# RATE CONTROL IN BI-LEVEL IMAGE CODING

by

DAVID ANDREW DOUGLAS TOMPKINS

BSc. (Computer Science), The University of Western Ontario, 1994
BESc. (Electrical Engineering), The University of Western Ontario, 1996

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

Department of Electrical Engineering

We accept this thesis as conforming to the required standard

..........................................................................................

..........................................................................................

..........................................................................................

THE UNIVERSITY OF BRITISH COLUMBIA

August 2000

# ABSTRACT

As we continue to pursue the illusive goal of a paperless society, we are increasingly digitizing our documents, and despite increases in computer speeds and capacities, the requirement for data compression is still paramount. The JBIG2 standard is the latest international bi-level image compression standard, and the first that supports lossy coding. A method is proposed that can achieve rate control while coding bi-level images with JBIG2. For compound images, the image is segmented into text and non-text regions. Non-text regions are lossy coded as generic regions with a bit-flipping approach. The bit-flipping approach has a compression limit, so for higher compression the region is coded as a halftone. For halftone regions, an appropriate grid size is selected and a reduced multi-level image is constructed. The multi-level image is lossy coded with either a bit-flipping approach or with a vector quantization approach, depending on target rate. For text regions, rate control is achieved by adjusting the number of unique symbols in the image. The symbols are a subset of the complete set in the lossless image, and are chosen to minimize the distortion. While the rate control in non-text regions is optimized in the mean squared error sense, the rate control for text regions can be adjusted for any distortion measure. To complete the rate control for a compound image, a weighting is assigned to each region so that the overall distortion is minimized. Results demonstrate that the proposed method is effective, and can produce a JBIG2 bitstream at any target compression rate for any bi-level image.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ACKNOWLEDGEMENTS

I would like to give thanks and a big hug to my friends, family and colleagues who have helped both directly and indirectly to support me throughout this undertaking. There are many important people not listed here, so please forgive any omissions or oversights.

Dr. Faouzi Kossentini has been an advisor, supervisor, manager, critic, boss, investment consultant, and most importantly, a friend. If it wasn't for him, I would never have discovered the joy that is bi-level image compression.

To the fellow members of the Signal Processing and Multimedia Group, both past and present, I have nothing but the utmost respect and admiration. Your advise, inspiration, and jovial ridicule of bi-level image compression has been indispensable. Special kudos go to officemates Shahram, Adriana, Michael, Jaehan, and newcomer Anthony who have put up with my constant barrage of cheesy music, uninformed questions, proofreading requests, and endless procrastination. To Alen and Mike, the 467 team, I equally appreciate your hard work ethic, your intelligence, and your knowledge of geek pop culture. Berna was my travelling Texas buddy, and was an endless source of amusement and nutritional advice. Guy was the first person to show me MAME, and so despite his good will, he would have to fall into the "contributed to procrastination" category. To Ismaeil, Patrick, Kahn, Thomas and George, well, let's be honest -- we didn't do that much together, but we always had fun, and you rounded out SPMG supporting cast nicely.

I'd like to thank the members of the Embedded Signal Processing Laboratory at The University of Texas in Austin, especially Dr. Brian Evans, who invited me. Magesh and Niranjan were excellent hard-working role models that encouraged me to work harder and smarter, while Wade was not.

I would also like to thank the fellow members of the JBIG committee, especially Dr. William Rucklidge and Ono-san, for their technical excellence and leadership, respectively.

To my immediate family, Mom, Dad and Tina, I thank you for your love and support, and acknowledge that I don't reciprocate it enough. To my friends, I must admit that living in Vancouver would have been a lot less fun if it wasn't for Mike and Erik, who enjoyed pointing out at every opportunity that I don't live in the "real world". Also in Vancouver, Suresh turned out to be a great partner in crime, and helped coin the expression that "procrastination loves company". To my engineering buddies, who include Jamie, Jay, Rob, Jon and especially Garb, I guess that living up to your high expectations never seemed as important as it should have been. To my fellow board members -- Chris, Chris, Jerry, Ron and Scott (listed in alphabetical order, so you won't fight) thanks in believing that I am the one who could climb any mountain, or at least pretending that you believe. And to Sarah, the chicken soup for my soul, you really do inspire me to be a better man.

I would also like to thank everyone who was involved in the production of *“The Four Seasons -- The Dance Album”* which, through careful analysis, I have determined to be the audio compact disc that increases my productivity the most.

And finally, to all the people who have been harassing me to finish my thesis (which, unfortunately, are too many to mention), I look forward to new projects that you can harass me to finish.

# CHAPTER 1. INTRODUCTION

This chapter presents an introduction to the material contained in this thesis. In the first section, the motivation and objectives of the research are presented. The second and final section contains an overview of the proposed method, and a brief description of each of the components.

## 1.1 OBJECTIVES

The past two centuries has seen incredible advances in communication technologies. Although sometimes forgotten, the transmission of bi-level images has been an important part of those advancements. The first patent for transmitting images over wire was filed in 1843 by Alexander Bain, and by 1865 images were being transmitted over telegraph lines. A century later, Xerox introduced the magnafax telecopier, which was the first facsimile machine that could be connected to any phone line. The facsimile machine has significantly impacted the business world and society itself, and has even been credited as the technology that brought down communism [Dobb90]. Just ten years ago, the facsimile machine was considered essential equipment for any business. However, with the explosion of the Internet and personal computers, the facsimile machine is no longer considered essential, and is even perceived by some as an outdated technology. But whether the device transmitting the bi-level image is an antique pantelegraph, a facsimile machine, a home computer, or a wireless personal digital assistant, there will always be a need to transmit bi-level images.

The expression *"paperless society"* has been used to describe the environmental utopia where all information is stored on computers, and paper is unnecessary. While it is true that the number of electronic documents is increasing, so is the total number of documents being produced. As a result, the goal of the paperless society has become more of an ideology than an achievable goal. As the paperless society continues to grow, there will be a large increase in the number of electronic documents that have to be stored. While there are numerous formats available to store those electronic documents, a large proportion of those documents are best stored as bi-level images.

For decades, the advance of computers has been following Moore's law, which loosely states that computers have been doubling in speed every 18 months. A similar relationship exists for storage capacity. With this increase in speed and capacity, it might seem that research in data compression is unnecessary. The problem arises from another well-established relationship known as Parkinson's law, which states that "data expands to fill the space available for storage". In other words, while capacities are increasing, so is the quantity of data that must be stored, and so compression becomes even more important.

With the increase in global connectivity and the shift towards a paperless society, the transmission and storage of bi-level images continues to be important, and so is the compression of those images. In some ways, the proliferation of the facsimile machine has held back advancements in bi-level image compression, since upgrading millions of established machines has not been a viable business proposition. However, with the evolution of the Internet and the communications world, there are new opportunities for advanced bi-level image compression to be used in a variety of applications.

The latest standard from the joint bi-level image experts group is the JBIG2 standard [T88], which is poised to become the compression method of choice for the next generation of bi-level image applications. JBIG2 is also the first standard that allows for lossy compression, which is an important feature for applications that require very high levels of compression. With lossy compression, choices must be made regarding the desired level of compression, or the desired level of quality. The JBIG2 standard defines the behaviour for decoding an image, but does not specify how an encoder should behave or how to perform lossy coding.

Ideally, a lossy compression method should be able to compress an image to any size, allowing the application to choose the appropriate level of compression. For example, an application may need to compress 30,000 pages so that they can all be contained on a single CD-ROM, or it may need to transmit one page per second over a low bandwidth channel, or it may wish to have several different levels of compression available for document browsing and downloading. Academically, the ability to achieve any compression level is also desirable.  If two different lossy compression methods are to be compared, then they should be compared at the same rate, which can be difficult if there is no mechanism available to control the rate.

While some work has been proposed to achieve lossy compression for certain types of bi-level images [Prat80,Koss96,Howa96,Mart99], there has been no method proposed to accommodate all types of images.  Furthermore, most lossy compression methods are designed to achieve a certain level of quality, and are not designed to achieve a target rate.  The proposed method accommodates both of those design considerations, and can compress any bi-level image to any rate, all within the framework of JBIG2.  An overview of the proposed method is given in the following section.

## 1.2 OVERVIEW

The proposed method for compressing bi-level images has several different components. Figure 1-1 illustrates how each of the different components contribute towards the overall goal.  First, the bi-level image is segmented into text and non-text region images.  If there is more then one region, a compound rate control method determines the target rate for each region image.  Next, depending on the region type, either a text or a generic rate control method is applied to each image.  If the target rate cannot be achieved, then a halftone rate control method is applied.  Depending upon the compound rate control method and the results obtained, the target rates may be recalculated, and the process can reiterate.

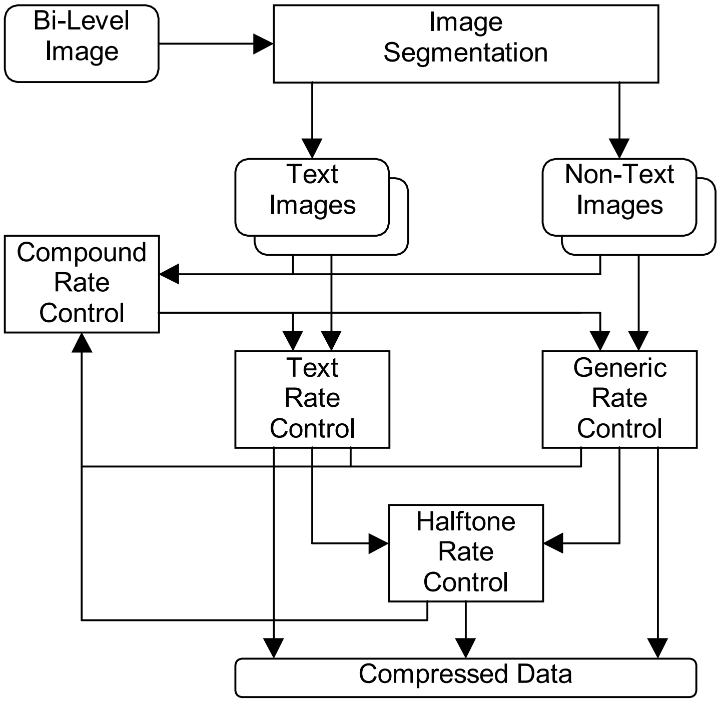


Figure 1-1 Overview of Proposed Method

In Chapter 2, rudimentary background material is presented. In the first section, there is an introduction to bi-level images and different types of bi-level images. The second section contains a general overview of image compression, and some bi-level image compression standards. The third and final section describes the JBIG2 standard.

In Chapter 3, mechanisms for measuring distortion in bi-level images are proposed. The first section outlines the difficulties encountered while measuring distortion. The next two sections describe distortion measures that can be used for textual and halftone images, respectively. The last section addresses the challenge of selecting an appropriate distortion measure.

In Chapter 4, a method is proposed for separating text from non-text regions. The first section reviews existing methods and addresses the requirements of a JBIG2 segmentation method. The second section describes the proposed method. The third and final section presents the results.

In Chapter 5, a method for generic rate control is presented. In the first section, some background on the topic is provided. In the second section, existing implementations are discussed. In the third section, improvements to the existing method are put forward. In the fourth and final section, the results are presented.

In Chapter 6, a method for refinement rate control is presented. While refinement control is not specifically identified as a component in Figure 1-1, it can used to complement other rate control methods. In the first section, some background on the topic is provided. The second section describes the proposed method. In the third and final section, the results are presented.

In Chapter 7, a method is proposed for controlling the rate of a text region. In the first section, an overview of the method is provided, which consists of two stages. In the second section, the first stage, which determines the image that will produce the smallest rate for a given distortion level is described. The third section describes how the target rate is achieved in the second stage. The fourth and fifth sections describe some implementation details and possible enhancements, respectively. In the sixth and final section, the results are presented.

In Chapter 8, a method is proposed to control the rate of a halftone region. In the first section, the general procedures for JBIG2 halftone coding are explained. In the next section, an overview of the proposed method is presented. In the following two sections, methods for higher and lower bitrates are explained. In the last section, results are presented.

In Chapter 9, a method for compressing compound documents is presented. In the first section, the details of the method are presented. In the second and final section, results are presented.

# CHAPTER 2. BACKGROUND

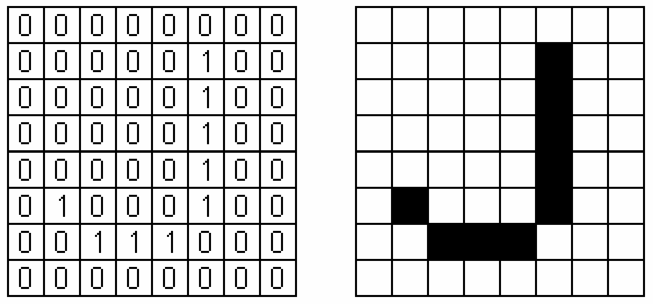
In this chapter, rudimentary background material is presented. In the first section, there is an introduction to bi-level images and different types of bi-level images. The second section contains a general overview of image compression, and some bi-level image compression standards. Finally, this chapter concludes with a section on the JBIG2 standard.

## 2.1 BI-LEVEL IMAGES

Humans have a very sophisticated system for image processing. They can detect a wide variety of colours and can focus with incredibly high resolution. These two features of the human visual system (HVS) are difficult to duplicate with a computer or similar electronic device. A digital image must have some physical restrictions. Specifically, the size and the resolution of the image must be fixed, and the number of colours must be restricted. In most circumstances, the image is confined to a rectangular shape, and is represented by a two-dimensional array of values. Each element of the array corresponds to a square picture element, or *pixel*. The size of each pixel is determined by the resolution of the image, and the value associated with each pixel determines its' colour.

The resolution of an image is often measured in dots per inch (dpi) and will depend on the devices that read or display the image. For example, most computer monitors display resolutions lower than 100 dpi, while laser printers can achieve resolutions greater than 1600 dpi. The number of different colours that a pixel can support is measured by the number of bits required to store the colour value for each pixel. For example, a pixel in a photographic quality 24-bit image can have values between 0 and $2^{24}$-1, or 16,777,216 different colours. A grayscale image is typically 8-bit, and can represent 256 different shades of gray.

Bi-level images are one-bit images: requiring only one bit of colour information for each pixel. This bit of information can represent two colours, which are usually black and white, and are represented by "1" and "0", respectively. Figure 2-1 illustrates a simple bi-level image.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 2-1 Bi-Level Image Representation

An image that has more than two colours is referred to as a *multi-level* image. Although the expression "black and white image" implies a bi-level image, it is often incorrectly applied to grayscale images, and should be avoided. Figure 2-2 illustrates the difference between a multi-level image and a bi-level image.

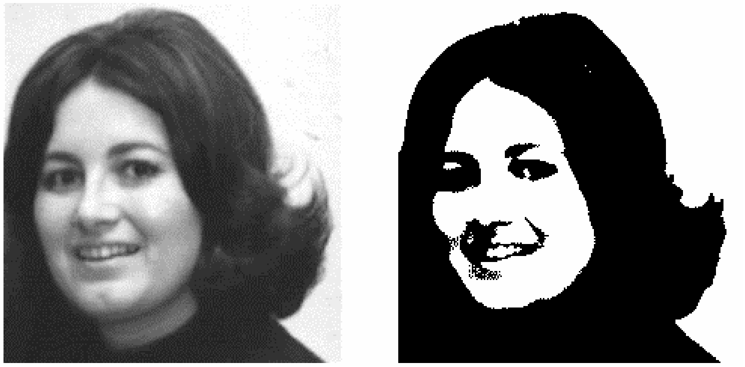
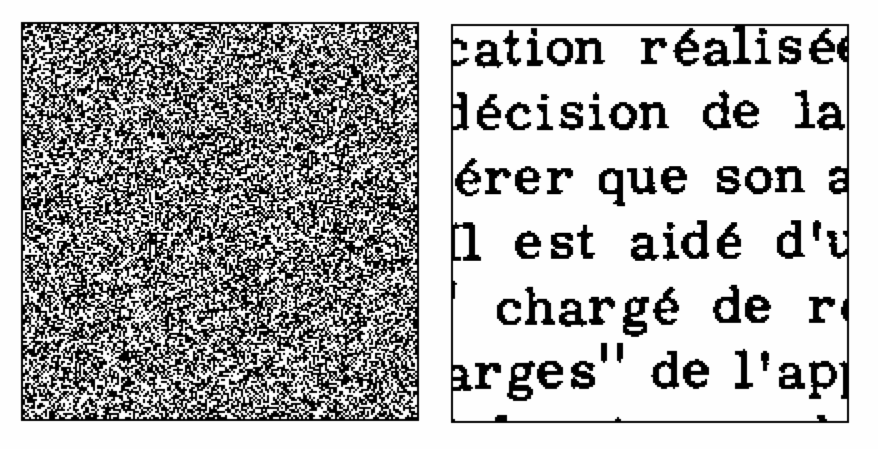


Figure 2-2 A Multi-Level Image and a Bi-Level Image

## 2.2 TYPES OF BI-LEVEL IMAGES

The majority of printers, photocopiers, facsimile machines and print media are bi-level, and most bi-level devices will encounter these types of images. Consider the two images in Figure 2-3. Both images are the same size, and are valid bi-level images, but the textual image is more likely to be encountered by a bi-level device. If the images are stored uncompressed, then the source and appearance of the images is not important. However, if any data compression is used, then the source and type of images encountered becomes very important. In practice, only a few categories of bi-level images types are encountered.



cation réalisé
décision de la
érer que son a
l est aidé d'u
chargé de r
arges" de l'ap

Figure 2-3  Two Bi-Level images: Noise (Unlikely) and Text (Likely)

### 2.2.1 TEXT IMAGES

As the name implies, text images contain textual data. Text images are well organized and structured, and are comprised of numerous characters. The characters are all approximately the same size, and depending on the native language, are organized in either rows or columns. Many of the characters will repeat on a page, occurring in several different locations. Text images are the most common form of bi-level image types. Figure 2-3 contains a sample text image.

### 2.2.2 HALFTONE IMAGES

Halftones are used to display multi-level images as bi-level image. Halftones have been used by the newspaper print media for decades to give the illusion of grayscale image quality [Fole82,Ulic87]. By placing black dots in a variety of patterns and arrangements, and viewing the image at a suitable distance, the eyes can be deceived into accepting the image as a grayscale. There are two general types of halftones: ordered and stochastic.

Ordered halftones sub-divide the image into a grid, and then fill each element of the grid with a pattern. The pattern is usually either a clustered-dot pattern, or a dispersed-dot pattern, depending on the media and the desired effect. In Figure 2-4, the first image is a clustered-dot, and the second image is a dispersed-dot.

Unlike ordered halftones, stochastic halftones [Floy76] do not use a grid to represent the image. Instead, dots are arranged so that the average gray intensity is preserved in the image, and errors are seen as high-frequency noise. This technique is often referred to as "error-diffusion", and is illustrated in the third image of Figure 2-4.

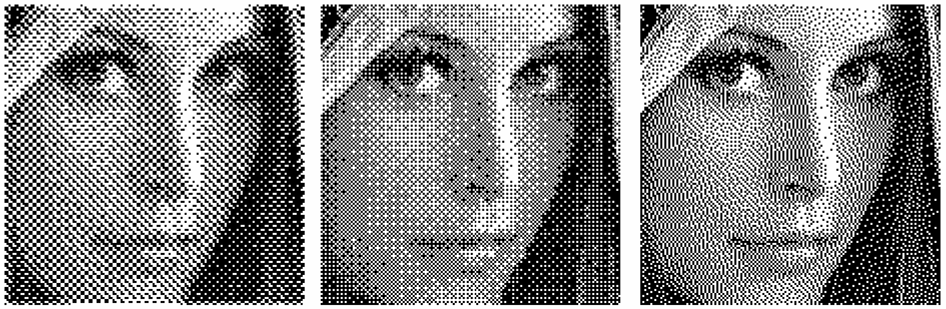


Figure 2-4 Halftone Images: Clustered Dot, Dispersed Dot, and Stochastic

### 2.2.3 LINE-ART IMAGES

All of the bi-level images that are not halftones, and do not exhibit the characteristics of a text image can be loosely categorized as a line-art. Line-art includes charts, graphs, figures, diagrams, handwriting and sketches. There is usually some basic structure in a line-art image, containing at the very least lines. Figure 2-5 shows some examples of line-art.

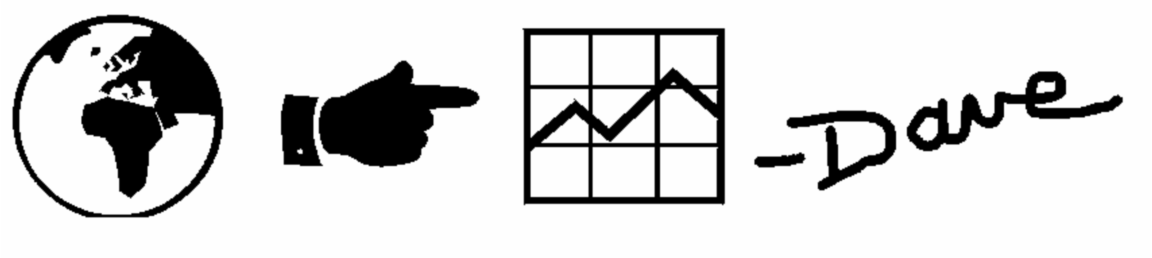


Figure 2-5 Line-Art Images

### 2.2.4 COMPOUND IMAGES

Often, an image will contain some regions that are text, some regions that are line-art, and perhaps some halftone regions. These images that contain more than one image type are known as compound images.

### 2.2.5 SCANNED AND GENERATED IMAGES

Each of the aforementioned image types, can be further subdivided into scanned and generated images. Scanned images are produced from optical devices that scan a physical medium to construct a digital image. Generated images are naturally digital, most likely produced through some graphical or desktop publishing software. Scanned images naturally have more noise and small imperfections that are introduced in the scanning process, while generated images appear much cleaner.

## 2.3 IMAGE COMPRESSION

Data compression is the process of representing a set of data with a smaller set of data. In other words, if there is data which requires N bits to represent it, and it is represented with fewer than N bits, then there is data compression. Data compression is extremely important in applications where the data must be stored, or transmitted over a medium. Figure 2-6 illustrates the basic components of a compression system. The input data of size N is compressed (or encoded) to become much smaller. This smaller set of data is called the *bitstream*. The bitstream can either be transmitted to another location or be stored for later retrieval. To be used, this compressed data must be expanded (or decoded) to the original size for the output device.

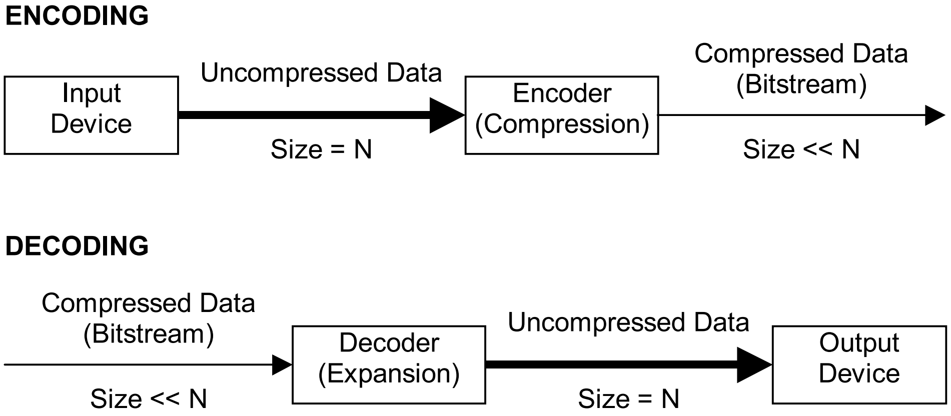
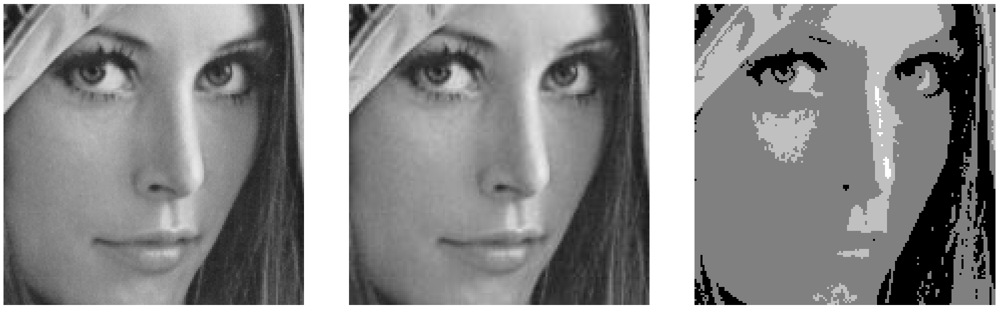


Figure 2-6 Basic Elements of a Data Compression System

An uncompressed bi-level image requires one bit of information per pixel, so with data compression, bi-level images are represented with less than one bit per pixel. A typical bi-level application uses images that are very large, making compression very important. For example, if a laser printer prints an 8.5” x 11” page at 600 dpi, there is over 4MB of data. Transmitting 4MB of data across a bus or a network without compression can be very slow and expensive.

