for Software Development. Software Engineering Notes, ACM Press, 19, 2, 38-51.

Deltener, J. (2000). Reading the Windows .BMP Graphic File Format. Retrieved June 22, 2001 from the World Wide Web: http://www.inversereality.org/tutorials/graphics%20programming/BitmapLoading.html

Dougherty, E. R. (Ed.). (1994). Digital Image Processing Methods. New York, NY: Marcel Dekker

Farrell, J., Trontelj, H., Rosenburg C., & Wiseman J. (1991). Perceptual Metrics for Monochrome Image Compression. SID Digest, 22, 631-634

Foley, J. D., van Dam, A., Feiner, S. K., & Hughes, J. F. (1990). Computer Graphics: Principles and Practice. Reading, MA: Addison-Wesley

Foster, L. R. (2000). Palm OS Programming Bible. Chicago, IL: IDG Books Worldwide.

Girod, B. (1997). What's Wrong with Mean-Squared Error? In Watson, A. B. (Ed), Digital Images and Human Vision (pp. 207-220). Cambridge, MS: The MIT Press.

Gonzales, R. C., & Wintz, P. (1987). Digital Image Processing (2nd ed.). Reading: Addison-Wesley

Graham, C. H. (1965) Vision and Visual Perception. New York: John Wiley and Sons, Inc.

Hearty, P. J. (1997). Achieving and Confirming Optimum Image Quality. In Watson, A. B. (Ed), Digital Images and Human Vision (pp. 149-162). Cambridge, MS: The MIT Press.

Higgins, G. (1977). Image Quality Criteria. Journal of Applied Photographic Engineering, 3, 2, 53-60.

Jahne, B. (1993). Digital Image Processing: Concepts, Algorithms, and Scientific Applications (2nd ed.). Berlin, Germany: Springer-Verlag.

Jonhnson, R. (1984). Elementary Statistics (4th ed.). Boston, MA: Duxbury Press

Kia, O. E. (1997). Document Image Compression and Analysis, Retrieved Aug. 8, 2001 from the World Wide Web: http://www.cfar.umd.edu/~kia/Publications/Thesis/Thesis-HTML.html

Kotera, H., Naka M. (1985). Method and apparatus for generating pseudo-halftone dots by comparing gray scale values of an original with dither threshold values stored in cells of a matrix array divided into imaginary matrices of elemental areas each containing one cell. Retrieved May 18, 2001, from USPTO on-line database (US Pat. No. 4736254) on the World Wide Web: http://www.uspto.gov

Levine, M. D. (1985). Vision in Man and Machine. New York, NY: McGraw-Hill

Lubin, J. (1997). The Use of Psychophysical Data and Models in the Analysis of Display System Performance. In Watson, A. B. (Ed), Digital Images and Human Vision (pp. 163-178). Cambridge, MS: The MIT Press.

Mannos, J. L., & Sakrison, D. J. (1974). The Effects of a Visual Fidelity Criterion on the Encoding of Images. IEEE Transactions on Information Theory, IT-20: 526-536.

Maron, M. J. (1987). Numerical Analysis: A Pracitcal Approach (2nd ed.). New York, NY: Macmillan

Mastro, J. (2000). VR3 Hardware Specifications. Retreived May 20, 2001, from the World Wide Web: http://developer.agendacomputing.com/resources.html

Nelson, M. (1995). The Data Compression Library. Retrieved Oct. 23, 2001, from http://www.dogma.net/DataCompression

Os, T. (1998). Gimp Plug-Ins. Retrieved Sept. 3, 2001 from the World Wide Web: http://thom.best.vwh.net/gimp/

Palm (2000). Palm OS Platform Developers. Retrieved May 21, 2000 from the World Wide Web: http://www.palmos.com/dev

Pratt, W. K. (1978). Digital Image Processing. New York: Wiley & Sons

Pressman, R. S. (1997). Software Engineering: A Practitioner's Approach. St. Louis, MO: McGraw-Hill

Rhodes, N., McKeehan, J., & Stone, M. (Ed.) (1999). Palm Programming: The Developer's Guide. O'Reilly & Associates.

Rohaly, A. M., & Ahumada, A. J. (1995). A Comparison of Image Quality Models and Metrics Predicting Object Detection. Retrieved Aug. 20, 2001 from the World Wide Web: http://vision.arc.nasa.gov/publications/sid95amr/text.html

Roorda A and Williams D. R. (1999) The arrangement of the three cone classes in the living human eye. *Nature* 11:520-522.

Rosen, K. H. (1995). Discrete Mathematics and Its Applications. St. Louis, MO: McGraw-Hill

Russ, J. C. (1999). The Image Processing Handbook. (3rd ed.). Raleigh: CRC Press

Schwartz, S. H. (1999) Visual Perception, 2nd ed. Connecticut: Appleton and Lange

Silverstein, D. A., & Farrell, J. E. (2001). The Relationship Between Image Fidelity and Image Quality. Retrieved Aug. 28, 2001 from the World Wide Web: http://www.best.com/~amnon/Homepage/Research/Papers/FidelityQuality.html

Storer, J. A. (1988). Data Compression: Methods and Theory. Rockville, MD: Computer Science Press

Tada & Kaoru (1988). Image Processing Apparatus. Retrieved May 18, 2001, from USPTO on-line database (US Pat. No. 4866534) on the World Wide Web: http://www.uspto.gov

Taylor, C. C., Allenbach, J. P., & Pizlo, Z. (1998). The Image Fidelity Assessor. Proceedings of the 1998 IS&T Image Processing, Image Quality, and Image Capture Systems Conference, Portland, OR, 17-20 May 1998

Taylor, C. C., Pizlo, Z., & Allenbach, J. P. (1998). Perceptually Relevant Image Fidelity. Proceedings of the 1998 IS&T/SPIE International Symposium on Electronic Imaging Science and Technology. San Jose, CA 24-30 January 1998, vol. 3299

Ulnichey, R. (1987). Digital Halftoning. Cambridge, MA: The MIT Press.

Wandell, B. (1995). Foundations of Vision. Sunderland, MA: Sinauer Press

Watt, A., & Policarp, F. (1998). The Computer Image. Reading, MA: Addison-Wesley

Watt, A., & Watt D. (1997). 3D Computer Graphics. Reading, MA: Addison-Wesley

Zhang, X., & Wandell B. (2000). Color Image Fidelity Metrics Evaluated Using Image Distortion Maps. Retrieved Aug. 27, 2001 from the World Wide Web: http://white.stanford.edu/~brian/scielab/compare/compare.html

Zhou, W., & Bovik, A. C. (2000). A Universal Image Quality Index. Retrieved Oct. 21, 2001 from the World Wide Web: http://anchovy.ece.utexas.edu/zwang/research/quality_index/demo.html (Also published in IEEE Signal Processing Letters).

VITA

Robert N. Villarreal was born on January 13, 1978. He earned his BS in physics from SWT in May 2000 and completed his MS of computer science from SWT in January 2002. Most of his youth was spent in DeWitt County, Texas. However, he currently resides in Houston, Texas, and works for ExxonMobil's Upstream Technical Computing Company as a Staff Systems Technical Analyst. Robert N. is the husband to Rina V. Villarreal and the son of Robert and Cindy Villarreal of Yorktown, Texas.