In order for most data compression systems to work, some assumptions have to be made about the nature of the data being compressed [Sayo96]. With bi-level image compression, the assumption is made that the image has some structure, and more specifically, that it is something that would be normally produced from a printer, facsimile machine, or a similar device. It is because of this structure and organization of the image that compression can be achieved. In the images of Figure 2-3, the textual image is clearly more organized than the noisy image, and is a much more viable candidate for compression. To design a compression scheme that could achieve equally high compression on both images would be nearly impossible, and foolhardy, as the textual image is far more likely to be encountered by a bi-level processing device.

There are two fundamental types of data compression. Lossless compression is when the original set of data can be perfectly reconstructed from the bitstream. Alternatively, lossy compression is when the reconstructed data is not the same. The difference between the lossy reconstructed data and the original data is called *distortion*. Figure 2-7 illustrates the lossy compression of a multi-level image. The first lossy re-constructed image shows very little visible distortion, and can be considered perceptually lossless. However, the distortion in the second lossy image is clearly visible. Most visible distortions in a lossy image are referred to as *artifacts*.



Original Image

Lossy Image
Perceptually Lossless
(Low Distortion)

Lossy Image
(High Distortion)

Figure 2-7  Lossy Image Compression

With lossy data compression, any amount of compression can be achieved, with a corresponding amount of distortion.  This tradeoff between the compression rate and the amount of distortion is illustrated via the rate-distortion curve of Figure 2-8.  This general relationship between distortion and rate holds for most lossy compression systems.

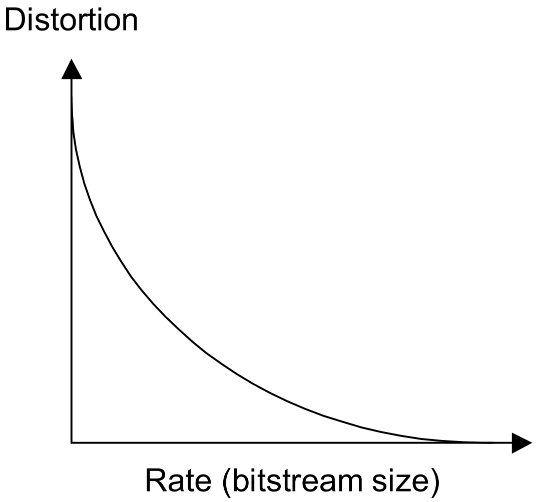


Figure 2-8  A Simple Rate-Distortion Curve

## 2.4 IMAGE COMPRESSION STANDARDS

In order for compression to work properly, the decoding device must know how to decompress the bitstream produced from the encoding device. If the encoding and decoding device are the same, then the system is simple. However, if the encoder and decoder are two completely separate devices, this problem becomes more difficult, especially if the two devices are from separate manufactures. Most people expect that two facsimile machines will be able to exchange data, even if they are from different companies. The reason facsimile machines are successful in communicating their information is because there is an international standard which determines their technical behaviour.

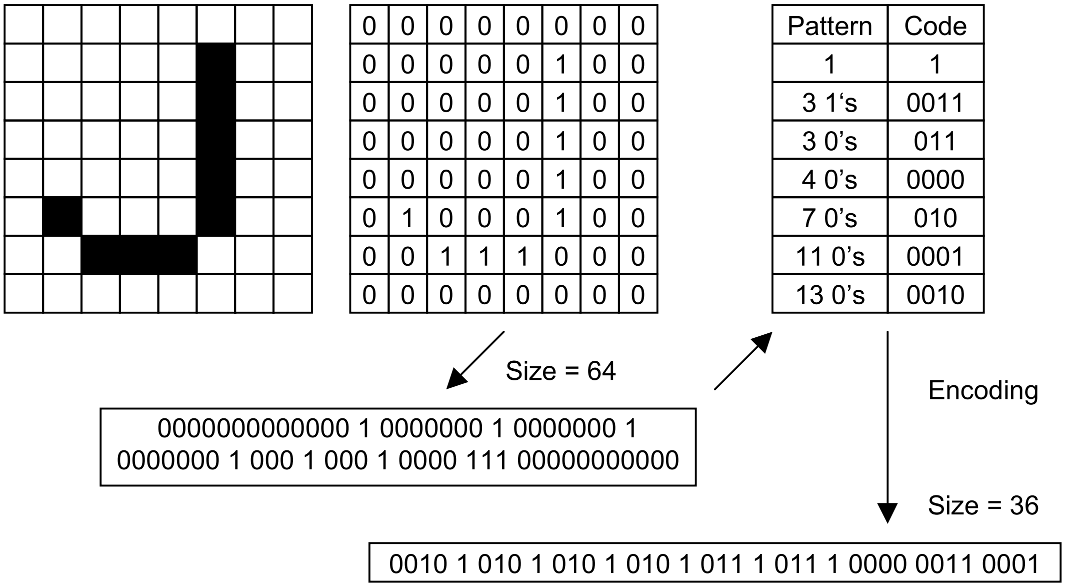
The International Organization for Standardization (ISO) is a non-governmental body that exists to establish and issue standards in a wide variety of fields and disciplines. The International Telecommunication Union - Telecommunication Standardization Sector (ITU-T) has a narrower scope, and works with governments and the private sector to coordinate and recommend standards. Together, the two organizations have developed standards that network the world.

Conducting research within a standard has several advantages. Academically, any research done with a standard becomes more useful, as the results can be easily interpreted and duplicated. By using the framework of standard, it can become much easier to focus on the core algorithmic issues, instead of the design and implementation details.

The proposed method is developed within the framework of the Joint Bi-Level Image Experts Group JBIG2 standard. This is a new international standard, and is the first bi-level image compression format sophisticated enough to support all of the research. The JBIG2 standard is explained in Section 2.5. For background information, some of the other bi-level image compression techniques and standards are explained first.

### 2.4.1 RUN LENGTH ENCODING (T.4 & T.6)

The vast majority of the facsimiles sent around the world today are sent with Huffman based run length encoding [Huff51,Arps94]. The basic concept of run length encoding is very simple: instead of sending every 1 and 0, send a count of sequential 1's (or a *run* of 1's) followed by a run of 0's, and so on. Figure 2-9 illustrates a simple run length coding system which was developed specifically for the sample image. Starting from the upper left corner and moving left to right for each row (called raster scan order) the image is defined as a sequence of runs of 1's and 0's. At the start, there is a run of 13 0's, so instead of transmitting 13 '0' bits, the code for 13 0's is determined from the table (0010) and transmitted, which in this example saves 7 bits. The rest of the runs of 1's and 0's are encoded in the same manner, compressing the total image 1.8:1. The table used in figure 2.9 was specifically designed for this example. In the standards, the tables were developed by analyzing numerous test images.



|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Pattern | Code |
|---|---|
| 1 | 1 |
| 3 1‘s | 0011 |
| 3 0’s | 011 |
| 4 0’s | 0000 |
| 7 0’s | 010 |
| 11 0’s | 0001 |
| 13 0’s | 0010 |

0000000000000 1 0000000 1 0000000 1
0000000 1 000 1 000 1 0000 111 00000000000

0010 1 010 1 010 1 010 1 011 1 011 1 0000 0011 0001

Figure 2-9  A simple Run Length Encoded Sequence

ITU-T Recommendation T.4 [T4] is the original Group 3 facsimile standard and defines a mode called Modified Huffman (MH) which is very similar to the method used in Figure 2-9.  The recommendation also supports a mode called Modified Read (MR) which is slightly more sophisticated.  With the MR mode, the run lengths are not given directly, but rather as changes between the current line and the previous line, as illustrated in Figure 2-10.  This is a simplified example, and does not demonstrate the full scope of the MR mode, but is sufficient enough to demonstrate how the MR mode is implemented.

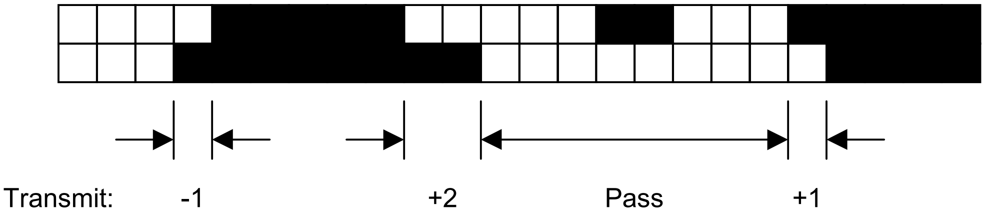


Figure 2-10  Simple Modified Read (MR) Coding

ITU-T Recommendation T.6 [T6] uses a modified version of the MR mode, creating the awkwardly named Modified Modified Read (MMR) mode. The MMR mode is designed for robust error-free environments, so many of the control and error protection codes in MR are not included, making it more efficient. The JBIG2 standard supports the MMR mode as an option for many coding procedures.

Run length encoding is popular with facsimile manufactures because it is simple, and can be executed very quickly. The compression achieved with run length coding is good, but much better techniques exist. One of the problems with run length coding is its' poor performance on halftoned images. Run length coding can actually expand halftones, because they have very short run lengths.

### 2.4.2 JBIG1

ITU-T Recommendation T.82 [T82] is also known as ISO/IEC 11544 and JBIG, and since the development of JBIG2, is most often referred to as JBIG1. JBIG1 is the first bi-level standard that supports a progressive transmission mode [Fowl95], which is mostly used in software applications. In the progressive mode, a low-resolution image is initially coded, followed by images with progressively increasing resolutions. This research does not use a progressive transmission mode, so the non-progressive mode is of more interest.

JBIG1 uses a form of data compression known as arithmetic coding. Arithmetic Coding is described in Appendix A. Specifically, JBIG1 uses the QM binary arithmetic coder

[Mitc88, Penn88a]. The QM coder is similar to the MQ coder, which is described in detail in Section 2.5.1.

As the very start of the JBIG1 coding procedure, the pixel values of black and white are equally probable. As coding proceeds, the statistics from the image already coded are used to determine more accurate probabilities. Figure 2-11 illustrates the simplified JBIG1 model that is used for determining probabilities. If the coloured pixel is the next pixel to be coded, the surrounding 10 pixels are known as the *context* for that pixel. A separate set of statistics is maintained for each unique context. After each pixel is coded, the statistics are updated for its' context, to improve the prediction for the next pixel that shares the same context. The context model is very effective at capturing the patterns and structure of an image. A JBIG1 coder adapts well to most image types, and can achieve much higher compression rates than run length coders.

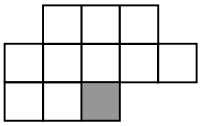


Figure 2-11 A Simplified JBIG1 Context

There are some drawbacks with the JBIG1 standard. Storing the context statistics increases the memory requirements for a coder. Also, arithmetic coding is much slower and more complicated than run length coding. An additional concern with JBIG1 is the intellectual property it contains. Because of patents held by Lucent, Mitsubishi and IBM, any implementation of JBIG1 must be registered with these companies and a licensing fee must be paid. Unfortunately, it is this intellectual property issue which has stunted the growth of JBIG1, and is why it has not significantly penetrated the North American and Internet market. Fortunately, the JBIG committee has learned from JBIG1, and these difficulties have been avoided in JBIG2.

## 2.5 **THE JBIG2 STANDARD**

The JBIG2 standard [T88,Howa99] does not define how a JBIG2 encoder shall operate, but rather how a compliant JBIG2 decoder shall interpret a JBIG2 bitstream. This subtle and important distinction creates a significant difference in complexity between encoders and decoders. For any image, there are numerous different methods of encoding it, and therefore numerous different JBIG2 bitstreams that can be produced. However, for every JBIG2 bitstream, there is only one resulting image that can be decoded. This flexibility in encoder design allows for encoders with varying levels of sophistication, complexity, speed and compression performance. However, all JBIG2 encoders must produce bitstreams that conform to the JBIG2 bitstream syntax, so their behaviour is implicitly limited by the structure and details of that syntax.

One of the key philosophies behind the development of the JBIG2 standard was to provide two principal coding methods: arithmetic coding & Huffman coding. For most of the JBIG2 segments, the data can be coded with either method. Although there are a few exceptions, arithmetic coding has better compression performance than Huffman coding, and so the focus will be placed on arithmetic coding. The arithmetic coder used in the JBIG2 standard is the MQ binary arithmetic coder [Slat98].

### 2.5.1 THE MQ BINARY ARITHMETIC CODER

A generic arithmetic coder is described in Appendix A. The ways in which the MQ binary arithmetic coder is different than a generic coder will now be explained.

The notation used within the MQ coder is slightly different than the notation used in Appendix A. Instead of updating the Interval ($R$) after each symbol, the MQ coder updates the lower interval value ($C$=$R_L$) and the size of the interval ($A$= $R_H$ -$R_L$). This reduces the complexity of the algorithm.

The MQ coder is a binary coder, and as such uses a binary alphabet. The alphabet is not {0,1} as one might suspect, but rather {LPS,MPS} where LPS and MPS are the Least Probable Symbol(s) and Most Probable Symbol(s), respectively. The most probable symbol (1 or 0) is determined for each data element. Because the system is binary, only one of the probabilities has to be determined. Specifically, the probability of the LPS is always determined. The symbol $P_{LPS}$ is used to represent the probability of the LPS, and the symbols $n_{MPS}$ and $n_{LPS}$ are used to count the number of times the MPS and LPS occur.

To achieve the maximum compression, an ideal arithmetic coder must have infinite precision. In practice, the MQ coder uses 16-bit precision to represent the size of the interval ($A$). For a large alphabet, 16-bit precision would not be sufficient, but for a binary system, using 16 bits is acceptable. The lower interval limit ($C$) requires a 32-bit value, but only 16 bits are used for calculations. To avoid the problems that can result from using finite registers, certain techniques can be used to minimize the loss in coding efficiency. Specifically, whenever the size of the interval is reduced to less than half of the largest possible interval, the interval size is doubled. This procedure is called *renormalization*. Whenever a renormalization occurs, the lower interval value is doubled, or shifted by one, which corresponds to one bit of the coded data. While encoding, the actual bit of data might not be known at the time of the shift, but with buffering and special mechanisms in place to bit stuff carry forward values, the system has been implemented with a minimal amount of complexity and coding loss.

While most arithmetic coding systems use the full interval [0,1) the MQ coder restricts the size of the interval to [0.75,1.5). Although this is not an intuitive implementation, it allows for a simplification which significantly reduces the complexity of the system. Because the size of the interval is in the range [0.75,1.5) it can be approximated with the value of 1. For example, the following simplifications can be made:

$$C_{k+1} = C_k + A_k \cdot P_{LPS} \cong C_k + P_{LPS}$$
$$A_{k+1} = A \cdot P_{LPS} \cong P_{LPS}$$

This technique improves the speed, but reduces the accuracy and coding efficiency of the MQ coder. This simplification can also cause some problems with the probability calculations, resulting in situations where the probability for the LPS will be higher than the MPS. An additional check is required to avoid this error and swap the LPS and MPS probabilities.

As discussed in Appendix A, arithmetic coders need a method to predict the value of the next element of data. If a uniform distribution of MPS and LPS is assumed, then the Bayesian estimate of $P_{LPS}$ would be given by the following equation [Grim92]:

$$P_{LPS} = \frac{n_{LPS} + 1}{n_{LPS} + n_{MPS} + 2}.$$

In the MQ coder, the estimate is modified to become

$$P_{LPS} = \frac{n_{LPS} + \delta}{n_{LPS} + n_{MPS} + 2\delta},$$

where the value of $\delta$ skews the value of $P_{LPS}$ towards 0 as $\delta$ approaches 0. In the Q Coders, the value of $\delta$ is 0.45, which was chosen experimentally [Penn88b].

The estimate of $P_{LPS}$ works very well if the data is stationary. The data is considered stationary if the 1's and 0's are uniformly distributed throughout the data. However, in many applications for which the MQ coder was designed, the data is not stationary and can change rapidly. In these situations, the estimate becomes detrimental because it can not change quickly enough. To model a probability estimate that works well for non-stationary data and is still accurate for stationary data, the value of $n_{LPS}$ is *clamped*. To clamp the value of $n_{LPS}$, whenever it reaches a maximum value, the values of $n_{LPS}$ and $m_{LPS}$ are scaled down by the same value.

The above discussion concerning the calculation of $P_{LPS}$ implies that the MQ coder maintains accurate counts of $n_{LPS}$ and $n_{MPS}$, which is not the case. Instead of collecting data statistics and accurately calculating the probability for each data element, a state table is used. There are 47 different states in the table, with each state having a corresponding $P_{LPS}$ ranging from 0.000023 to 0.503937. The coder is always in one of the 47 states, and changes states conditionally upon whether the next symbol is the LPS or MPS. To illustrate how the MQ state table transitions occur, three of the 47 states are illustrated in Figure 2-12.

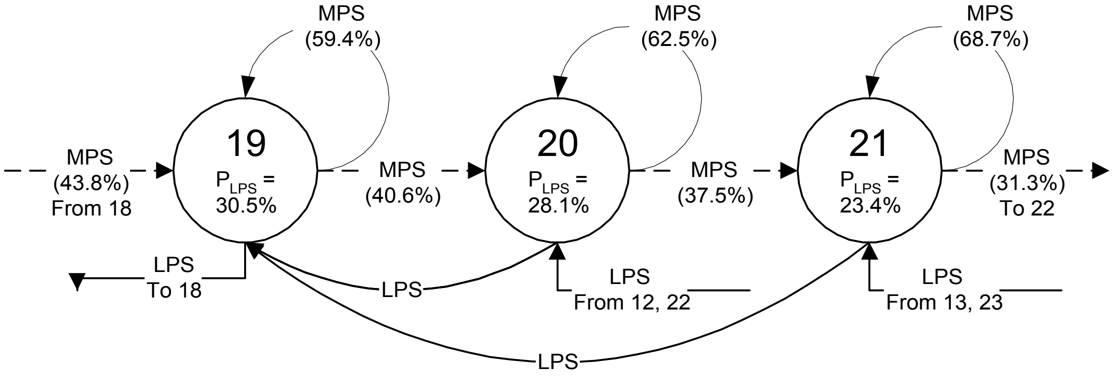


Figure 2-12 Partial State Diagram for the MQ Coder

As illustrated above, when the coder is in state 20 and an LPS occurs, the coder switches to state 19. The coder also switches to state 19 when an LPS occurs in state 21, which is a much larger change in value of $P_{LPS}$. This transition from state 21 to 19 is an example of how the $n_{LPS}$ value is periodically clamped to allow for non-stationary data. While the state diagram paths in Figure 2-12 are simple for LPS transitions, there are two different paths shown for MPS transitions. In the MQ coder, the state is only changed for the MPS if a normalization is required. For the MQ coder, the probability of a renormalization is equal to $P_{LPS}$ / 0.75. For states with low values of $P_{LPS}$, it could take thousands of MPS before a transition occurs, and the MQ state table has been designed accordingly. The fixed probability state table greatly reduces the run-time complexity of the $P_{LPS}$ calculations, and of the entire MQ coder. However, the probability approximations are made at the cost of some coding efficiency, and can make an analysis of the data very complicated.

Although the MQ coder is designed to code just single bits, it is also used to code integers, which are essentially sequences of bits. A table is used to determine the sequence of bits for a specific integer. Small integers are assigned the shortest sequences. While coding the sequence, a context is constructed with the previous bits, in a manner similar to the context construction used in generic coding, which is explained in detail in Section 2.5.3.

### 2.5.2 JBIG2 SEGMENTS

A JBIG2 bitstream is a collection of ordered JBIG2 segments. Each segment contains a segment header part, and most segments contain a data part. In each segment header there is a type identifier that specifies how the data part is to be interpreted. Some of the segment types are very straightforward, such as an end of page segment, and do not require a data part. However, most segments contain a large amount of data. Within each data part, there is a data header, which contains more specific details about how the data was coded. By examining the segment header and the data header, the data can be interpreted and processed by the decoder. For some segments, to properly process the data, the decoder must have already processed the data in some of the previous segments. In these circumstances, it is said that a segment *refers* to another segment.

All of the JBIG2 segments can be loosely classified into three categories: control segments, support segments, and region segments. A control segment is used to provide overhead information to the decoder such as the page dimensions, and end of page markers. A support segment contains data that will be used by region segments, but is not a region segment itself. There are two important support segments that will be examined: dictionary segments and pattern segments. A region segment contains the necessary data to produce an image that will appear on a page. The region segment may correspond to the entire page, or specify a specific rectangular region of the page. A single page may contain several region segments, which are allowed to overlap. There are four different types of region segments: generic regions, refinement regions, text regions, and halftone regions. There are six major segment types that will be examined more closely.

### 2.5.3 GENERIC REGIONS

Generic regions were given the name because they are useful for all types of images. With the arithmetic coding option, generic regions are coded in a manner very similar to JBIG1. There are four context templates that can be used with Generic regions, and they are illustrated in Figure 2-13.

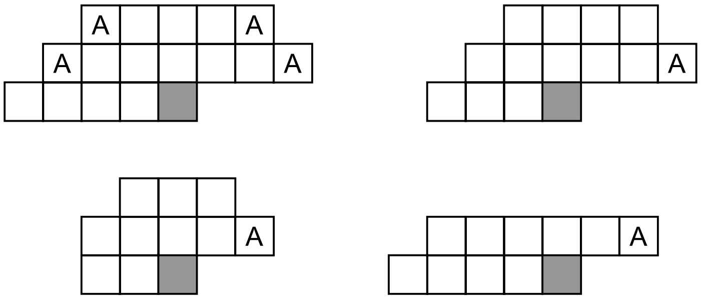


Figure 2-13 The Four JBIG2 Context Templates

In Figure 2-13, the coloured pixel is the next pixel to be coded, and the pixels with an ‘A’ are known as adaptive pixels, which will be discussed shortly. The different templates have specific advantages and properties. In general, the more pixels in a template, the better the compression will be, unless the image is small, in which case the medium sized template will most likely perform the best. The two-line template is much faster than the other templates, because less memory has to be accessed to generate the context. One of the primary concerns for hardware manufactures is the memory requirement for each template. In JBIG2, one byte of memory is required for each possible context, so for a template of size N, $2^N$ bytes of context memory are required.

The context memory is usually placed in the fastest RAM available, as it is accessed frequently while coding a generic region. The context memory is an array of bytes, with each byte representing a possible state in the MQ coder. The construction of a context ID is illustrated in Figure 2-14.

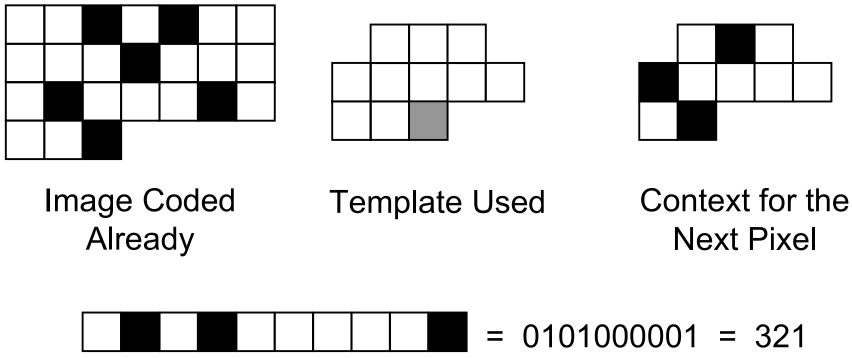


Figure 2-14 Generating a Context ID

From this example, it can be seen why $2^N$ different contexts are required, one for each possible arrangement of pixels in the context. Practically, each context only requires 7 bits of memory, but one byte is usually used. There is 1 bit required to store the current MPS, and 6 bits required to store which of the 47 states of the MQ coder that context is in.

A generic region segment is coded as follows. At the start, all of the contexts are initialized to an MPS value of 0 (white) and to state 0 (from the MQ coder state table). For each pixel in the region, the context ID is generated, the state and MPS for that ID are retrieved, and then the actual colour of the next pixel is coded with MQ coder. The MQ coder then updates the state and MPS information for that context ID and proceeds to the next pixel.

Adaptive pixels, as seen in Figure 2-13, can be moved from their nominal position to help detect patterns in the image. Figure 2-15 illustrates the possible advantages of moving one of the adaptive pixels.

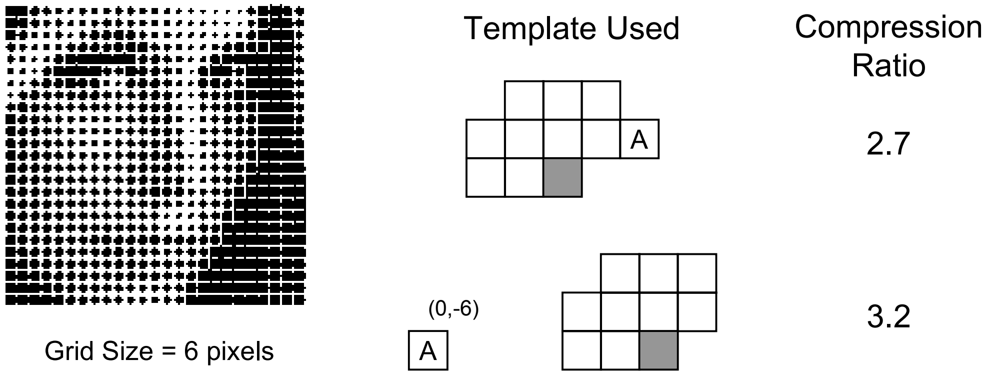


Figure 2-15 The Advantage of using Adaptive Pixels

In this example, the image to be coded has been halftoned with a grid size of six. When the adaptive pixel is relocated to a position that corresponds to the periodicity of the halftone, the context can now more accurately predict the colour of the next pixel, which is demonstrated in this example with an 18% increase in compression.

### 2.5.4 REFINEMENT REGIONS

Refinement coding, the method used in Refinement regions, is a new coding method. With refinement coding, the source image is *refined* to become the destination image. There must already exist a source image to be refined, which may be the entire page, a specific region of the page, or another region altogether. Although refinement coding can be applied to any source image, refinement coding works the best when there are only subtle differences between the source and destination images, such as the images in Figure 2-16.

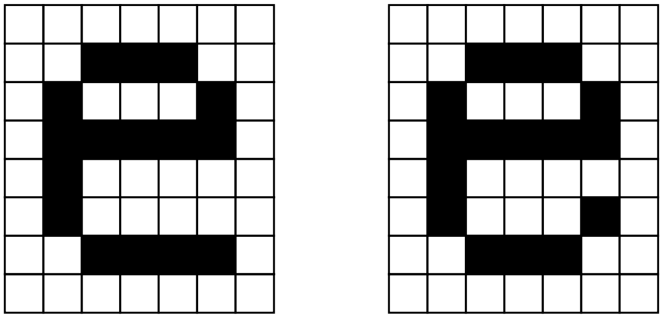


Figure 2-16 Two Suitable Candidates for Refinement Coding

If the first image of Figure 2-16 has already been coded, and the second image is to be coded, then there would be an advantage to using the information from the first image to code the second image. Theoretically, this is performed automatically by the MQ coder in generic coding, because the statistics from the first image have been incorporated into the context probability model. However, with refinement coding, the information from the first image can be used more directly.

The mechanism for refinement coding is nearly identical to the mechanism for generic coding. The significant difference between the two methods is the context. With refinement coding, the context uses information from both the source and the destination image, while generic coding only uses information from the destination image. This can be seen in the refinement context in Figure 2-17.

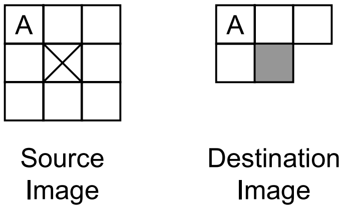


Figure 2-17 Refinement Coding Context

In the above figure, the coloured pixel represents the next pixel to be coded, the 'X' pixel represents the corresponding pixel in the source image, and the 'A' pixels are adaptive pixels, as explained in Section 2.5.3.

Refinement coding was designed to take advantage of the small differences between letters in text documents, especially scanned documents. Small spatial distortions such as the ones in Figure 2.14 are often introduced in the scanning process.

### 2.5.5 SYMBOL DICTIONARIES

A symbol dictionary segment is a support segment that is referred to by text regions or other symbol dictionaries. A symbol dictionary segment produces a collection of numbered symbols, where each symbol is a separate image. Although the symbols in a symbol dictionary could be anything, in practice they are usually small, and correspond to the individual characters of a text document. There are four basic types of symbol dictionaries: new dictionaries, refinement dictionaries, aggregate dictionaries, and administrative dictionaries. A symbol dictionary is not assigned to a specific dictionary type, and may in fact be a combination of two or more of these types.

The purpose of a new dictionary is to create symbols for the first time. For the first symbol, the height and width are coded with the MQ integer coder, followed by the symbol itself coded with the generic coder. For each subsequent symbol, the difference in height and width from the previous symbol is coded, followed by a generic region for the symbol. This procedure is repeated until all of the symbols have been coded.

A refinement dictionary is used to produce a new dictionary, based upon symbols that already exist in another dictionary. Naturally, a refinement dictionary uses refinement coding, and the source symbol may be from a separate dictionary, or be a previous symbol from the current dictionary. Some symbols may in fact be several refinements deep, being refinements upon refinements, and so on. The structure of refinement dictionaries is slightly different than the one for new dictionaries, as the additional information required to identify the source symbol must also be included.

An aggregate dictionary is used to combine existing symbols into new symbols. For example, three symbols that represent the images of “t”, “h” and “e” could be combined into a fourth symbol, representing the image “the”. Aggregate symbols are actually small text regions, which will be discussed in Section 2.5.6.

An administrative dictionary can be used to prune dictionaries, or to combine two or more dictionaries. For example, consider a document with two pages. A dictionary could be produced for the first page, and a second dictionary could be produced for the new symbols located in the second page. An administrative dictionary could take a subset of the symbols from the first dictionary and combine them with the symbols in the second dictionary to produce a third dictionary. In practice, administrative dictionaries are combined with other types of dictionaries, and in the previous example the second dictionary could import symbols from the first dictionary as well as produce the new symbols for the second dictionary.

### 2.5.6 TEXT REGIONS

The purpose of a text region is to take symbols from one or more symbol dictionaries and place them in a region, which is usually part of the page. Essentially, the text region contains a set of symbol numbers along with the necessary co-ordinate information to place the symbols in the region. The actual syntax of the text region is very flexible, and can organize the text in any order, and may reference symbols by any of their four corners. Traditionally, the symbols are ordered from left to right, row by row, and are referenced by their bottom left hand corner. Instead of specifying the co-ordinate for each symbol, the distance from the previous symbol is coded. The symbol numbers and the relative co-ordinate information are coded with the MQ integer coder. Text regions can also include refinement information, if the symbol for the region is not exactly identical to the image from the dictionary.

### 2.5.7 PATTERN DICTIONARIES

A pattern dictionary segment is very similar to a symbol dictionary, in that it contains pattern images to be used by a halftone region segment. Pattern dictionaries are actually much simpler than symbol dictionaries, and contain a specific number of patterns, with each pattern being the same size. Pattern dictionaries are traditionally halftone patterns, ordered by their intensity. The pattern dictionaries are generically coded as one large horizontal image, requiring the individual patterns to be extracted from the image. An example of a halftone pattern dictionary region is shown in Figure 2-18:

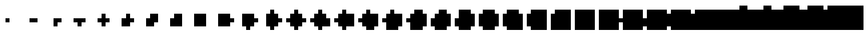
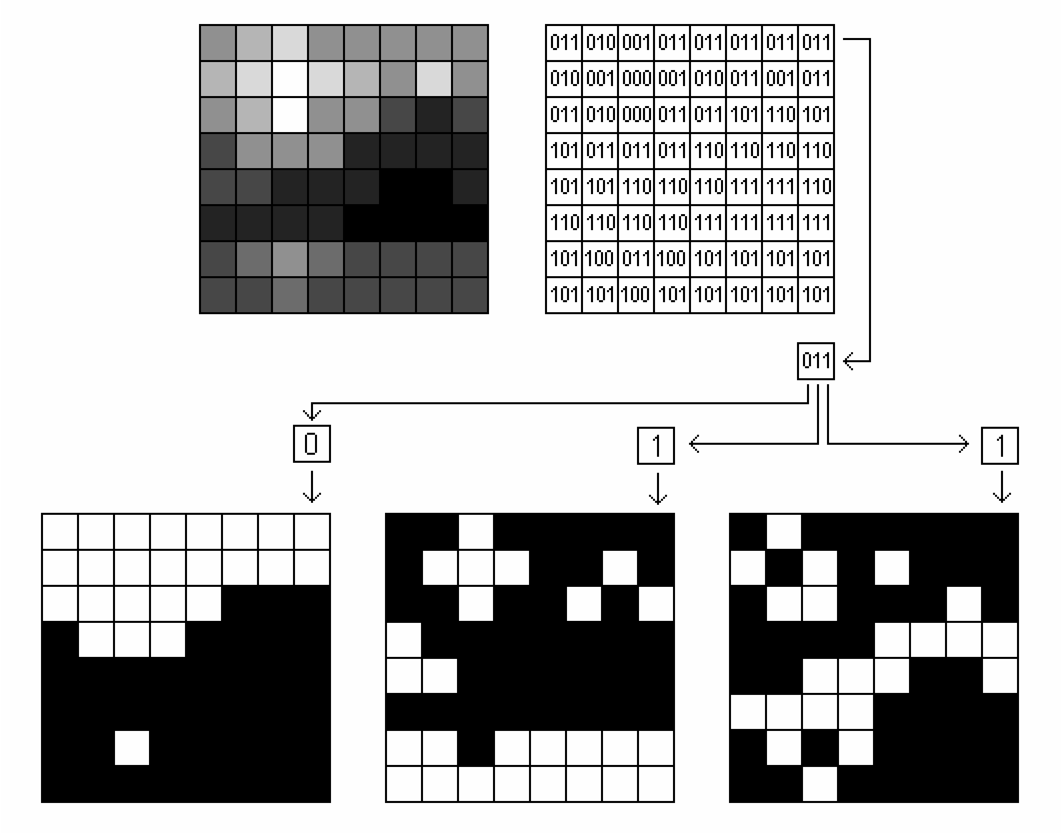


Figure 2-18 A Halftone Pattern Dictionary Region (6x6 grid)

### 2.5.8 HALFTONE REGIONS

The content of a halftone region is actually a multi-level image (or grayscale) image. Since JBIG2 supports only bi-level images, the multi-level image is stored in the bitstream as a sequence of bi-level images, with each bi-level image corresponding to one plane of the multi-level image. This method of constructing a multi-level image from a bi-level image is a common use of bi-level image compression, and an example is illustrated in Figure 2-19.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 011 | 010 | 001 | 011 | 011 | 011 | 011 | 011 |
| 010 | 001 | 000 | 001 | 010 | 011 | 001 | 011 |
| 011 | 010 | 000 | 011 | 011 | 101 | 110 | 101 |
| 101 | 011 | 011 | 011 | 110 | 110 | 110 | 110 |
| 101 | 101 | 110 | 110 | 110 | 111 | 111 | 110 |
| 110 | 110 | 110 | 110 | 111 | 111 | 111 | 111 |
| 101 | 100 | 011 | 100 | 101 | 101 | 101 | 101 |
| 101 | 101 | 100 | 101 | 101 | 101 | 101 | 101 |

Figure 2-19 Representing a 3-bit Multi-Level image with 3 Bi-Level Images

The method used to store the multi-level image in a halftone region differs slightly from this example, in that the values used to represent the grayscale levels are Gray coded, which is a purely co-incidental use of the word gray. Each binary plane is actually the exclusive-or (XOR) of itself and the previous plane. The halftone region is simply each bit plane coded with the generic coder.

When the decoder receives the halftone region segment, it generates the multi-level image and then constructs a halftone into the region using the image and a pattern dictionary. JBIG2 halftones are always ordered, and must have a fixed grid size. The grid size, grid placement and grid angle information are all stored in the data header of the halftone region. For every grayscale value in the multi-level image, there must be a corresponding pattern in the pattern dictionary. In Figure 2-20, the decoding procedure for a JBIG2 halftone is illustrated. In this example, the grid size is 6x6 which gives 37 possible grayscale values, requiring 6 bits per value, and therefore 6 planes. The advantages of halftone coding are visible in this example, where 6 small images and a small pattern dictionary are coded instead of one large image.

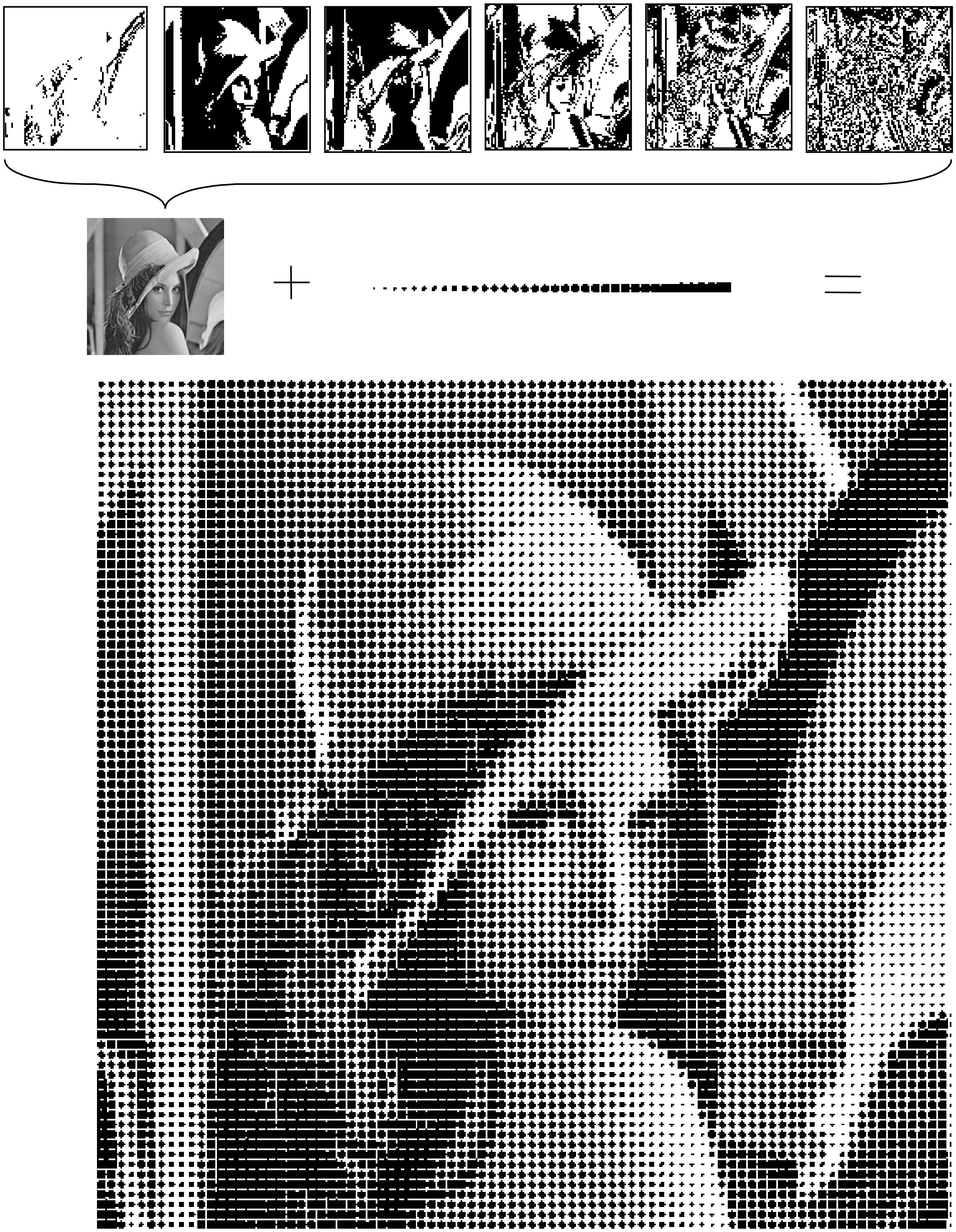


Figure 2-20  Decoding Procedure for JBIG2 Halftones

This method of spatial reduction can be used in JBIG2 for non-halftone applications. For example, if a pattern dictionary has only two patterns, then the multi-level image would actually be a bi-level image, but much smaller than the original. This method of resolution reduction can be used to achieve very high compression, and is illustrated in Figure 2-21.

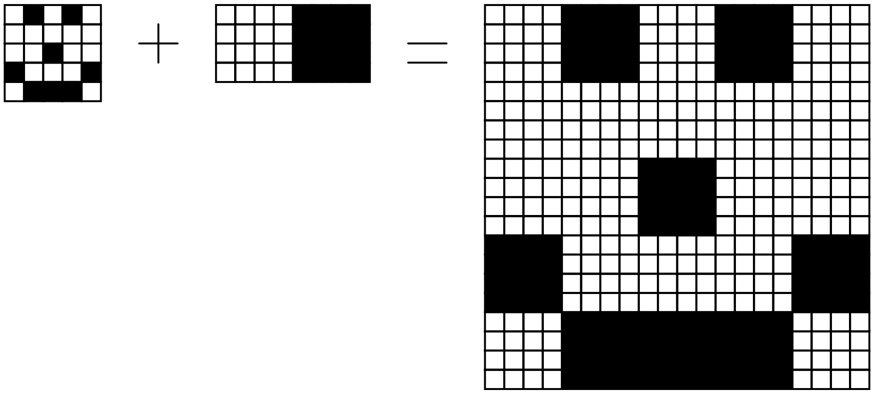


Figure 2-21 Using JBIG2 Halftones for Resolution Reduction

# CHAPTER 3. MEASURING DISTORTION

To properly evaluate a lossy compression method, the distortion must be measured. In this chapter, mechanisms for measuring distortion in bi-level images are presented. The first section outlines the difficulties encountered while measuring distortion. The next two sections describe distortion measures that can be used for textual and halftone images, respectively. The last section addresses the challenge of selecting an appropriate distortion measure.

## 3.1 BI-LEVEL DISTORTION MEASUREMENT PROBLEMS

In this section, the problems associated with measuring bi-level image distortion are demonstrated. When measuring distortion in an image, there must be a source or reference image that it will be compared against. Any differences that may exist between the two images is called the distortion. A distortion or distance function d(a,b) measures the differences between two images.

The most intuitive measure of distortion for bi-level images is the Hamming distance [Witt94a]. The Hamming distance is simply the number of bits (or pixels) that are different in the two images. The Hamming distance d<sub>H</sub> between an image a and an image b is formally defined as:

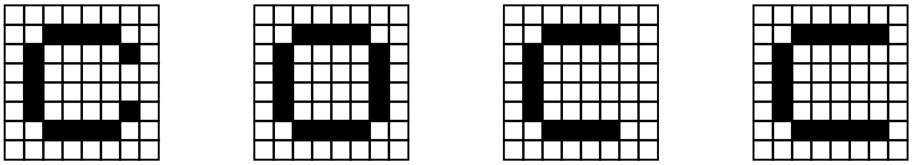
$$d_H = \sum\nolimits_{i,j} \left| a(i,j) - b(i,j) \right| \qquad \begin{array}{l} 0 < i < W-1 \\ 0 < j < H-1 \end{array}$$

Where a and b are the same size and have height H and width W. The Hamming distance is easily obtained by counting the number of ones remaining after the two images are combined with an XOR operator. Two related quantities to the Hamming distance are the Bit Error Rate (BER) and the Peak Signal to Noise Ratio (PSNR), which for bi-level images are defined as:

$$BER = \frac{d_H}{H \cdot W} \cdot 100\%$$

$$PSNR(dB) = 10 \log_{10} \left( \frac{H \cdot W}{d_H} \right)$$

Figure 3-1 gives the Hamming distances between an original image (a) and three other images.



(a) $d_H(a,a) = 0$

(b) $d_H(a,b) = 2$

(c) $d_H(a,c) = 2$

(d) $d_H(a,d) = 4$

Figure 3-1 Measuring the Hamming Distance $d_H$

From the simple images in Figure 3-1, one of the difficulties with the Hamming distance measurement can be seen. To the casual observer, it might appear that images (c) and (d) are closer to the original image than (b), while the Hamming distance measurement does not reflect this. That is simply because the casual observer does not use the Hamming distance on an array of bits, but rather interprets the images as the letters ‘c’ and ‘o’. This simple example illustrates one of the fundamental problems with all distortion measures: the measurement may not reflect how a human will interpret the distortion. The images in Figure 3-2 help to further illustrate this problem.

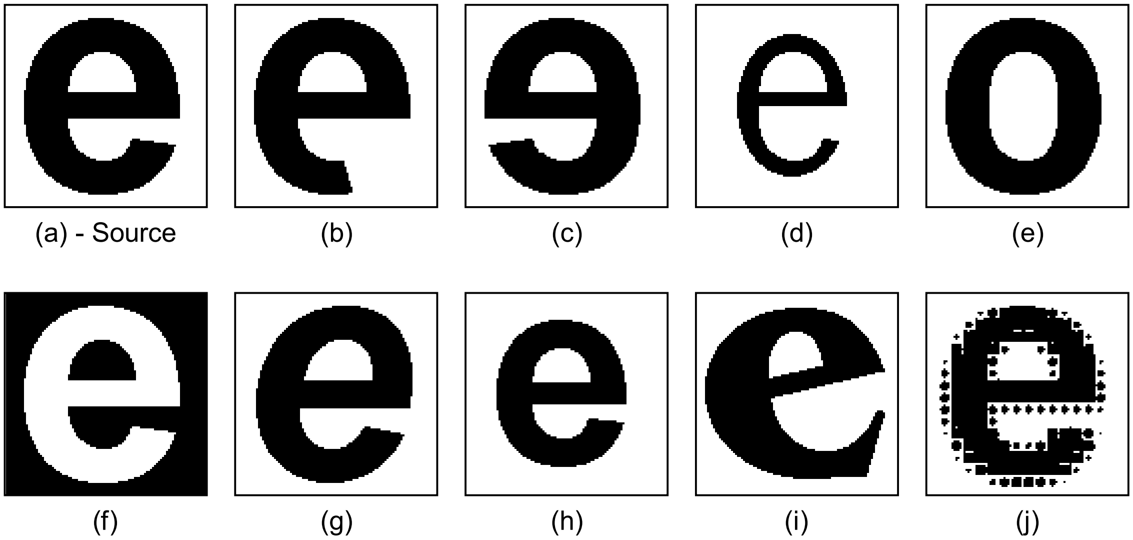

(a) - Source
(b)
(c)
(d)
(e)
(f)
(g)
(h)
(i)
(j)

Figure 3-2 Textual Test Images for Measuring Distortion

In Figure 3-2, image (c) is has the smallest Hamming distance to the source (a), demonstrating again that the Hamming distance is not always a useful measure of distortion. Each of the images in Figure 3-2 have some characteristics in common with the source image. If several different individuals were to rank the images from “closest” to “farthest” then there would be large discrepancies in their responses. Surprisingly, if the images are viewed from a sufficiently large distance, image (j) becomes the closest to the source. This phenomenon will be explained in Section 3.3.

If the source image is part of a larger text document, then distortion can be measured when one of the images is substituted for the original image. Under these circumstances, one would expect that images (o) and (m) would introduce the most distortion, as they could change the way that the text document is interpreted. This is illustrated in Figure 3-3. If the actual font characteristics are to be considered, then images (g), (h) and (i) all have only one font characteristic changed from the original source: italicized, point size and font type respectively.

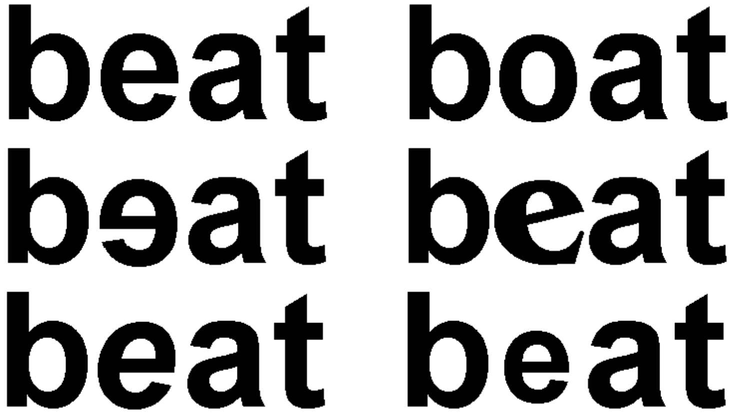


Figure 3-3 Distortion Introduced by Character Substitution

As demonstrated above, when textual images are considered, distortion can appear in many different forms. However, distortion in halftone images can appear in even more ways, which is illustrated in Figure 3-4.

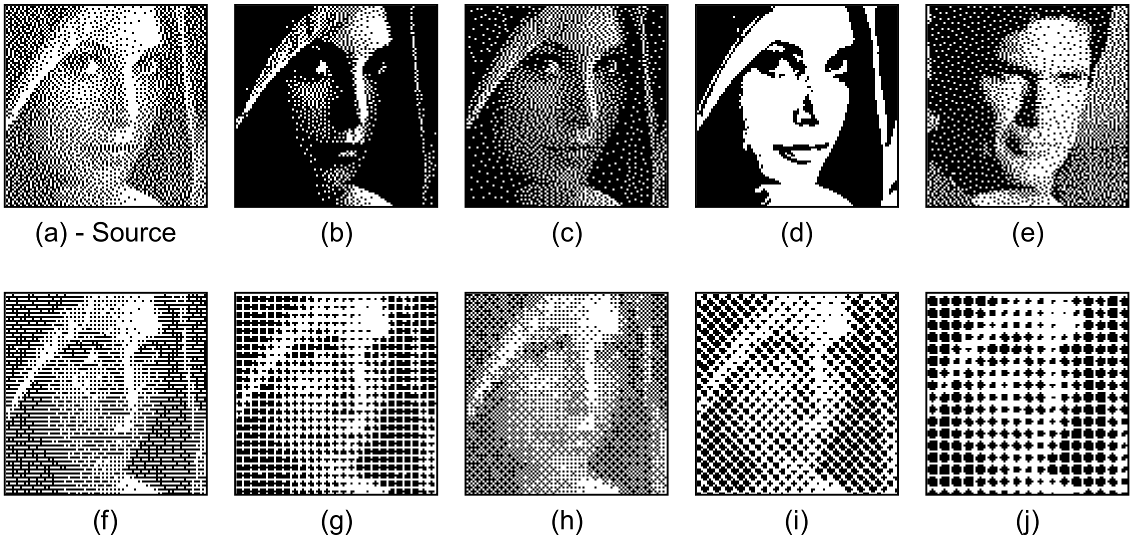


Figure 3-4 Halftone Test Images for Measuring Distortion

Figure 3-4 presents some interesting results. Using the Hamming distance, image (d) is the closest image to (a). However, what is even more surprising is that according to Hamming distance distortion measure, image (e) has a smaller distortion rate than figures (b) and (c), which is an observation that very few humans would make. Comparing the second row of images to the source image, the effect of various halftone patterns can be seen, and how the quantity of distortion seems to reflect the quality of the halftone. Images (f), (g) and (j) represent horizontal clustered dot patterns of 2, 4 and 6 respectively, which get progressively worse in visual quality, but maintain the correct average gray level. Images (g), (h) and (i) all have the same approximate grid size, but use three different halftone patterns. In general, the quality of the halftone depends on the halftone size, the halftone angle, and the viewing distance. Figures (g) and (i) illustrate the importance of grid angle, where their relative quality depends on the angle at which the page is viewed, unless the viewing distance is sufficiently large, at which point their quality is about the same. From these images, it can be seen how difficult it is to judge the quality and measure the distortion of a halftoned image.

It is clear that measuring distortion in a bi-level image is not a simple task. Caution must be used when applying any distortion measure to an image, as any numerical result may be nonsensical, or not properly reflect how a human will interpret the results.

## 3.2 MEASURING DISTORTION IN TEXT IMAGES

There has been much interest in measuring the distortion or difference between two textual images, for the applications of both data compression and Optical Character Recognition (OCR). Most distortion measures are focused on individual characters, or on a small region of the image, not on the entire image. In addition to the Hamming distance, there are several other important distortion measures, which are described below.

### 3.2.1 WEIGHTED XOR (WXOR)

The use of the WXOR function in compression was first suggested in [Prat80], and has been an extremely popular measurement for textual images. The WXOR function is defined as

$$d_{WXOR} = \sum_{i,j} \sum_{x,y} \left| a(i+x, j+y) - b(i+x, j+y) \right| \qquad \begin{array}{l} 0 < i < W-1 \\ 0 < j < H-1 \\ -1 < x < +1 \\ -1 < y < +1 \end{array}$$

where a and b are images of height H and width W. In other words, an XOR of the two images is produced, and for every pixel in the XOR, the number of pixels that are black in the surrounding 3x3 window are counted. Essentially, the WXOR function weighs clustered pixel errors much more than isolated pixel errors. Consider again the images of Figure 3-1. The value of $d_{WXOR}(a,b) = 4$, while the value of $d_{WXOR}(a,c) = 2$, indicating that the image (c) is indeed closer to (a). There are some excellent modified WXOR techniques described in [Witt97a] that compensate for small changes between nearly identical images, such as one technique that weighs the black-to-white differences and white-to-black differences differently.

### 3.2.2 CHARACTER PROPERTIES

Some very simple properties of the two characters being compared can be used to measure their distance. The most obvious choices are height, width, perimeter, the black pixel count, and the area enclosed by the character [Prat80].

### 3.2.3 HAUSDORFF DISTANCE

The Hausdorff distance has been used to compare characters for compression [Hutt93]. The Hausdorff distance measures how close one shape is to another shape, and is defined as

$$\begin{aligned} d_{HAUS}(a,b) &= \max(h(A,B),h(B,A)) & \qquad i \in A \\ h(A,B) &= \max_{i} \min_{j} \left\| a_i - b_j \right\| & \qquad j \in B \end{aligned}$$

where A and B are the set of black pixels in images a and b, and $||a_i - a_j||$ measures the Euclidian distance between element i of A and element j of B. The Hausdorff distance is a very effective measurement when one of the two images is a simple transformation of the other (ie: stretch, scale, rotation, tilt, etc.). For example, by using the Hausdorff distance it can be found that the images (a) and (g) in Figure 3-2 are similar.

### 3.2.4 FONT ATTRIBUTES

There has been a large volume of work dedicated to detecting characters for OCR applications, and a good summary of that research is in [Case96]. Many of these papers use neural networks and similar classification methods to determine the font attributes, and those font attributes can be used to determine the distance between two characters.

### 3.2.5 ADVANCED SHAPE CHARACTERISTICS

There appears to be a great opportunity to use advanced shape characteristics, such as the methods described in [Roch92], for the purposes of data compression. These methods include comparing the individual strokes of two images to determine how closely they match. Figure 3-5 gives a rough indication of what is involved with these methods.

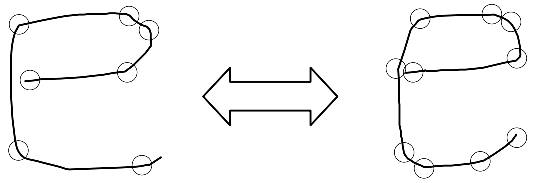


Figure 3-5 Advanced Shape Characteristics

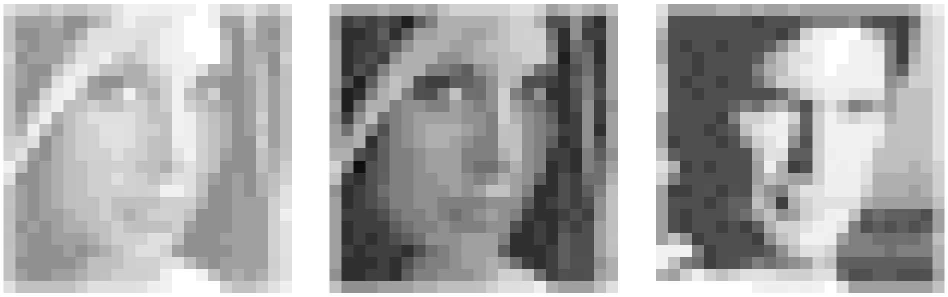
## 3.3 MEASURING DISTORTION IN HALFTONE IMAGES

The main objective of a halftoned image is to give the appearance of a multi-level image. When considering a distortion measure for halftoned images, it seems prudent to keep this objective in mind. In this section, several different measures of distortion for halftoned images are discussed.

### 3.3.1 MULTI-LEVEL MEASUREMENTS

As mentioned previously, the Hamming distance, or BER is not a very good measure in halftoned images, emphasized by the example of image (e) in Figure 3-4. For bi-level images, the BER is directly related to the Mean Squared Error (MSE) and the PSNR, which are two common related distortion measurements for multi-level images. Often, these measurements are performed on images that has been transformed with a transform such as the Discrete Cosine Transform (DCT). Because the BER, MSE and PSNR are generally invalid for bi-level images, they are equally invalid for transformed bi-level images.

Several techniques exist for performing an inverse halftone operation [Kite98,Dame98,Vall99,Mart00] which constructs a multi-level image from a bi-level image. The topic of inverse halftoning is further discussed in Chapter 8. Once the image is represented as a multi-level image, then many of the aforementioned techniques become relevant. When the MSE is calculated for the inverse halftoned images of Figure 3-6, it is found that image (c) is indeed closer to (a) than (e).



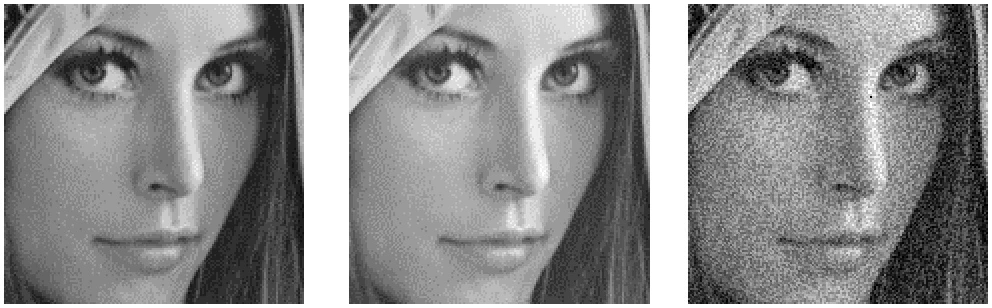
(a) - Source (c) (e)

Figure 3-6 Inverse-Halftoned Images (from Figure 3-4)

### 3.3.2 HUMAN VISUAL SYSTEM (HVS) MEASUREMENTS

Since the final processors of a halftone will eventually be the human eye and the human brain, it is worth exploring what role they will have in the interpretation of a halftone. There is a vast amount of literature on the subject, including the method that is briefly described here [Dame00].

In the HVS, there are two primary ways in which error is measured: linear and non-linear distortion, where non-linear distortion generally appears much worse. This is illustrated in Figure 3-7, where the image (b) has the same MSE as (c), but does not appear to have as much distortion. There are straightforward methods available for detecting linear distortion, but non-linear distortion, which is sometimes called *additive noise,* is more difficult to measure, and requires a detailed analysis of the HVS.



(a) - Source (b) - Linear Distortion (c) - Non-Linear Distortion

Figure 3-7 HVS Effects of Linear and Non-Linear Distortion

As mentioned previously, the quality of a halftone seems to depend on both the angle and distance at which it is viewed. This phenomenon can be related to the actual sensitivity and alignment of the rods in the eye [Hunt95]. To measure this HVS property, experiments have been conducted where human observers view images such as the ones in Figure 3-8 to determine at which distance the images appear completely gray. A function has been developed to describe this HVS behaviour, and is known as the Contrast Sensitivity Function (CSF) [Mann74,Sull93], defined as

$$H(f_r) = 2.6(0.0192 + 0.114f_r)e^{-(0.114f_r)^{1.1}}$$

$$f_r = 2\frac{\sqrt{f_x^{\,2} + f_y^{\,2}}}{(0.3)\cos\left(4\tan^{-1}\left(\frac{f_x}{f_y}\right)\right)+1.7}$$

where $f_x$ and $f_y$ are the angular frequencies of the image, measured in cycles per degree of viewing angle, which is illustrated in Figure 3-9. The CSF is illustrated in Figure 3-10. In this figure, the dip at 45° can be seen, illustrating how the HVS is less sensitive to angled patterns. This figure also demonstrates how the CSF can be flattened out at the top, corresponding to the low-pass filter effect that occurs when the human eye wanders.

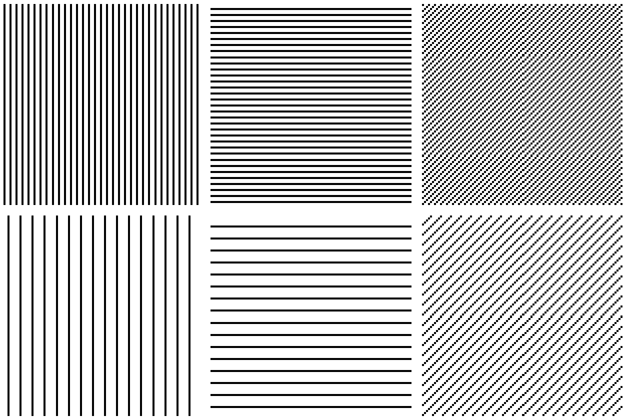


Figure 3-8  Sample Images for Measuring HVS Responses to Contrast

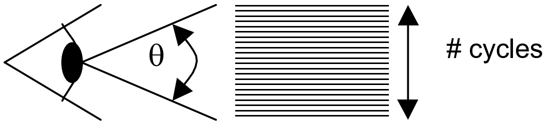


Figure 3-9  Number of Cycles per Degree of Viewing Angle

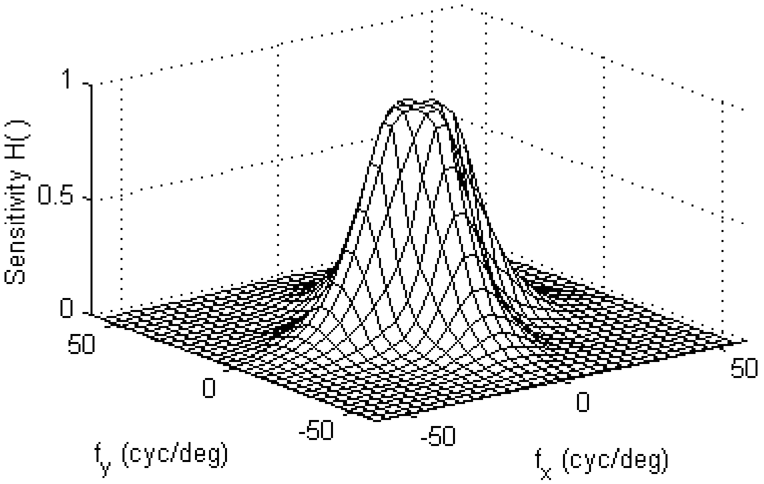


Figure 3-10  Contrast Sensitivity Function (CSF)

## 3.4 CHOOSING A DISTORTION MEASURE

With all of the distortion measures listed in Sections 3.2 and 3.3, in addition to the numerous measures not listed, the task of choosing a suitable measure for a rate-control application seems daunting. Without the proper distortion measurement, it may be impossible to properly evaluate the lossy compression performance of a system. The ideal solution would be to design a method of rate-control that is independent of the distortion measure. However, as the subsequent sections will outline, some of the rate-control methods proposed are designed around specific measures of distortion.

For halftone images, the objective would be to have a method that incorporates both linear distortion and additive noise, which would be weighted with the CSF. While these measurements can be performed in a post-processing step to analyze results, having them as an integral part of the rate-control algorithm seems impractical. Fortunately, an approximation can be used. If a MSE analysis is performed on a filtered, downsampled version of the image, it will approximate the low-pass nature of the CSF. In other words, if smaller, multi-level images are used to represent the actual images, such as the images in Figure 3-6, then MSE distortion measurements can be used without much penalty. This distortion measure does not isolate and separately measure linear distortion, but since distortion in a rate-control method is usually added in a non-linear fashion, linear distortion can be ignored for the sake of simplicity.

For generic images, there is no suitable method to select a distortion measure, since the type of image is generally unknown. The Hamming distance can always be used, and for many generic images, it may actually be the best measure available. Alternatively, the halftone method described above could be used, or possibly one of the methods used for text images, such as the WXOR, the Hausdorff distance, or another advanced shape method. However, as Chapter 5 will explain, unless there is outside information that can assign a specific distortion weight to each pixel, the only method that is suitable for the proposed generic rate-control method is the Hamming distance.

The proposed method for text images is actually quite flexible, and can use any type of distortion measure. As discussed in Section 3.2, it is difficult to choose one single distortion measure that will work for all types of distorted textual images. Fortunately, a weighted average of several distortion measures can be chosen to improve the robustness. For example, consider the following distortion measure:

$$d(a,b) = \begin{bmatrix} |Height(a) - Height(b)| \\ |Width(a) - Width(b)| \\ |Area(a) - Area(b)| \\ |Perimeter(a) - Perimeter(b)| \\ \left|\sum_{i,j} a(i,j) - \sum_{i,j} b(i,j)\right| \\ d_{HAMMING}(a,b) \\ d_{WXOR}(a,b) \\ d_{HAUSDORFF}(a,b) \end{bmatrix} \bullet \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \end{bmatrix}$$

where the result of each individual measure would be weighted and scaled appropriately, to give an overall measurement of how close the two textual images are.

# CHAPTER 4. IMAGE SEGMENTATION

In the following chapters, methods for controlling the rate of a region are proposed which require the region to be pre-classified. In this chapter, a method is proposed for separating text from non-text regions. The first section reviews existing methods and addresses the requirements of a JBIG2 segmentation method.  The second section describes the proposed method.  The third and final section presents the results.

## 4.1 BACKGROUND

Traditionally, document segmentation algorithms have been developed for Optical Character Recognition (OCR) applications.  A historical summary of the available methods can be found in [Mori92,Jain98].  Segmentation methods can be loosely classified as bottom-up, top-down or both.  Bottom-up approaches to document segmentation generally start with individual letters on a page, and then based on text-layout conventions, group letters into words, words into paragraphs, and so on.  Line-art and halftones are often detected by their size, or their non-text layout.  Top-down approaches take advantage of the fact that formatted documents usually have margins surrounding each region. The page can be subdivided into different regions by examining the white space in the document. Alternatively, top-down methods will use the bit-density or texture of the document to identify and classify regions. Some modern segmentation methods still have problems with skew (text appearing on an angle), non-rectangular regions, reverse-coloured text, and foreign languages. Many methods also require large quantities of training sets to properly identify regions.

Although the general objectives of JBIG2 segmentation and OCR segmentation are similar, their requirements are quite different. In general, an OCR segmentation method has to be much more accurate. Misinterpreting a block of text as a graphic or not detecting a region of reverse-coloured text may be catastrophic in an OCR application, while in a JBIG2 environment similar errors will only lead to sub-optimal compression performance or increased distortion.

Because they are required to be more accurate, OCR algorithms are generally slower. In general, symbol based bottom-up approaches are faster than top-down strategies as the number of symbols is much smaller than the number of pixels [Witt97a]. Most bottom-up strategies require that all of the symbols in the document are extracted. Although symbol extraction can be executed relatively quickly, the analysis of the symbols can be quite costly. For a proper analysis, some sorting or comparison operations must be performed, which can be of complexity $O(n \cdot \log_2 n)$ or more. Images with halftone regions can easily have over 10,000 symbols, which may make even a simple analysis too costly. JBIG2 segmentation methods should avoid a full symbol analysis.

A JBIG2 segmentation method must also consider the consequences of misinterpreting a region type. There will be considerable loss in quality if a lossy halftone coding method is used for a text region. However, the danger in misinterpreting non-text data as text is not the slightly smaller compression performance, but rather the potentially large difference in the number of computations required.

The symbol analysis required in a text coder is even more complex than a segmentation scheme, and may have a worst-case complexity of $O(n^2)$. If a halftone region with 10,000 symbols is misinterpreted as text, the result can be catastrophic. As a result, performance sensitive JBIG2 encoders should be biased towards non-text regions, which is the opposite of the guideline for OCR applications.

## 4.2 SEGMENTATION METHOD

The objective was to develop a fast method of separating text from non-text regions, avoid performing a costly symbol analysis at the segmentation stage, and avoid classifying a halftone region as text. An additional objective was to find a mechanism for detecting regions of reverse-coloured text.

The proposed method requires that an analysis be performed on a reduced image. Instead of reducing the image by downsampling, a block technique is employed [Sait91]. Each pixel in the reduced image corresponds to a NxM block in the original image. A reduced pixel is white if and only if all of the pixels in the corresponding block are white. Based on this reduction criterion, the reduced image appears dark and smudged, which is why the technique is often called smearing. The technique is illustrated in Figure 4-1.

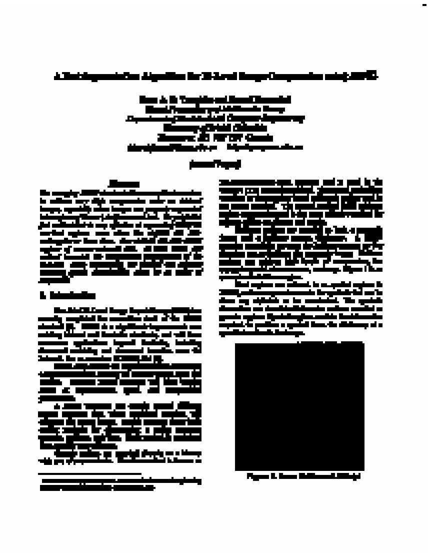


Figure 4-1  Smearing Reduction Technique

Where a bottom-up approach could have been quite slow on the full image, it is now quite feasible on the reduced image.  After reducing the image, all of the symbols are extracted using the 8-connected technique described in [Witt97a].  Each symbol is then examined to determine if it has non-text characteristics.  In general, halftone regions and line-art will appear as large black symbols, while text regions will consist of several small symbols.  If a symbol is determined to be non-text, then all of the pixels corresponding to the symbol in the original image are removed as a region.

In the reduced image, reverse-coloured text regions will have the same characteristics as halftone regions.  When a non-text region is removed from the original image, a test us performed to see if it is reverse-coloured text.  To perform this test, the region is reversed and then the analysis is repeated.  If the reduced image is still a large black blob, then it is most likely a halftone. Conversely, if there are a large number of small symbols, then it is most likely text. The process of reversing the region, reducing the image and extracting the symbols can all be performed very quickly, and is illustrated in Figure 4-2.

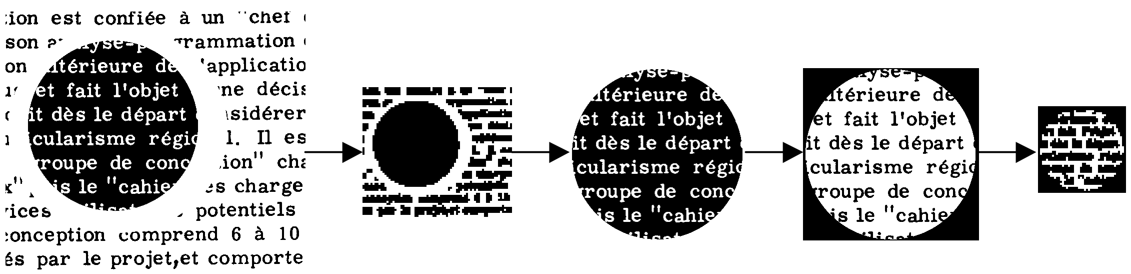


Figure 4-2 Detecting Reverse-Coloured Text Regions

The segmentation method can be summarized as follows. The original image is reduced, and all of the symbols from the reduced image are extracted. Each symbol corresponds to a region in the original image, and is checked for non-text characteristics. If the symbol is classified as non-text, the corresponding region from the original image is removed. The removed region is reversed, reduced and the symbols are extracted. If there are a large number of symbols, the region is classified as reverse-coloured text. Otherwise, the region is classified as non-text. After the non-text and reverse-text regions have been removed from the original image, the remaining region is classified as text. For each of these steps, there are thresholds and parameters that will determine how sensitive the implementation is.

The first parameters to consider are the dimensions of the reduction block, N and M. For an efficient implementation, it is vital that N be a multiple of 8. More than likely, the image data will be stored as 8 horizontal pixels per byte, and restricting N to multiples of

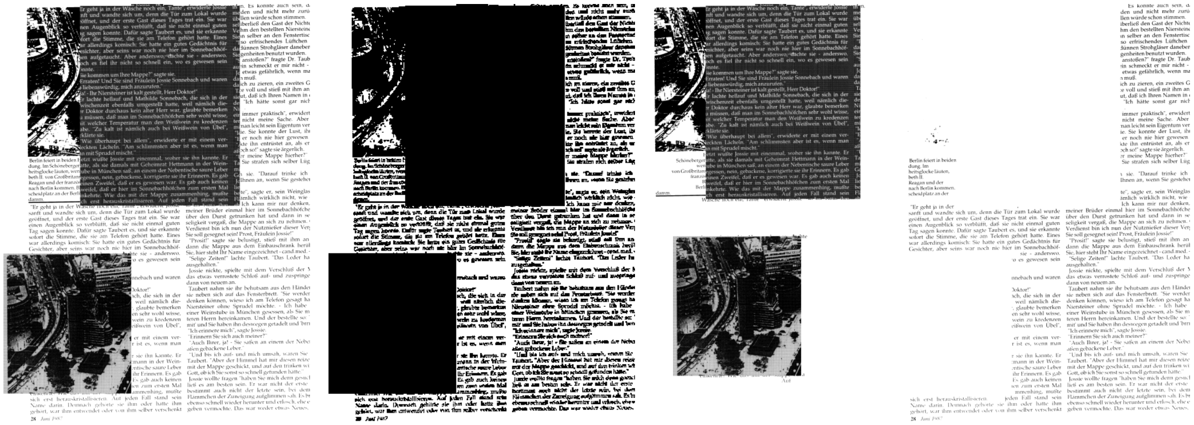
8 will allow the reduction and region extraction operations to be performed in a byte-wise manner, significantly improving the speed of the encoder. There is no such restriction on M, but by keeping M and N equal, the method becomes invariant to image orientation, which may be a desirable feature. For images at 200 dpi with 10 point (or higher) text, a block size of 8x8 works well. The 8x8-block size is also effective at higher resolutions, but a larger block size can be used to improve the speed even further.

To determine if a symbol in the reduced image corresponds to a non-text region, the weight and the size of the symbol are considered. The weight of a symbol is the number of black pixels it contains. The symbol is classified as non-text if its' weight is above a certain percentage of the weight of the entire reduced image. Additionally, if the area of the symbol's bounding box is above a certain percentage of the entire area, then it is very likely the corresponding region is line-art or a figure. The threshold percentages will determine the sensitivity of the segmentation method. High percentages may allow small halftone regions to go undetected, while low percentages may misinterpret dense text regions as non-text. The thresholds of 15% worked well for the test images.

To determine if a region contains reverse-coloured text, the number of symbols in the reversed, reduced image is counted. If a sufficient number of symbols are detected, then it will be worthwhile to encode the region as text. The threshold for the number of symbols must be large enough to avoid misinterpreting very dark halftones as text, and small enough to ensure reverse-text regions are not ignored. If large symbols are excluded from the reverse-text region (using the same criteria as before) the risk of interpreting a halftone as text is reduced, and the threshold can be much smaller. The value of 30 symbols worked well for the test images. It should be noted that the example in Figure 4-2 uses a small threshold for illustrative purposes.

## 4.3 RESULTS

Figure 4-3 illustrates the results the proposed method on the image known as CCITT2 or F17_400.



**(a)** **(b)** **(c)** **(d)**

Figure 4-3 Segmentation Method

The original image (a) contains 42,423 symbols. The image is reduced (b) and now contains only 717 symbols. Two of the symbols are detected as a non-text and their corresponding regions are removed (c). The remaining image (d) can now be easily compressed as a text region, with only 4,933 symbols. In (c) two errors can be seen. Even though the top region contains two different sub-regions, a line-art and a halftone region, it is removed and encoded as one region. In addition, some lines of text are included in the region. Both of these errors occur because there was not enough white space between the regions. In (d) it can be seen how part of the line art has remained with the text region. These errors result in a small reduction in the compression performance, and possibly some additional distortion. Even though the image in Figure 4-3 is very complicated, the segmentation works well, and demonstrates how the proposed method handles irregular shapes and text regions with ease.

# CHAPTER 5. RATE CONTROL IN GENERIC REGIONS

At the core of the JBIG2 standard is generic coding. As seen in Section 2.5, most of the data contained in a JBIG2 is coded generically, either through generic segments directly, or through symbol dictionaries, pattern dictionaries and halftone regions. Since the overall objective is to achieve rate-control for a JBIG2 coded image, it would seem that the best approach would be to develop a method of achieving rate-control for a generic region. However, this is not the case. In this chapter the difficulties involved with achieving rate control for a generic region are demonstrated, and the limited amount of rate-control that can be achieved with a method known as bit-flipping.

## 5.1 BACKGROUND

The method of bit-flipping was first proposed in [Koss96] and is a method of introducing distortion into an image by “flipping” pixels either from black to white or vice-versa in a manner that improves compression. Essentially, if a pixel is the Least Probable Symbol (LPS), it can be very expensive to code. The approximate cost to code an LPS is

$$-\log P_{LPS} \tag{5-1}$$

For example, if the LPS is white, and $P_{LPS}$ is 0.2, then it will cost 0.32 bits to encode a black pixel, and 2.32 bits to encode a white pixel. If the next pixel in an image is white, then there would be a coding gain if it was "flipped" to black, and encoded with much fewer bits.

Although the mechanism of bit-flipping seems straightforward, there are numerous repercussions when a pixel is flipped. The image in the following figure will help illustrate some of those issues.

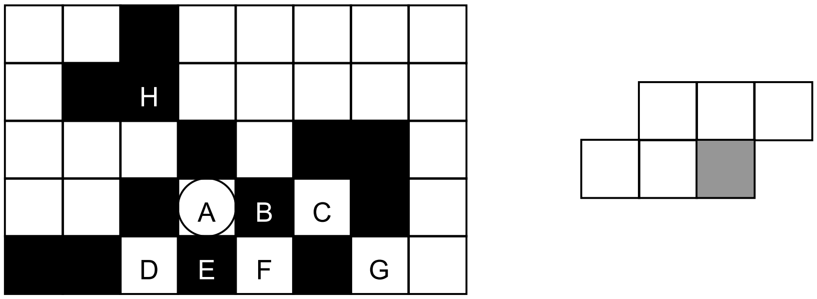


Figure 5-1 Image for Analyzing Bit-Flipping

The image in Figure 5-1 is a small section of a larger image to be generically coded. This example uses the context shown, which is not a valid JBIG2 context, but rather a simplified context for this example. The next pixel to be coded is pixel A, and the context for A specifies that the MPS is black. The effects of flipping the value of A from white to black will be considered.

Since black is the MPS, flipping the value of A will result in fewer bits required to code that pixel. However, flipping that value of A will not just change the code length for that pixel, but also every other subsequent pixel that shares the same context. Consider pixels G and H, which have the same context as A. Flipping A will not change the cost to code H, but by flipping the value of A, the statistics for coding that context become even more biased towards black. As a result of flipping A, the cost to encode G has actually increased and is larger than the original cost to code A. Now consider the situation where every subsequent time the context of A is encountered, the pixel is white. Under those circumstances, it would be better to leave A white, which would start changing the statistics for that context to become white more rapidly. Knowing this information, it may turn out that better compression could have been achieved if H was flipped, even though it was the MPS at the time.

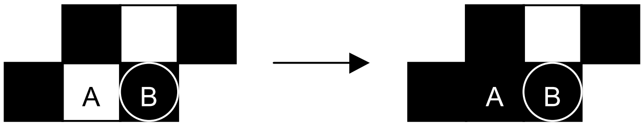


Figure 5-2 Effects of a Bit-Flip on Subsequent Contexts

Moreover, flipping A not only changes the statistics for A's context, but also all of the contexts for pixels B, C, D, E and F. This will cause a variety of changes to the compressed image size. First, consider the changed context for pixel B, illustrated in Figure 5-2. Instead of the context on the left, pixel B now has the context on the right. If the MPS for the first context was black, and the MPS of the second context is white, then it will now cost more to code pixel B. This additional cost in coding pixel B may be more than the benefits gained by flipping pixel A. This could also happen with pixels C, D, E, and F. This change in contexts can become even more convoluted, as B and E both previously had the same context, but now they have different contexts. Following this even further, if A is not flipped, then pixel F has the same context as A, and C and D have different contexts, but if A is flipped, then F no longer shares the same context, and suddenly C and D have the same context.

This elaborate example has demonstrated some of the numerous ways that flipping just one single pixel can change the entire compressed image size. Without a careful analysis, there is no simple way of knowing if flipping a pixel will result in an increase or decrease in compression performance.

## 5.2 EXISTING IMPLEMENTATIONS

In the method described in [Koss96], rate control is attempted with a single pass through an image. If coding a LPS will increase the rate above a threshold, then it should be coded as the MPS. The following equation determines the behaviour of the encoder:

$$\frac{w_i}{R_{MPS} - R_{LPS}} < T \tag{5-2}$$

where $R_{MPS}$, $R_{LPS}$ are the rates for coding with the MPS and the LPS, which are compared against a threshold T. The weighting in the current region is represented by ($w_i$). The application for which this method was developed had regions of interest where excessive bit-flipping was to be avoided, and was controlled with the value of $w_i$. In JBIG2 experiments, attempting to achieve rate control in one pass with this approach is rarely successful. Moderate success can be achieved on some simpler images by imposing constraints and rules to find the appropriate flip candidates and to avoid cascading errors.

More comprehensive approaches to bit-flipping are presented in [Mart98] and [Mart99]. The method described in [Mart99] will be referred to as the Martins-Forchhammer Bit-Flipping (MFBP) method. In MFBF, there is an initial pass of the image to gather statistics. For each context, a count of the ones and zeros is obtained, represented with the symbols $n_1$ and $n_0$, respectively. The compressed length of data required for coding the $n_1$+$n_0$ symbols is estimated with the following equation:

$$l(n_0, n_1) = -\log\left( \frac{\prod_{j=0}^{n_0-1}(\delta + j) \cdot \prod_{j=0}^{n_1-1}(\delta + j)}{\prod_{j=0}^{n_0+n_1-1}(2\delta + j)} \right) \tag{5-3}$$

where $l$ is the length of the data, and $\delta$ is the offset value of the arithmetic coder. To determine the total effect of flipping a pixel, the change in data length is recalculated for the pixel's context, and the for of the affected contexts. If the resulting change in data length is negative, then the pixel is a candidate for flipping. After an initial pass through the image, the potential change in data length can be calculated for each pixel, and a sorted list of flip candidates is produced. Once the ranked flip candidate list is available, a specific data rate can be achieved by flipping a sufficient number of the top candidates. Approximations and techniques are available to improve the complexity of the MFBF approach.

There are numerous problems encountered when using the above method for JBIG2 generic regions. The most significant problem is that Equation (5-3) assumes the data is stationary, which for a large range of images is an incorrect assumption. For non-stationary data, the MPS can change throughout the image, making calculations with $n_0$ and $n_1$ inaccurate. As discussed in Section 2.5.1, the value of $n_{LPS}$ is clamped, allowing the MQ coder to adapt to non-stationary data. This can introduce a significant amount of error into the estimate of Equation (5-3). The difference in data length that can be obtained with stationary and non-stationary data is shown in Table 5-1, which illustrates how difficult it is to make accurate length predictions based solely upon the values of $n_0$ and $n_1$.

| Pattern:<br>($n_1 = 33$, $n_0 = 66$) | Code Length<br>(First 99 bits) | LPS | $P_{LPS}$ | $-\log P_{LPS}$ |
|---|---|---|---|---|
| 111111……0000000000 | 47.1 | 1 | .063 | 3.99 |
| 0000000000……11111 | 52.1 | 0 | .164 | 2.61 |
| 0000…..11111…..0000 | 67.0 | 1 | .141 | 2.83 |
| 001001001001001….. | 106.0 | 1 | .422 | 1.25 |
| 000000111000000111…. | 97.3 | 1 | .422 | 1.25 |
| Random Configuration | 95.5 | 1 | .475 | 1.08 |
| Predicted Value | 94.6 | 1 | .335 | 1.58 |

Table 5-1 Non-Stationary Characteristics of the MQ Coder

## 5.3 PROPOSED IMPROVEMENTS

Despite the many inaccuracies and limitations of the MFBF method, it's rate-distortion performance is difficult to surpass. Numerous experiments have been conducted to achieve similar or better results with a one-pass, two-pass or N-pass system. The results of those experiments have ranged from catastrophic to good, and while there have been some exceptional specific results, no one single method has been as effective or suitable for a wide range of images and image characteristics. In the following sections, various procedures are proposed to improve the MFBF method.

### 5.3.1 DISTORTION MEASUREMENTS

The problem with all bit-flipping approaches is that it is difficult to judge the distortion introduced by flipping just one bit. The distortion measurement that is automatically used with bit-flipping is the Hamming distance, but as Chapter 3 discusses, that measurement is rarely optimal. Theoretically, the entire image could be evaluated with the bit correct, and then the image could be evaluated with the bit flipped, and there would be a certain amount of distortion introduced that could be measured. However, this approach would most likely be too costly, and furthermore, the distortion would also likely depend on whether adjacent bits were going to be flipped as well, which is probably not known at that moment. In the MFBF method, a mechanism is suggested to maintain the average gray level for halftoned images, but that may or may not be appropriate for other types of images. In some applications, there will be additional information which can provide useful distortion information. In those circumstances, a distortion array can be provided, which weighs the relative distortion that is introduced if a particular bit is flipped. If a distortion array is available, then the sorting process of the MFBF can be modified to re-order the flip candidates appropriately. This modification is used along with halftone coding in Chapter 8.

### 5.3.2 MQ ADJUSTMENTS

Since the MFBF method was developed for an ideal arithmetic coder with stationary data, the calculations are not always accurate. If the MFBF is to be used in a JBIG2 application, then there must be some method of compensating for those inaccuracies. As Table 5-1 suggests, knowing the sequence of bits to be coded will improve the quality of the estimate. However, storing such information requires vast amounts of memory, and becomes quite cumbersome. The simplest solution to the problem is to continuously replace MFBF estimates with actual values. In the first pass through the image, the codelengths can be calculated with the MFBF method, and measured directly from the MQ coder. The initial difference between the two, or the initial error, will be an indication of how the dynamics of the data and the specifics of the MQ coder deviate from the theoretical. One approach to improve the accuracy of the MFBF method is to scale all future estimates by the initial error. For small reductions in rate, this approach can be sufficient, but for large changes to the image, the error continues to creep. Although more adjustments and refinements can be made to the MFBF estimates, the most effective way to achieve accurate rate-control is to periodically encode the image to obtain fresh, accurate data. For two-pass and N-pass methods, this is performed automatically.

### 5.3.3 N-PASS METHOD

In an n-pass approach to the MFBF method, the image is encoded several times, each time reducing the rate further, until the target is achieved. In this method a list of flip candidates is not generated, but rather each pixel as it is encountered in raster-scan order is evaluated to determine if it should be flipped, which is a significant reduction in memory. A threshold value is established for each pass, and any bit that will save more than the threshold value is flipped. Initially, the threshold value is set high, and it is reduced at each iteration. Each iteration allows more bits to be flipped, reducing the rate until the target is achieved. If the threshold is lowered quickly, then the number of iterations will be reduced, but the quality of the image will slightly degrade, and the rate control will be less accurate. If the resulting rate is below the target, then there are three options: the method can repeat the last iteration with a higher threshold, the method can keep the last iteration as is, or a refinement step can be added, as proposed in Chapter 6. From experimental results, cascading artifacts are reduced if the values of $n_0$ and $n_1$ are updated at the end of each pass. To estimate the savings of flipping a bit, numerous contexts have to be calculated. With an N-pass method (or two-pass method), the contexts can be calculated quite easily, by sliding all of the context windows one pixel at a time. One interesting phenomenon of an N-pass approach is that the value of one pixel may continuously flip back and forth between black and white with each iteration, depending on the dynamics of the surrounding pixels. Although this is generally not a problem, care must be taken to ensure that this does not create an endless loop, and that bit-flipping savings are not accidentally re-counted. Overall, an N-pass method can be used as an alternative to the traditional MFBF to achieve very accurate rate control, with a lower memory requirement.

### 5.3.4 TWO-PASS METHOD

In a two-pass method, the statistics are first generated for the image, and then all of the bit-flipping and rate-control is performed in the second pass. The two-pass method does not achieve the lowest possible distortion, and it may be impossible to achieve low target rates with just two-passes, so this method should only be used when there are performance constraints, or when there is a small reduction in rate. In the first pass, additional information is collected, specifically, the compression rate for each scan line in the image. By measuring the rate for each line, a target rate can be specified for each line. By knowing this information, the decision to flip an individual bit is not solely based on the cost savings, but also whether or not the encoder is on track to achieve the target rate. If the encoder is behind the target rate, then all cost-saving bits will be flipped. If the encoder is ahead of the target rate, then only bits with large savings (above some threshold) will be flipped, which will reduce the total number of bits required to be flipped subsequently. Because the actual bitstream is encoded in the second pass, some additional adjustments to the MFBF method can be implemented, to model the behaviour of the MQ coder. If the MPS is the same as the maximum of $n_0$ and $n_1$, and an MQ renormalization will occur with the coding of an MPS, then that pixel should be flipped, unless there is an overwhelming detriment to the surrounding contexts. If for a particular context, the values of $n_0$ and $n_1$ are close (within 20% of each other), then the codelength formula of the MFBF can be ignored, and the decision to flip a bit or not can be based upon the current MPS. Overall, the two-pass method can produce reasonably good results for small reductions in bitrate, but does not produce the quality that a general MFBF method can.

## 5.4 RESULTS

To demonstrate the performance of the modified MFBF method, the N-pass method has been performed on three 250x250 images: a generated halftone (Small_Lena), a scanned textual image (Small_F04_200) and a generated line-art image (Graduate). Similar results have been achieved with larger images, but smaller images have been chosen so the distortion artifacts can be seen. The rate-distortion curves from the experiments are illustrated in Figure 6-1.

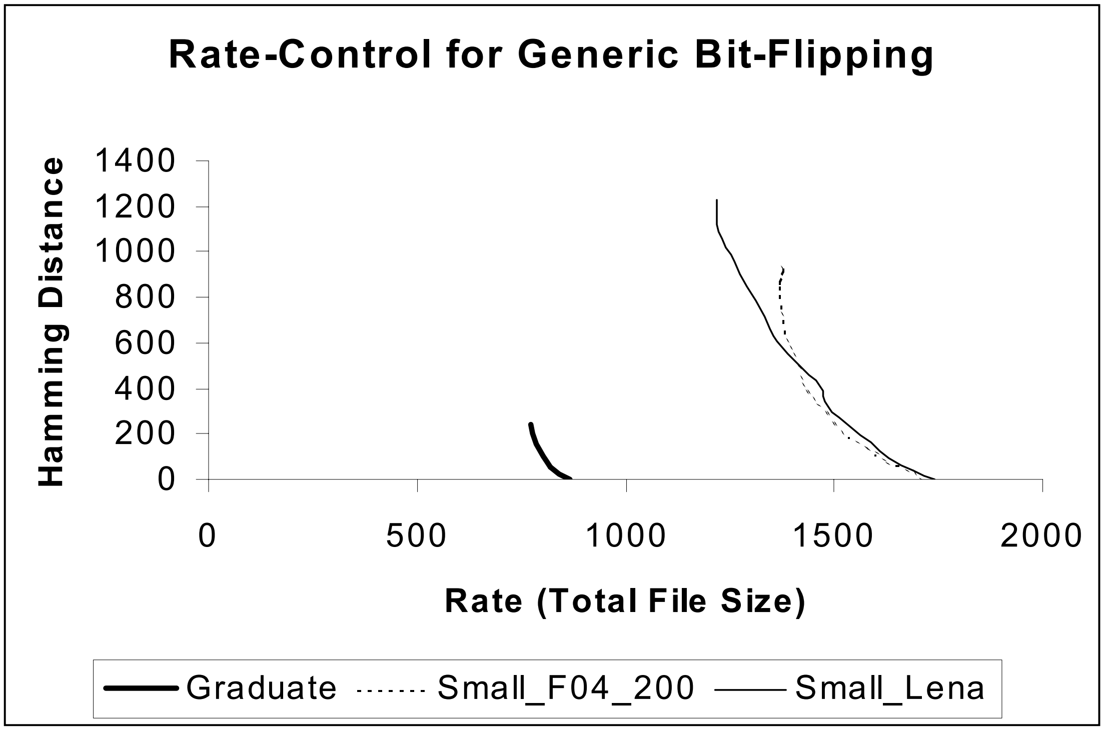
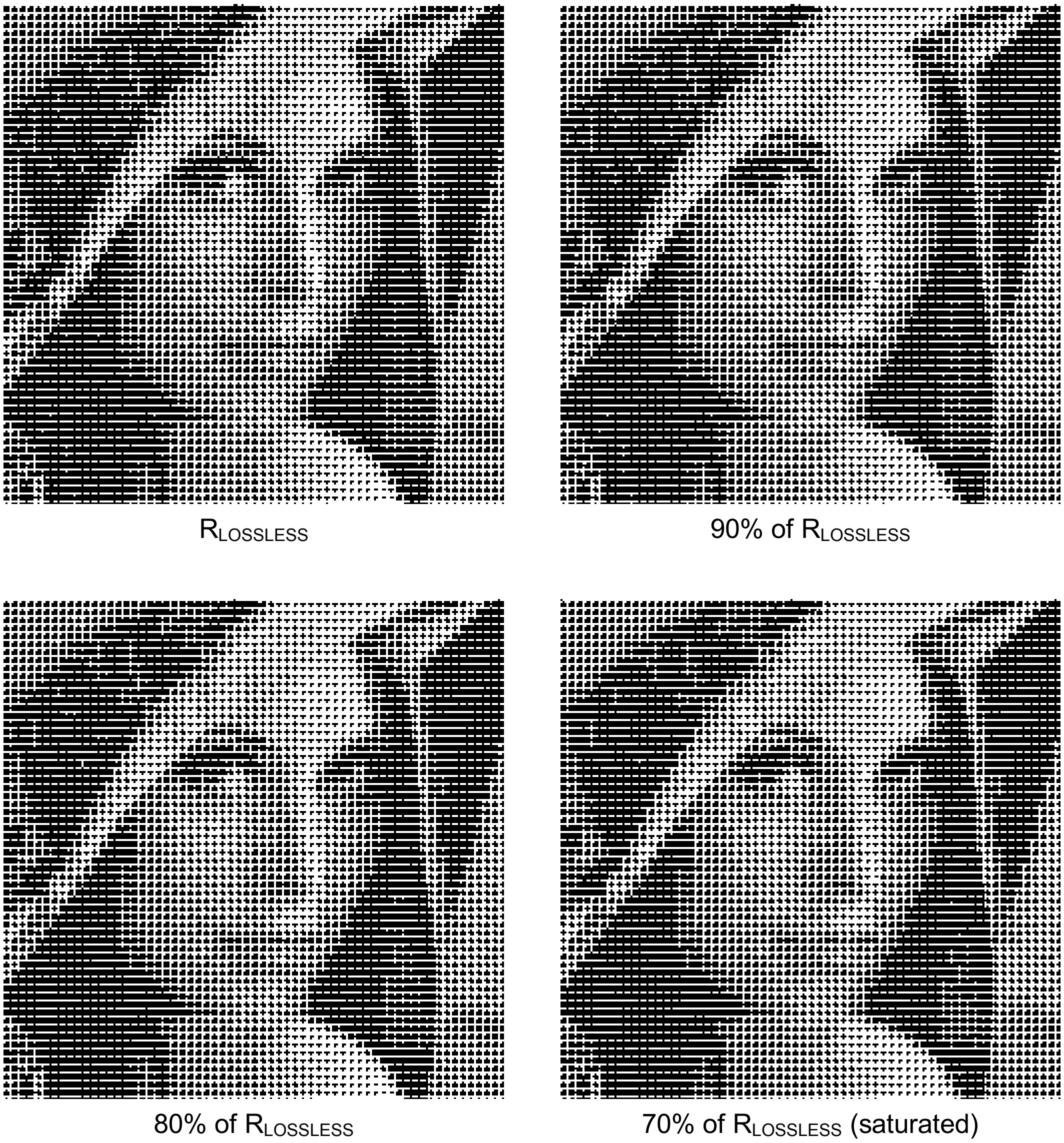


Figure 5-3 Rate-Distortion Curve for Generic Bit-Flipping

From the above graph, the general shape of all bit-flipping rate-distortion curves can be seen. Starting with the undistorted image, some bits can be immediately flipped to achieve some modest distortion gain. Gradually, the rate gain achieved by flipping an individual bit decreases, until large increases in distortion are introduced for very small rate gains. Eventually, a point is reached where there are no bits that can be flipped to improve the rate, which is known as the *saturation* point. To decrease the rate past the saturation point, then a higher-ordered analysis must be conducted, which considers the consequences of flipping two or more bits simultaneously. This higher-ordered approach increases the complexity considerably, and has not yet been attempted in experiments.

The original MFBF method was proposed for halftoned images, so it would be expected that it would perform well on those types of images. Figure 5-4 illustrates the results on the image Small_Lena. The image at 90% of the lossless rate seems to be perceptually lossless, but the saturated image clearly shows artifacts, which appear as contours.



$R_{LOSSLESS}$

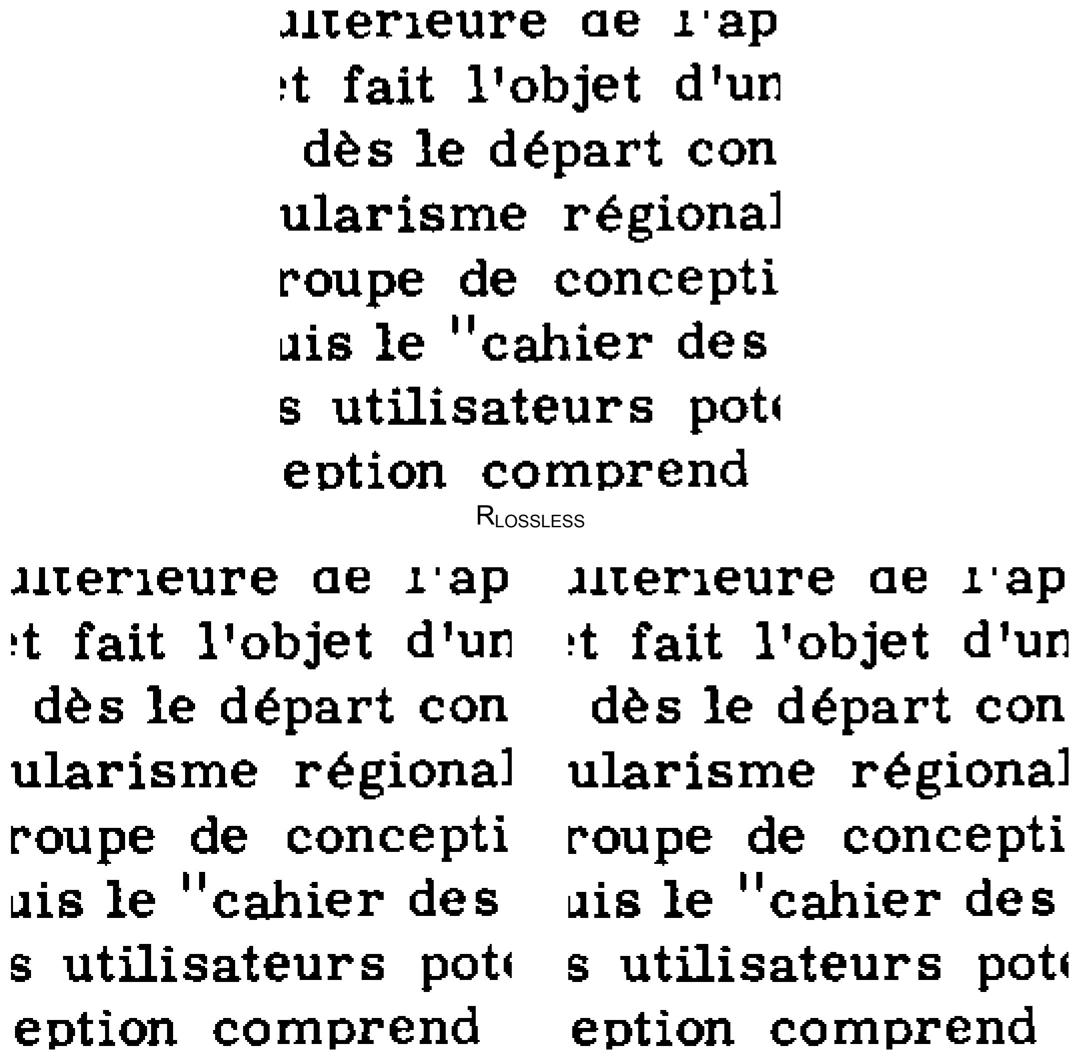
90% of $R_{LOSSLESS}$

80% of $R_{LOSSLESS}$

70% of $R_{LOSSLESS}$ (saturated)

Figure 5-4 Rate-Controlled Generic Bit-Flipping on Small_Lena

Although bit-flipping methods have never claimed to achieve competitive lossy rates on textual images, it is interesting to see how the method performs, if for no other reason than to consider the consequences of incorrectly classifying an image. Figure 5-5 illustrates the results on the image Small_F04_200, which is a small region of a larger page of text. As it can be seen in the figure, bit-flipping will smooth out text. Reasonable rate improvements can be achieved, but they are not competitive with text based methods.



ılterıeure ae ı'ap
:t fait l'objet d'un
dès le départ con
ularisme régional
roupe de concepti
uis le "cahier des
s utilisateurs pot
eption comprend

$R_{LOSSLESS}$

ılterıeure ae ı'ap
:t fait l'objet d'un
dès le départ con
ularisme régional
roupe de concepti
uis le "cahier des
s utilisateurs pot
eption comprend

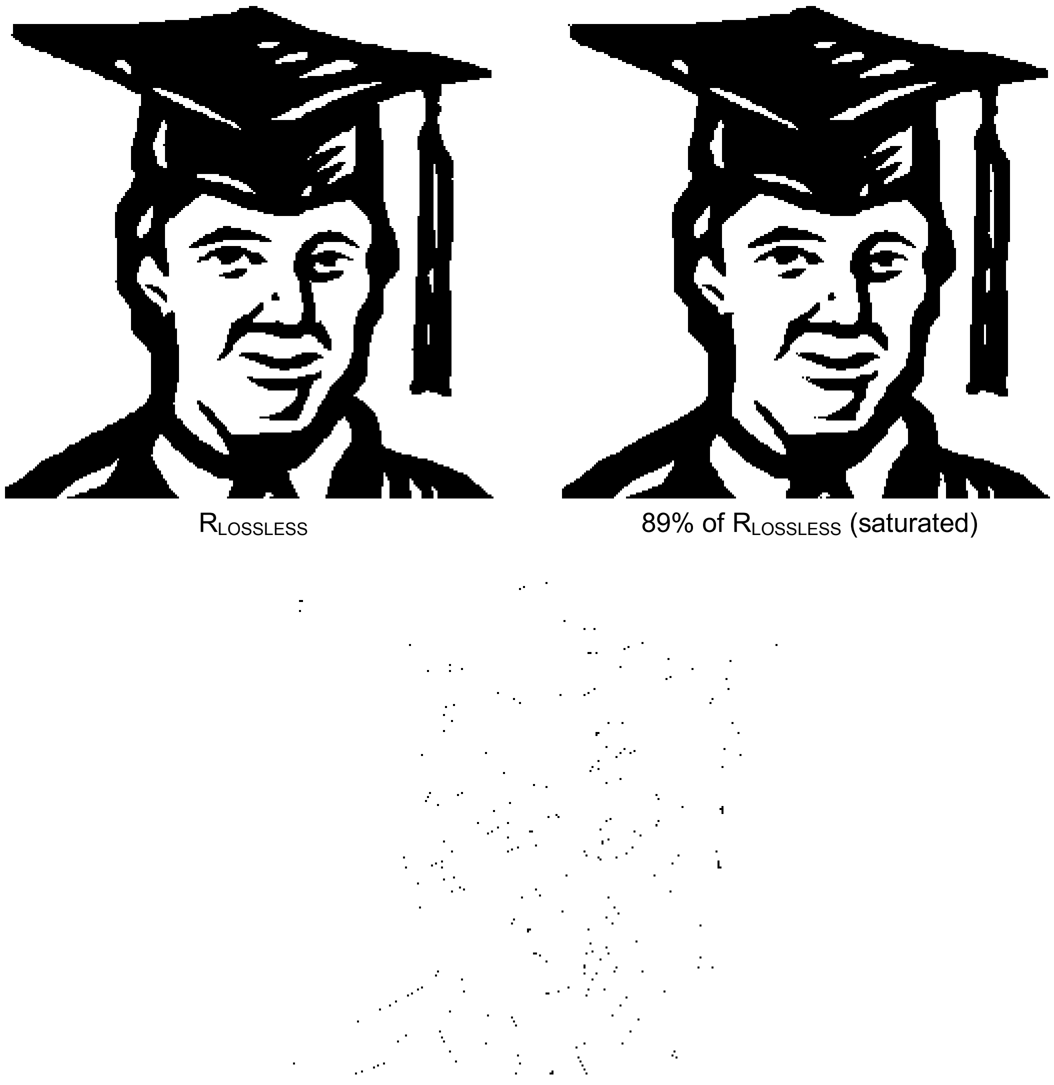
90% of $R_{LOSSLESS}$

ılterıeure ae ı'ap
:t fait l'objet d'un
dès le départ con
ularisme régional
roupe de concepti
uis le "cahier des
s utilisateurs pot
eption comprend

80% of $R_{LOSSLESS}$ (saturated)

Figure 5-5 Rate-Controlled Generic Bit-Flipping on Small_F04_200

The unfortunate irony of generic rate control is that it does not perform well on the images that require it the most. Typically, generic regions are used for line-art, graphs, charts and handwriting, all of which generate poor results when bit-flipping rate control is applied to them. Figure 5-6 illustrates the method applied to the image Graduate. From this figure, and the rate-distortion curve in Figure 5-3, it can be seen that there is little room available for improving the rate. In general, bit-flipping will clear up noise, as well as smooth and straighten out regions of the image. Since the changes to the image are subtle, the difference image (XOR) has been provided to show where bit-flipping has occurred.



$R_{LOSSLESS}$

89% of $R_{LOSSLESS}$ (saturated)

Flipped Pixel Locations

Figure 5-6 Rate-Controlled Generic Bit-Flipping on Graduate

Overall, generic region bit-flipping can achieve good results on a small operating range of the rate-distortion curve. The general method of  bit-flipping is based on the MFBF method, but numerous adjustments and enhancements must be made to make the method appropriate for real applications with the MQ coder. Eventually, the method reaches a saturation point, where no more compression can be achieved, and the more aggressive halftone coding method must be considered.

# CHAPTER 6. RATE CONTROL IN REFINEMENT REGIONS

In this chapter, a method is presented to control the rate of a JBIG2 refinement region. The proposed method is very accurate, and can be used to improve the quality of an image while meeting any specific rate constraints. This chapter is divided into three sections, starting with a background on rate controlled refinement regions, followed by the proposed method, and then the results.

## 6.1 BACKGROUND

Refinement coding was originally designed to achieve high rates in compressing textual information [Mohi84,Witt94a,Witt94b,Howa96,Howa97] but has since been applied to halftone coding [Mart99] and with the flexibility of the JBIG2 standard, can now be easily applied to any type of image [T88]. As discussed in Section 2.5.4, refinement coding codes a destination image while referencing a source image. The vast majority of the bits in a refinement section are spent coding the error pixels which are the difference between the source and destination images [Witt94a]. If the source and destination images are identical, then the resulting bitstream size will be small, consisting only of overhead bits, and the bits required to skew the context probabilities. If the destination is a lossless reconstruction, then the resulting bitstream will be considerably larger, and the size will roughly depend on the number of bits that are different in the two images. These two extremes in refinement coding provide the upper and lower bound on the size of a refinement region, and so the objective of rate controlled refinement coding is to achieve a rate that lies between these two bounds.

The task of achieving the desired rate can be reduced to the problem of deciding which error pixels in the destination image should be corrected. This seems to be identical to the bit-flipping situation encountered with generic regions in Chapter 5. Although there are many similarities to generic bit-flipping, there are some significant differences. Bit-flipping in generic coding is used to introduce errors, while with refinement coding, bit-flipping is used to fix existing errors, which significantly reduces the number of candidates for bit-flipping. As discussed in Section 5.1, there are numerous repercussions when a bit is flipped, making the analysis of bit-flipping quite complicated, especially with the MQ coder. While refinement bit-flipping can have similar repercussions, they are not nearly as serious as in the generic case. The contexts are smaller and the majority of those pixels are from the source image, which are unaffected by any bit-flipping, so the analysis of any repercussions becomes much simpler, and can be safely ignored.

## 6.2 PROPOSED METHOD

The elusive goal of generic bit-flipping is to achieve the desired rate with only one pass through the image. While a single pass algorithm seems unobtainable in generic coding, the proposed method for refinement bit-flipping can achieve this goal. As mentioned previously, the objective is to achieve the desired rate, $R_{TARGET}$, which is bounded between the rates achieved by flipping either all or none of the error bits. Figure 6-1 illustrates the basic rate-distortion curve for refinement coding. There is the maximum amount of distortion, $D_{SOURCE}$, which corresponds to a refinement region that does not flip any error bits and has a rate of $R_{NOFLIP}$. If all of the error bits are flipped, the distortion becomes zero, and the rate $R_{LOSSLESS}$ is achieved.

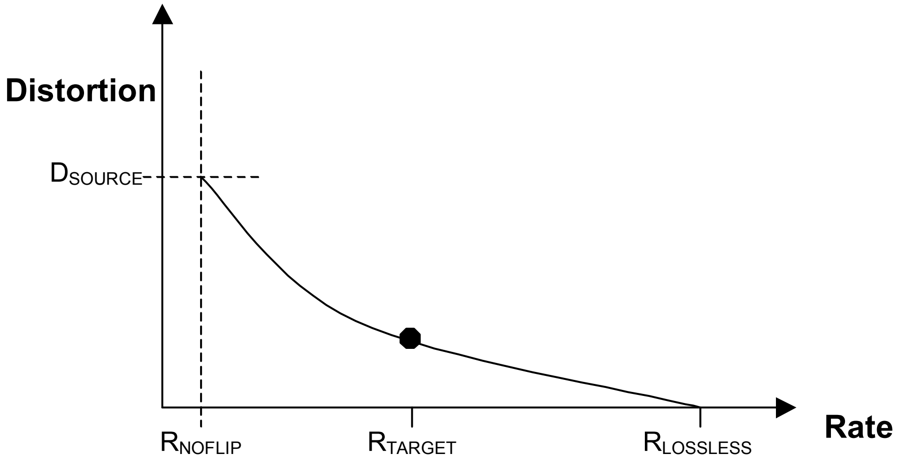


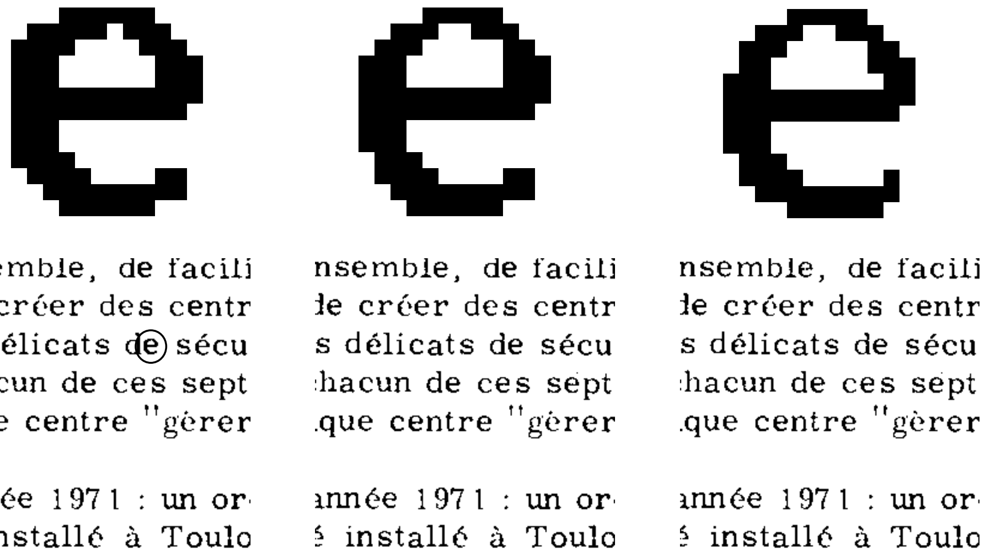
Figure 6-1 Basic Rate-Distortion Curve for Refinement Coding

An interesting technique that can be employed with refinement coding is that the refinement section can be prematurely truncated. That is, the refinement section does not have to complete the image. Since it is a refinement, the remainder of the image will simply be the same as the source image. To implement this, the buffer of the MQ coder from the previous row is saved, so that the coder can backtrack and finish the segment at that row if the subsequent row exceeds the target rate. With this technique, it becomes possible to code the refinement with a smaller rate than $R_{NOFLIP}$. This technique also provides a simple solution to one of the difficulties of one-pass coding methods. With one-pass coding methods, the data at the bottom of the image may not cost as many bits as expected, resulting in a final rate that is lower than $R_{TARGET}$. However, with this truncation technique, if a rate is chosen that is sufficiently larger than the target rate, ($R_{TARGET} + R_{OFFSET}$) then the target rate will be reached prematurely, and the segment can be truncated accordingly. By using this truncation technique, the rate control method is very effective at achieving the exact target rate, but has the disadvantage of fixing distortion in an uneven way. In particular, the distortion at the top of the image will be much smaller than the distortion at the bottom of the image. The consequences of this unbalanced distortion will depend on the distortion measure.

With the use of the truncation technique, the proposed method is actually quite simple. The target bitrate is increased by the value $R_{OFFSET}$. The refinement region is coded normally for all non-error bits, and a simple test is performed for every error bit to determine if it should be corrected or not. The error bit is corrected if the current rate is below the target rate. If at any time the total refinement size exceeds the target size, then the refinement segment is truncated.

## 6.3 RESULTS

The proposed method is very effective at achieving the target rate for a refinement segment, and can obtain the target rate within a few bytes. In order for this method to work properly, the target rate has to be artificially increased by some value, $R_{OFFSET}$. The choice of $R_{OFFSET}$ is somewhat arbitrary, and will only affect the row of the image at which the segment will be truncated. The value of $R_{OFFSET}$ can be chosen as a constant, such as 100 bytes, or as a proportion of the target rate, $R_{TARGET}$. In Figure 6-2, the results of choosing a very large value of $R_{OFFSET}$ can be seen, and although the images may be perceptually lossless, there is more distortion in the bottom of the refined image than the top. Small sections of the images in Figure 6-2 have been enlarged to illustrate the partial refinement that has occurred.



Original Lossy

Partial Refinement:
$R_{TARGET} = R_{LOSSLESS} / 2$
$(R_{OFFSET} = 2 * R_{TARGET})$

Lossless

Figure 6-2 Rate Control in a Refinement Segment

# CHAPTER 7. RATE CONTROL IN TEXT REGIONS

In this chapter, a method is proposed for controlling the rate of a JBIG2 coded text region. First, an overview of the method is provided, which consists of two stages. Next, the first stage, which determines the subset of symbols that will produce the smallest rate for a given distortion level is described. The third section describes how the target rate is achieved in the second stage. The fourth and fifth sections describe some implementation details and possible enhancements, respectively. In the final section, the results are presented.

## 7.1 BACKGROUND

There have been numerous text compression methods, but there are two primary methods of performing text compression in JBIG2 [Asch74, John83, Mohi84, Silv84, Howa96, Witt97a, Howa97, Howa98]. The first method, known as combined symbol matching (CSM), is occasionally referred to as hard pattern matching, and is inherently a lossy compression method. With CSM, if two symbols (characters) are considered similar, one of them is replaced by the other throughout the document. With CSM, several symbols may be replaced with a single representative symbol. The alternative to CSM, known as soft pattern matching (SPM), can be used as both a lossy and lossless compression method. With SPM, if two symbols are similar, than refinement coding is used to losslessly code the second symbol with a reference to the first symbol. Traditionally, some lossy pre-filtering is used to improve the performance of the SPM method.

## 7.2 OVERVIEW

The proposed method for controlling the rate of a JBIG2 coded text region consists of two stages. The aim of the first stage is to determine the set of symbols that will have the smallest rate for any level of distortion. This is achieved by starting with the complete set of symbols, and then matching two symbols, combining them to become a single symbol. This procedure is repeated until the desired level of distortion is achieved or there are no more feasible pairs of symbols to match.

Once the full range of symbol sets has been found, the rates for a few select distortion levels are measured to obtain a rough estimate of the rate-distortion curve. The second stage of the proposed method depends on where the target rate falls on the curve, and involves either converging to the desired rate, or overestimating the distortion and then refining the image to the desired rate.

As discussed previously, there are two primary strategies for coding text regions, known as CSM and SPM. In the proposed method, the strategy most appropriate for the desired bitrate is chosen. However, instead of using a pre-filtering method to achieve lossy SPM, lossy SPM results are found by compressing the equivalent image that would have been generated with CSM. Figure 7-1 illustrates the general rate-distortion curves for a typical text image.

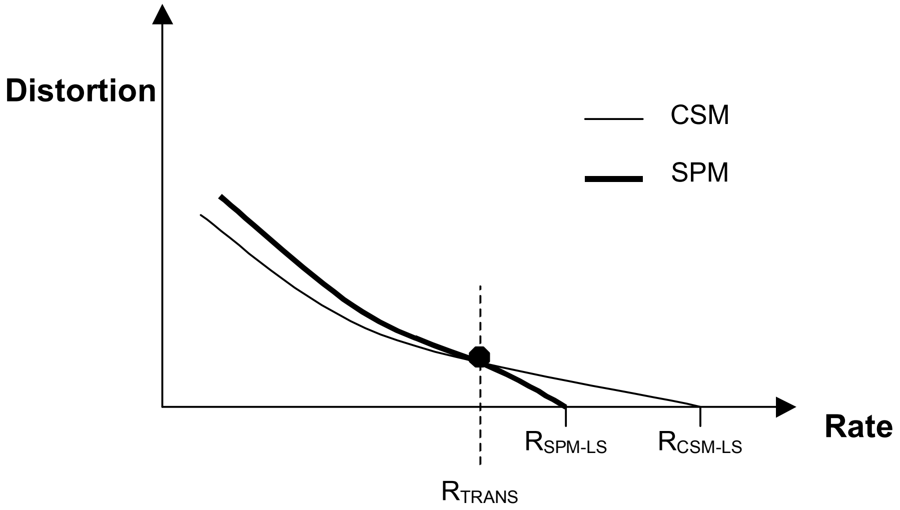


Figure 7-1 General Shape of a Text Region Rate-Distortion Curve

The SPM approach was designed for text regions that contain a very large number of similar symbols, suitable for refinement coding. If there are only a few similar symbols then the SPM approach can actually decrease the compression performance. There is a small amount of overhead introduced when using refinement coding, but the decrease in performance is mostly because there is not enough information to make good predictions in the refinement data.

An overview of the proposed method is illustrated in the flow diagram of Figure 7-2.

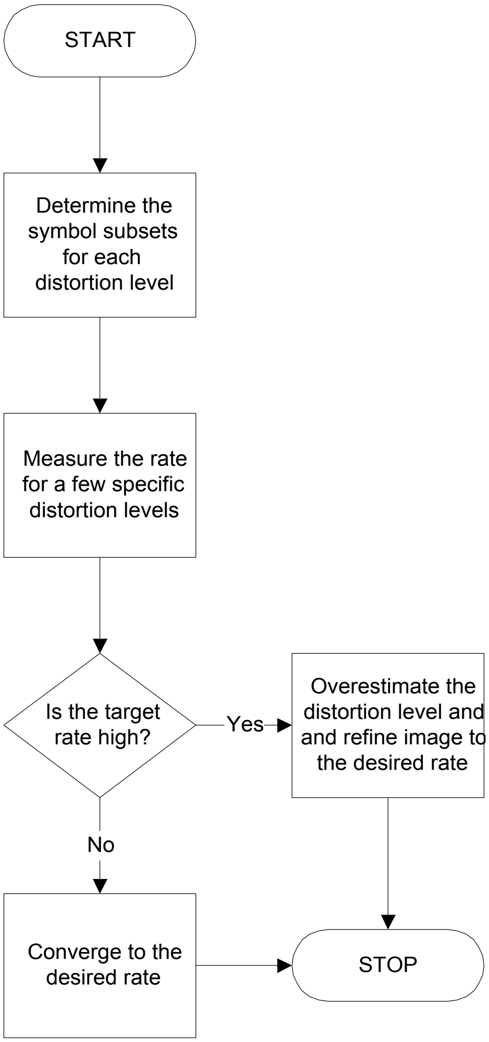

START
Determine the symbol subsets for each distortion level
Measure the rate for a few specific distortion levels
Is the target rate high?
Yes
No
Overestimate the distortion level and and refine image to the desired rate
Converge to the desired rate
STOP

Figure 7-2 Overview of Text Rate-Control Method

## 7.3 FINDING SYMBOL SUBSETS

Every text region contains a number *(N)* of unique symbols. At the simplest level, distortion occurs when the region is represented with a fewer than N symbols. In the proposed method, the lossy image is represented with a subset of the N symbols, where an optimal subset of symbols for every possible set size is determined.

The proposed method is independent of the distortion measure, but requires a distance function d(a,b) which measures the distortion resulting from substituting one symbol (a) with another (b). For example, if the Hamming distortion is used, then the distance function is simply the Hamming distance. A *weighted* distance function $d_w(a,b)$ is defined as:

$$d_w(a,b) = count(a) \cdot d(a,b),$$

where count(a) is the number of times that the unique symbol a occurs in the text region. The weighted distance function represents the distortion that would occur by replacing all instances of the symbol a with the symbol b. The function $d_w$ is not symmetric, as $d_w(a,b)$ does not necessarily equal $d_w(b,a)$.

The set distance function g(A) is defined as:

$$A = \{a_1, a_2, a_3 \ldots a_M\}$$
$$g(A) = \min\left(\sum_{j=1}^{M} d_w(a_j, a_1), \sum_{j=1}^{M} d_w(a_j, a_2) \ldots \sum_{j=1}^{M} d_w(a_j, a_M)\right)$$

The set distance function measures the minimal total distortion that results when all of the symbols in the set are replaced with one of the symbols from the set. The symbol which would generate the minimal distortion level is known as the *representative symbol* of all the other symbols in the set.

An incremental set distance function h(A,B) is required to measure the increase in distortion that occurs when sets A and B are combined, and is defined as follows:

$$h(A,B) = g(A \cup B) - g(A) - g(B).$$

In the proposed method,  the total distortion value is D, and there are N unique symbols. The value $D_M$ is defined as the minimal distortion level achieved with M unique symbols, and therefore $D_N = 0$. To start, N sets are created, with one unique symbol in each set. Then the iterative process begins, finding the two sets (A,B) that have the smallest incremental set distance h(A,B).  Two sets are combined to make one larger set, reducing the remaining number of sets (M) by one.  At this iteration, the image is represented with M unique symbols, one from each set.  The incremental set distance h(A,B) is added to D, which now has the value of $D_M$.  This iterative step can then be repeated until an exit condition occurs, which could be a maximum distortion level, a minimum number of sets, or until M is one.

To construct the image at a specific distortion level ($D_T$) the contents of all of the T sets must be known.  Once the T sets are known, all of the symbols in the image are replaced by their representative symbol.  To make the image construction process more effective, it is useful to record at each stage in the iteration the value of $D_M$ and sufficient information to reconstruct the M sets.  This collection of D values and set information is referred to as the *distortion log*.

The proposed method of for finding the symbol subset for each level of distortion is illustrated in the flow diagram of Figure 7-3.

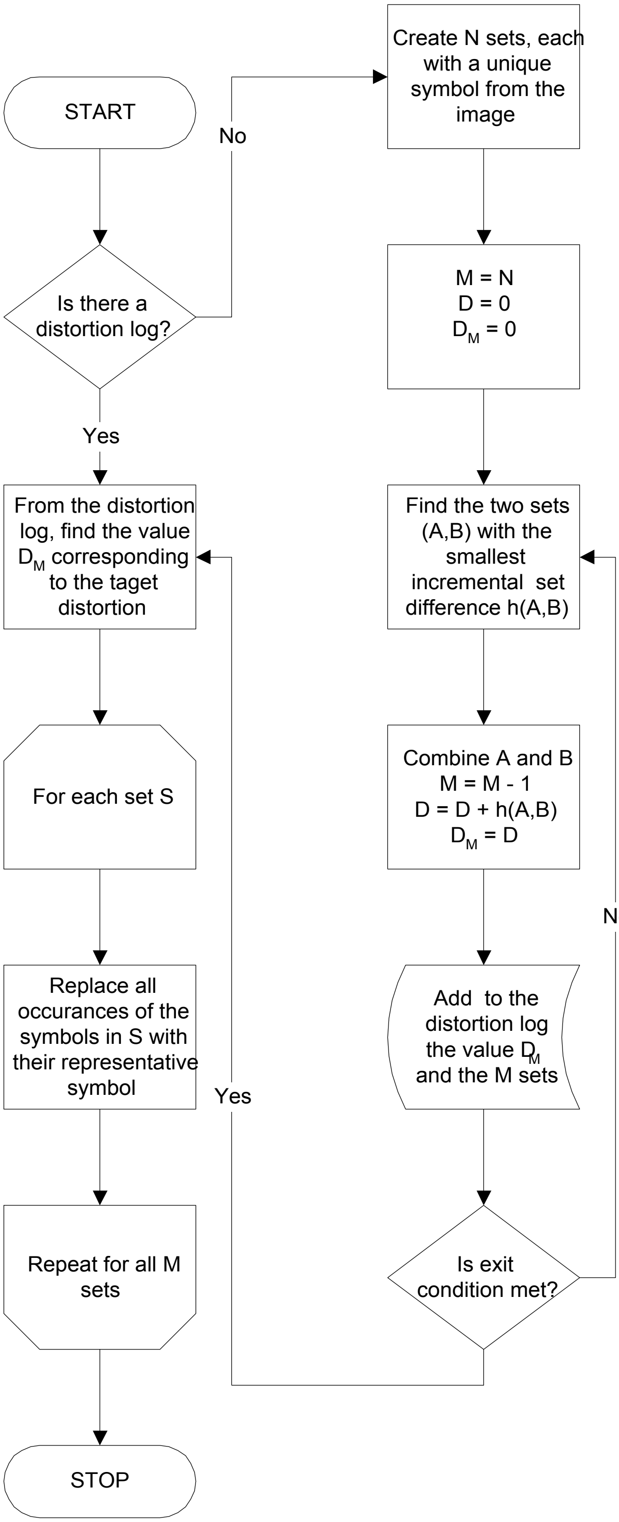

START
Is there a distortion log?
No
Yes
Create N sets, each with a unique symbol from the image
M = N
D = 0
$D_M$ = 0
From the distortion log, find the value $D_M$ corresponding to the taget distortion
Find the two sets (A,B) with the smallest incremental set difference h(A,B)
For each set S
Combine A and B
M = M - 1
D = D + h(A,B)
$D_M$ = D
No
Replace all occurances of the symbols in S with their representative symbol
Yes
Add to the distortion log the value $D_M$ and the M sets
Repeat for all M sets
Is exit condition met?
STOP

Figure 7-3 Finding the Symbol Subset for a Given Level of Distortion

## 7.4 ACHIEVING THE TARGET RATE

After the symbol subsets for all levels of distortion have been determined, the second stage of the proposed method begins, which achieves the target rate. The exact distortion can be found exhaustively by coding each image with both CSM and SPM until the desired rate is achieved. However, this approach would be impractical for many implementations, especially since SPM coding can be very expensive. The proposed method first determines the rates for a few select distortion levels, and then uses these rates to estimate the distortion level of the desired rate.

The method is based on the general shapes of the curves illustrated in Figure 7-1. First, the lossless compression rates $R_{SPM-LS}$ and $R_{CSM-LS}$ are determined. If either of these rates are below the target rate, then the solution is trivial. If the two rates are very close (i.e.: within 2%) then the image has very few similar symbols. In this case only the CSM (lower bitrate) approach is considered. Moreover, if the target rate is much lower than the lossless rates (i.e.: < 50%) then only the CSM approach needs to be considered.

The next step in the method is to estimate the transition rate $R_{TRANS}$ which represents the rate at which both CSM and SPM have the same level of distortion. To find $R_{TRANS}$, the distortion level $D_{k*N}$ is found, where the constant *k* should be high, with a typical value being 0.9. Next, the rates for SPM and CSM that correspond to the distortion level $D_{K*N}$ are found, labeled $R_{SPM-K}$ and $R_{CSM-K}$ respectively. The value of $R_{TRANS}$ is estimated with the following formula

$$R_{TRANS} = \frac{R_{CSM-LS} \cdot R_{SPM-K} - R_{SPM-LS} \cdot R_{CSM-K}}{R_{CSM-LS} + R_{SPM-K} - R_{SPM-LS} - R_{CSM-K}},$$

which is illustrated in Figure 7-4. If the target rate is above $R_{TRANS}$ the SPM approach is used, otherwise, the CSM approach is used.

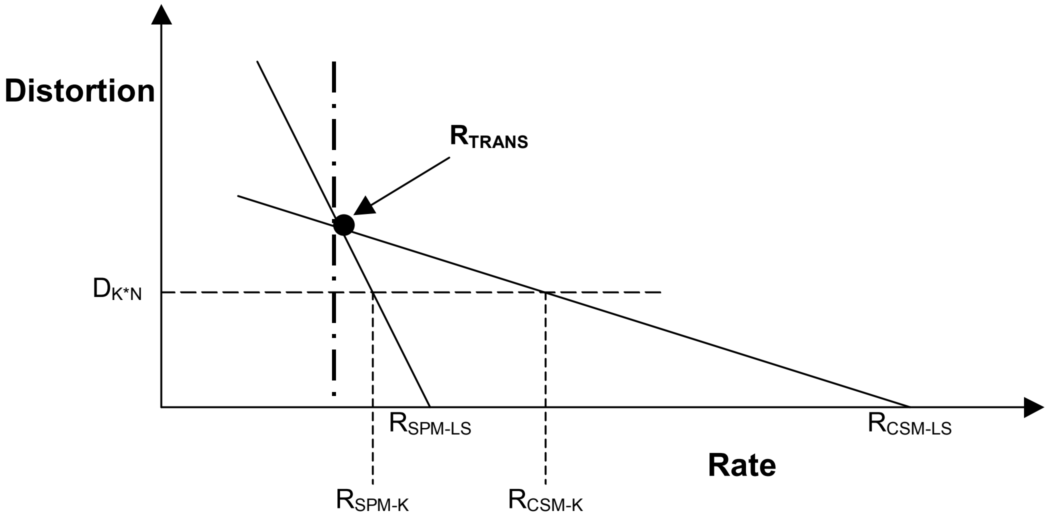


Figure 7-4 Estimating the value of $R_{TRANS}$

Since coding with SPM can be computationally expensive, calculating numerous SPM rates should be avoided. Therefore, the distortion level is intentionally overestimated, and then refinement coding can be applied to achieve the target rate. This is achieved by multiplying the target rate by some constant value $k_{SPM}$, where a typical value of $k_{SPM}$ would be 0.95. To find the target distortion level, linear extrapolation is used again, using the reference values $R_{SPM-LS}$ and $R_{SPM-K}$:

$$D_{TARGET} = D_K \frac{R_{SPM-LS} - k_{SPM} \cdot R_{TARGET}}{R_{SPM-LS} - R_{SPM-K}}$$

After calculating the distortion, the SPM rate is found. If the rate is below the target rate, then refinement coding can be used to code the difference. If the rate is too high, the value of $k_{SPM}$ is reduced and the distortion level re-calculated. This approach may not achieve the optimal distortion, since the refinement will reduce distortion in sub-optimal areas, but since the size of the refinement region will be small, there would only be a small penalty.

Coding a text region with CSM is fast, and so convergence to the target rate is feasible. For the CSM approach, the rate-distortion curve is approximated with a quadratic equation, fitting to three distinct (R,D) points. In the first iteration, only one point has been determined, so any two additional points are required. The rates corresponding to the distortion values of $D_{0.8*N}$ and $D_{0.6*N}$ are measured. Once three (R,D) points on the curve are available, the estimated distortion for the target rate is found using the following Lagrange formula:

$$D = D_1 \frac{(R_T - R_2)\cdot(R_T - R_3)}{(R_1 - R_2)\cdot(R_1 - R_3)} + D_2 \frac{(R_T - R_1)\cdot(R_T - R_3)}{(R_2 - R_1)\cdot(R_2 - R_3)} + D_3 \frac{(R_T - R_1)\cdot(R_T - R_2)}{(R_3 - R_1)\cdot(R_3 - R_2)}$$

The rate at the estimated level of distortion is then determined. If the resulting rate is too high or not within some threshold, then the target rate is re-calculated, using this new (R,D) pair and ignoring the pair that is farthest from the target. This process is repeated until the target rate is achieved. If the target rate is very low, then it may be impossible to use a text region to represent the image. Under those circumstances, it would be necessary to use the methods described in Chapter 3.

The overall method for achieving the target rate is illustrated in Figure 7-5.

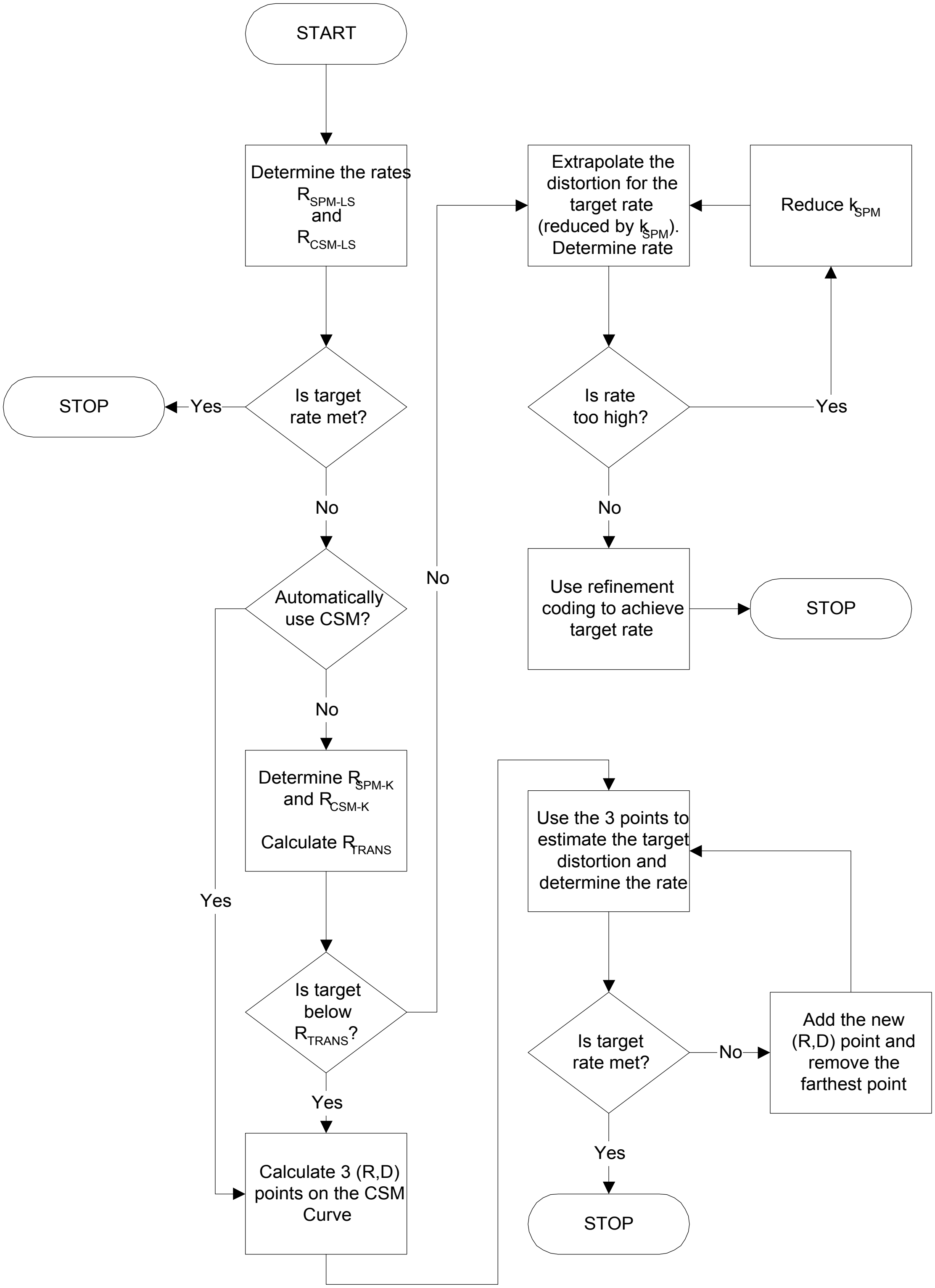

START
Determine the rates $R_{SPM-LS}$ and $R_{CSM-LS}$
Is target rate met?
Yes
STOP
No
Automatically use CSM?
No
Yes
Determine $R_{SPM-K}$ and $R_{CSM-K}$
Calculate $R_{TRANS}$
Is target below $R_{TRANS}$?
Yes
No
Calculate 3 (R,D) points on the CSM Curve
Extrapolate the distortion for the target rate (reduced by $k_{SPM}$). Determine rate
Reduce $k_{SPM}$
Is rate too high?
Yes
No
Use refinement coding to achieve target rate
STOP
Use the 3 points to estimate the target distortion and determine the rate
Is target rate met?
No
Add the new (R,D) point and remove the farthest point
Yes
STOP

Figure 7-5 Achieving the Target Rate

## 7.5 IMPLEMENTATION CONSIDERATIONS

The proposed method finds the image with a minimal amount of distortion for a given rate. To implement the method, there are some procedures that can be computationally expensive. In this section, numerous techniques to reduce the complexity of the proposed method are introduced, with some possible penalties in distortion.

### 7.5.1 PARTITIONING THE SYMBOLS

The proposed method requires that for every iteration in the first stage, the minimal incremental set distance h(A,B) be found. By narrowing the search, the time required to find the smallest h(A,B) can be significantly reduced, at the risk of selecting the wrong two sets. At the start of the method, the symbols are placed into different partitions, based on some partitioning criterion. The partitioning criterion should be designed to place symbols with small distances in the same partition, without actually measuring their distance. This is similar to the procedure of screening in [Witt97a], which contains some excellent screening procedures. While iterating, only sets from the same partition are used to calculate the incremental set difference.

### 7.5.2 STORING $D_W$ VALUES

In the very first iteration of the first stage, the weighted distance $d_w$ between all symbols is calculated. Even with simple distortion measures, calculating $d_w$ can be very expensive. By storing those values, all subsequent iterations can be executed faster. The memory required to store all of the $d_w$ values is very large ($O(N^2)$) and may be impractical. However, a partitioning strategy from Section 7.5.1 can be implemented to bound the memory requirements.

### 7.5.3 ESTIMATING THE VALUE OF H(A,B)

To properly calculate the incremental set distance h(A,B) the set distances g(A), g(B) and g(A∪B) are calculated. In any implementation, the set distance of each set can be stored, requiring only the calculation of g(A∪B). The set distance g(A∪B) finds the minimal distortion required to replace all of the symbols in A∪B with one of the symbols in A∪B, which can be very expensive to calculate. Instead of calculating g(A∪B), it can be estimated using the representative symbols of A and B. If the representative symbols of A and B are $a_r$ and $b_r$, respectively, h(A,B) can be estimated as

$$A = \{a_1, a_2, a_3 \ldots a_M\}$$
$$B = \{b_1, b_2, b_3 \ldots b_L\}$$
$$h(A,B) \cong \min\left(\sum_{j=1}^{M} d_w(a_j, b_r), \sum_{j=1}^{L} d_w(b_j, a_r)\right).$$

In other words, the additional distortion introduced by replacing all of the symbols in one set with a representative symbol from the other set. To further reduce complexity, the representative symbol from the larger of the two sets can be used as the representative symbol for the combined set, instead of re-calculating a new one.

### 7.5.4 FAST REGION CODING

In the proposed method, the rates of coded regions for varying numbers of symbols are measured. The rates will depend on both the fixed characteristics of the image and the variable number of symbols (dictionary size). Most of the fixed costs are segment headers and the positional data of the text region, while the variable costs are in the dictionary segment and the dictionary IDs of the text region. The positional data in the text region is especially fixed if the symbols are partitioned by size as in Section 7.5.1. When measuring the rate for a given dictionary size, it can be approximated by only measuring the variable costs. Although this method may not be completely accurate, it is sufficient for interpolating and converging to a target rate, and can be replaced with a full rate measurement for the last few iterations.

### 7.5.5 POSTPONING SPM MEASUREMENTS

In the proposed method, there are at most three SPM rate calculations, assuming that the $k_{SPM}$ value is not set too high. With a small amount of additional distortion, it is possible to reduce that number to one. If the SPM rate calculations are ignored and a CSM approach is adopted, then the SPM rate can be measured only after the image for the CSM approach has been obtained. At that point, if the SPM rate is smaller than the CSM rate, then the SPM coded data is used and the image is refined until the desired rate is achieved.

## 7.6 ADDITIONAL ENHANCEMENTS

There are some additional enhancements that could be added to an implementation of the proposed method. These enhancements would change some of the characteristics of the rate-distortion curve, and would increase the complexity of the method.

### 7.6.1 FLYSPECKS

In the proposed method of finding distortion levels, all of the unique symbols in an image are found. The unique symbols are then combined together to form sets of symbols. However, the possible distortion that would be introduced by removing a symbol, or a set of symbols, from the image is not considered. In other words, the distortion that would be introduced by replacing a set of symbols with a null set. This is often referred to as *flyspeck* removal, as it often removes very small symbols on the page that look like flyspecks.

One consideration of flyspeck removal is that the particular distortion measurement may not have a deterministic value of $d(a,\varnothing)$. Assuming that the distortion is measurable, at certain distortion levels, the next higher level would require removing a set. This incremental change in distortion could cause a large change in the rate and introduce discontinuities in the rate-distortion curve. Furthermore, in the proposed method all identical symbols are grouped together as one unique symbol. Hence, there would be no change in rate if only one of those symbols were replaced with another symbol. However, when removing symbols, there is a potentially large difference between removing one or all of the identical symbols. Adding the additional detail required to consider the removal of individual symbols would significantly complicate the proposed method. It is for the above reasons that flyspeck removal was not included in the method. For lower to mid-range rates, it is much simpler to remove all flyspecks of a certain size before using the proposed method.

### 7.6.2 HYBRID REPRESENTATIVE CHARACTERS

In the proposed method, all members of a set are replaced with a representative symbol, which is a member of the set. However, there may exist another symbol that does not actually appear in the image which would make a better representative symbol. For example, a representative symbol generated by taking the average value of all the set members could produce a distortion lower than any individual member. Generating the optimal symbol would require detailed knowledge of the distortion measure, and may be an open-ended problem for some distortion measures. Introducing a hybrid symbol generator to the proposed method would not change the method significantly, and could be desirable in many applications.

## 7.7 RESULTS

The proposed method is quite effective at achieving the compression rate with a minimal amount of distortion. Since the method is independent of the distortion measure, it can be used in a wide variety of applications and have different performance requirements. To demonstrate the results of the proposed method, it was applied to the image F04_200 at various bitrates. In this experiment, the WXOR was used as the distortion measure, and the symbols were initially partitioned by size. The rate-distortion curves are plotted in Figure 7-6 and Figure 7-7. From these curves, the general shape that was illustrated in Figure 7-1 can be seen.

A selected region from the image F04_200 has been illustrated at several different bitrates in Figure 7-8. From this figure, substitution errors can be seen at the 50% bitrate, which become much worse at the 35% bitrate.

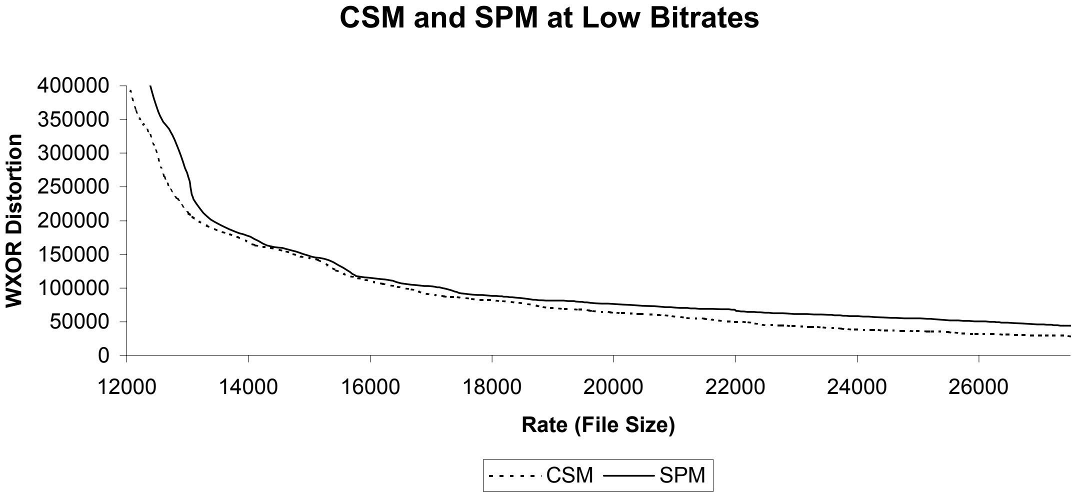


Figure 7-6 Rate-distortion Curve for Image F04_200 at Low Bitrates

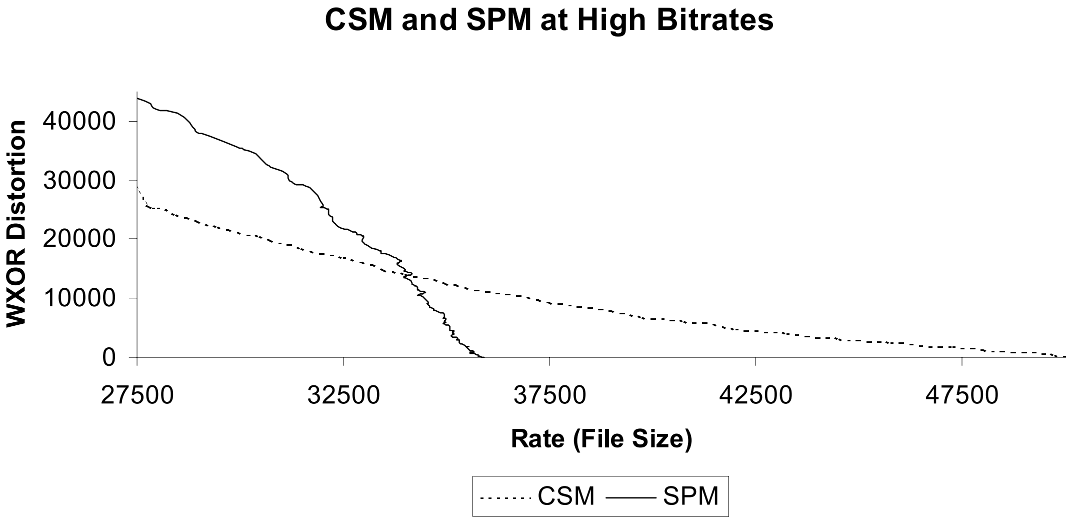
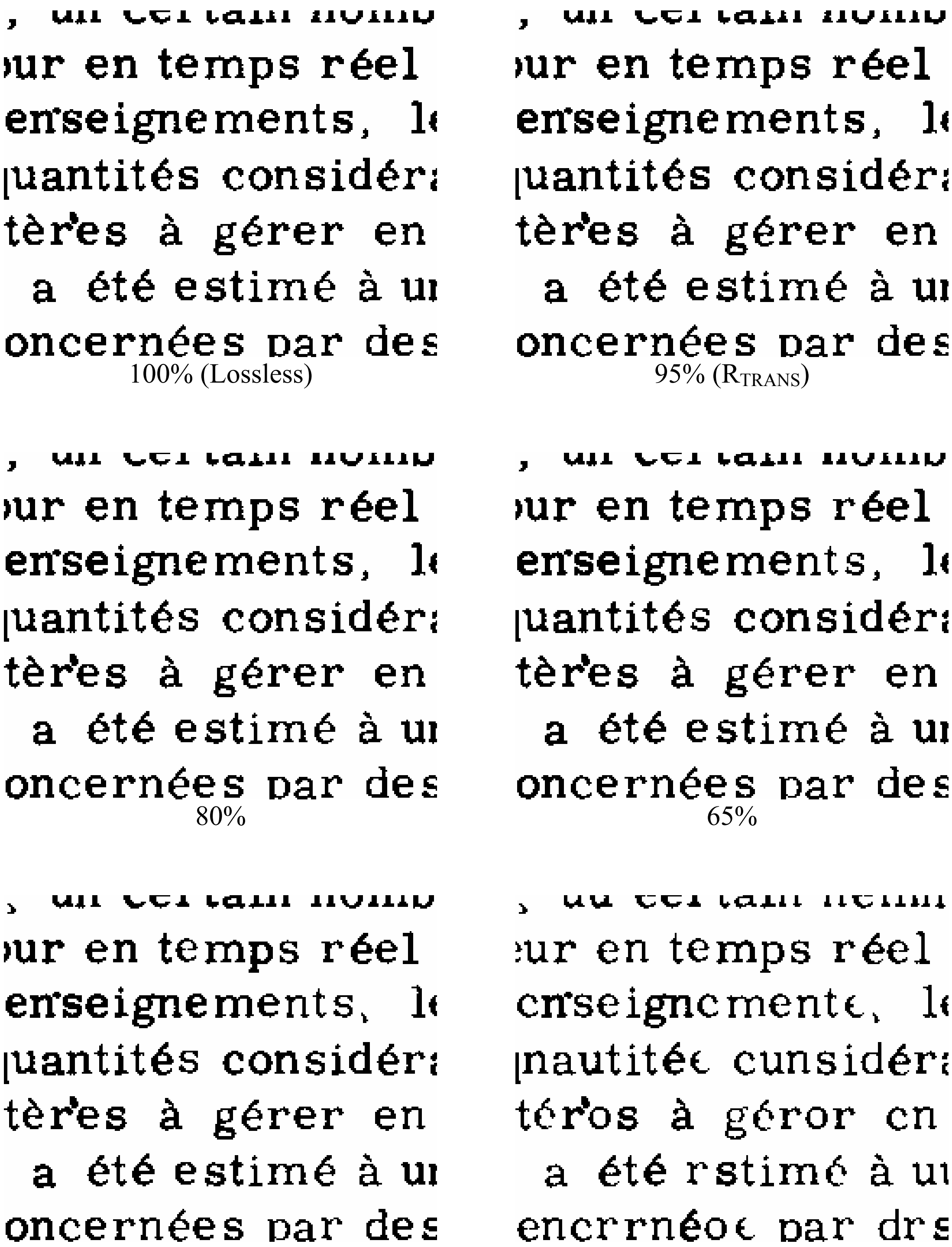


Figure 7-7 Rate-distortion Curve for Image F04_200 at High Bitrates



100% (Lossless)

95% ($R_{TRANS}$)

80%

65%

50%

35%

Figure 7-8 Lossy Images and Bitrates as a Percent of the Lossless Bitrate

# CHAPTER 8. RATE CONTROL IN HALFTONE REGIONS

In this chapter, a method is proposed to control the rate of a JBIG2 halftone region. In the first section, the general procedures for JBIG2 halftone coding are explained. In the next section, an overview of the proposed method is presented. In the following two sections, methods for higher and lower bitrates are explained. In the last section, results are presented.

## 8.1 HALFTONE CODING PROCEDURES

As discussed in Section 2.5.8, a JBIG2 halftone decoder receives a multi-level image and a pattern dictionary, and then constructs a halftone pattern for output. The multi-level image is generically coded as a sequence of bi-level images. To produce this information for a decoder, an encoder must generate a multi-level image and a halftone pattern. The input to a JBIG2 encoder may or may not be a halftone pattern. As suggested in Section 2.5.8, since halftone coding involves a spatial reduction, it may be the only viable JBIG2 compression method for very high bitrates. Additionally, it is possible that regions of an image may be incorrectly classified as a halftone.

The first step in performing JBIG2 halftone encoding is to generate the base multi-level image. This process of generating a multi-level image from a bi-level image is known as inverse halftoning, and the most appropriate method depends on the type of the halftone [Ulic87,Dame98,Kite98,Vall99,Mart00]. In the proposed method, one of two techniques is employed. First, a test is performed with an autocorrelation function to detect if there is any periodicity in the halftone, which would indicate an ordered halftone. If no ordered halftone is detected, then the method suggested in [Vall99] is chosen. If the image is an ordered halftone, then the method suggested in [Mart00] can be used to find the inverse halftone, with a specific grid size and a specific grid angle. Since the result from the method in [Mart00] is targeted for a specific grid size and angle, it is transformed so that it is of the same scale and orientation that would have been produced from the non-ordered method.

Once the base grayscale image has been obtained, the encoder has to choose four important halftone parameters: the patterns, the angle, the grid size, and the number of patterns. The patterns in the proposed method are always clustered dot, generated with a modified version of the method proposed in [Mart00]. The angle in the proposed method is always chosen as 45°, which as mentioned in Section 3.3 and in [Ulic87], produces the best results for a given spatial resolution. The choice of grid size and the number of patterns dramatically affects the compression rate, and is addressed in the subsequent sections.

## 8.2 OVERVIEW

The proposed method for controlling the rate of a halftone region is straightforward. In the first stage, the size of the pattern is chosen. In the second stage, the target rate is compared against the lossless rate for that pattern size, and is classified as either a high or low bitrate. For the third and final stage, either the high or low bitrate rate-control method is applied.

The first stage of the halftone coding method is to select the pattern size. In general, large pattern sizes will correspond to a greater spatial reduction, and more compression. The halftone pattern width is represented by $N$, and the pattern is of size $NxN$. With an unangled halftone pattern, a pattern of width $N$ will reduce the size of the image by $N^2$. In the proposed method, the halftone is angled at 45°, so the actual pattern width is $N/\sqrt{2}$, which reduces the size of the image by $N^2/2$. While increasing $N$ decreases the size of the image, it also increases the number of patterns required to represent all of the gray levels. An unangled $NxN$ pattern can represent $N^2+1$ different gray levels, while a 45° angle pattern can represent $N^2/2+1$ levels. In practice, it becomes unnecessary to represent more than 32 gray levels, as it becomes difficult to distinguish between them [Ulic87,Hunt95].

As indicated in Chapter 3, it can be very difficult to measure the distortion between two images that have different pattern widths, even under the rare conditions when the exact printing resolution and viewing distance are known. However, in general, the smaller the value of *N*, the better the image will appear. The proposed method to choose *N* is simply to find the smallest value of *N* for which the rate of the compressed image with pattern width *N* is larger than the target rate.

As indicated earlier, there are two different methods proposed for halftone rate-control, one for high rates and one for low rates. The general shape of the two rate-distortion curves are illustrated in Figure 8-1. The direction that the two methods operate in is shown in the figure. The high bitrate method starts at a pattern width of *N* and adds distortion until the rate is achieved. The low bitrate method starts at a pattern width of *2N* and improves the image quality until the rate is achieved. The point at which the two curves intersect is dependent upon the characteristics of the image, and the value of *N*. In the proposed method for rate control, the high or low method is chosen by a simple heuristic: if the target rate is above 66% of the rate, then the high rate method is chosen. As it will be demonstrated later, the high rate method will saturate, and so if this occurs, or starts to occur, then the algorithm can switch to the low rate method.

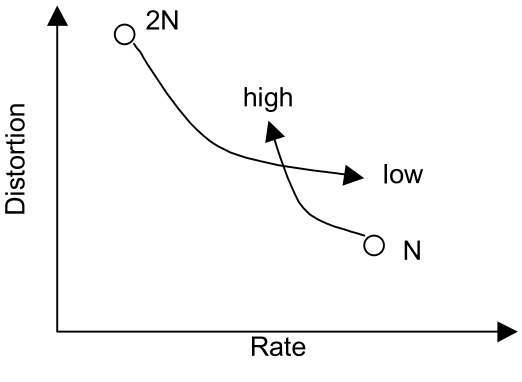


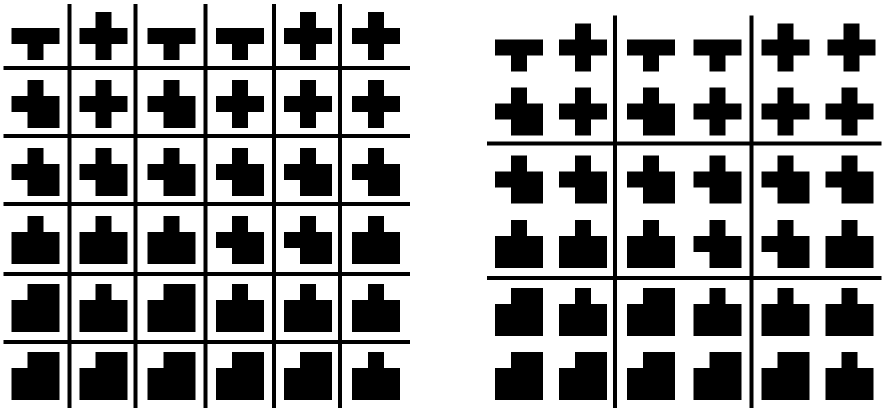
Figure 8-1 General Shapes of the two Halftone Rate-Distortion Curves

## 8.3 HIGH RATE APPROACH

The high-rate method of achieving rate-control is essentially the bit-flipping method that is proposed in Chapter 5. Since each plane in the multi-level image is generically coded, bit-flipping can be applied to reduce its' rate. The method must be modified slightly to flip bits across *N* different images, all of which share the same MQ context information. One of the problems with the methods described in Chapter 5, is that it is difficult to know how much distortion is being introduced by flipping an individual bit. However, while bit-flipping within the bitplanes, the amount of distortion can be calculated more easily, since flipping a bit does not change the image by one bit, but rather the grayscale value by one bit. Flipping a bit in a higher plane will change a bit in a more significant position, and will cause more distortion than a flip in a lower plane. However, the error calculation must consider that the planes are Gray-coded. For a pixel in bit plane *L,* the average MSE distortion introduced by flipping it will be $(4L^2-1)/3$, but for a particular number, it could be as low as one. In the proposed method, the average distortion value is used.

## 8.4 LOW RATE APPROACH

The proposed low rate method for achieving rate control in halftone regions uses JBIG2 in an unconventional manner. Traditionally, each pattern in the pattern dictionary corresponds to an indexed grayscale value, where the value of the index corresponds to the intensity of the pattern. However, there is no requirement in the JBIG2 standard that this is the case. In the proposed method, each pattern in the dictionary actually corresponds to a collection of four halftone patterns, as illustrated in Figure 8-2. In general, the low rate approach increases the spatial reduction of the grayscale image *(N)*, while increasing the number of patterns *(M)*.



Traditional Halftone Grid

Proposed Halftone Grid

Figure 8-2 Proposed Grid for JBIG2 Halftone Patterns

In the proposed method, the first G bitplanes correspond to an image coded with the grid size of *2N*, and $log_2G$ patterns. If the coding is stopped at this point, then each of the patterns will correspond to a mean value of gray, such as the pattern in the upper right-hand corner of Figure 8-2. At this point the rate of the image would be approximately the same as a traditionally coded image at a resolution of *2N*. However, the proposed method does not stop at this point, but rather continues to increase the number of patterns *(M)* and bitplanes $( L=\lceil log_2M \rceil)$ until the desired rate is achieved.

Each pattern is considered a *vector* of 4 smaller patterns, and so the method of choosing the M patterns for the image is known as Vector Quantization (VQ). There are numerous different methods of performing VQ [Gers91,Sayo96], and numerous solutions to this problem. In the proposed method, a VQ technique is chosen to coincide with the bitplane coding method of JBIG2. As mentioned previously, the first *G* bitplanes are fixed, and so the initial partitioning scheme is fixed. The vectors in the image are partitioned into $2^G$ groups, where each group corresponds to a mean gray level. In VQ terminology, this collection of groups is called the *codebook.* Although this may not be the optimal partitioning in the MSE sense, it is a fast method of producing the first $2^G$ groups, and has good compression performance. From this point onward, a tree structured VQ approach is taken, and each group is further partitioned to produce two vectors that are optimized in the MSE sense. This partitioning procedure is illustrated in Figure 8-2.

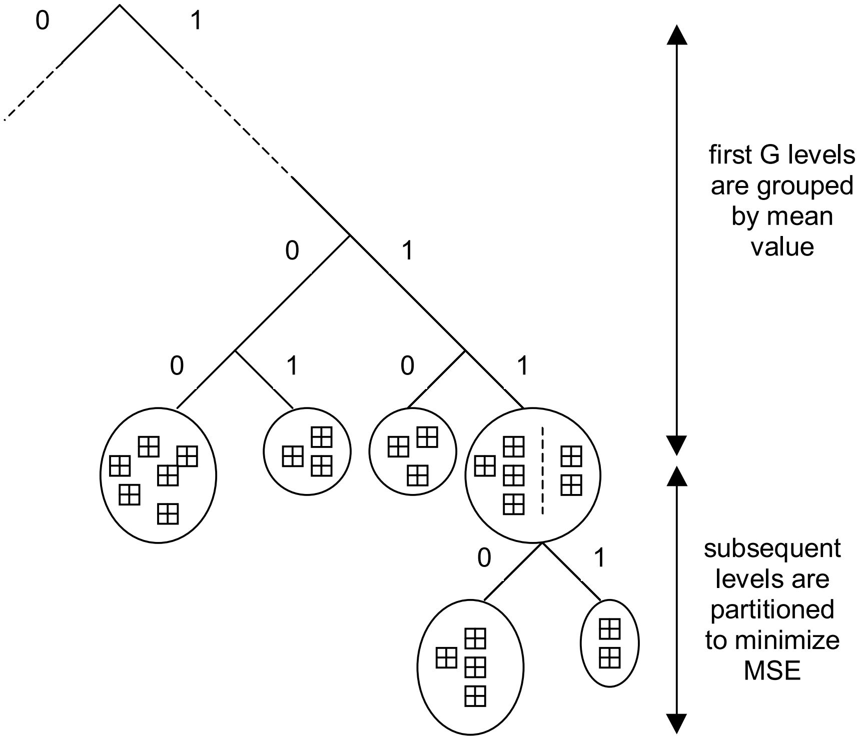

0
1
first G levels are grouped by mean value
0
1
0
1
0
1
0
1
subsequent levels are partitioned to minimize MSE

Figure 8-3 Partitioning the Pattern Vectors

The overview of the proposed low-rate method is as follows. First, the number of bitplanes *(L)* is set to *G*, and the resulting bitrate is found. If the target rate has not been surpassed, then the number of bitplanes is increased, and all of the leaves of the tree are partitioned. The new bitplane is coded, and if the target rate has not been surpassed, then the number of bitplanes continues to increase. Once the target rate has been exceeded, then the number of patterns *(M)* is reduced until the target rate has been achieved.

To perform the additional partitions, the Linde-Buzo-Gray (LBG) method [Lind80] is used. LBG requires a pair of initial vectors as a basis for partitioning the set. In the proposed method, the DCT of each vector is calculated, and the DCT coefficient with the largest variance is selected. The initial vectors for the LBG method are the current codebook value, perturbed in the direction of the selected DCT coefficient, but the final split is determined after several iterations of the LBG method. Initializing the partition along the DCT coefficient improves the edges of the halftoned image, and increases the visual quality. In the tree, the largest of the partitions is assigned "0", while the smallest is assigned "1", which improves the compression rate.

Once the target rate has been exceeded, the tree must be pruned until the target rate is reached. While pruning the tree, the vectors that reduce the MSE of the resulting image the least are pruned first. Technically, $2^L$ halftone patterns are included in the JBIG2 pattern dictionary, although not all of the patterns are referenced. This is done so that the previous *(L-1)* planes do not have to be re-coded, and to improve the compressibility of each plane. In the pattern dictionary, these unreferenced patterns are filled with the previous valid pattern.

## 8.5 RESULTS

This section presents the results of applying the proposed method to the image illustrated in Figure 8-4.

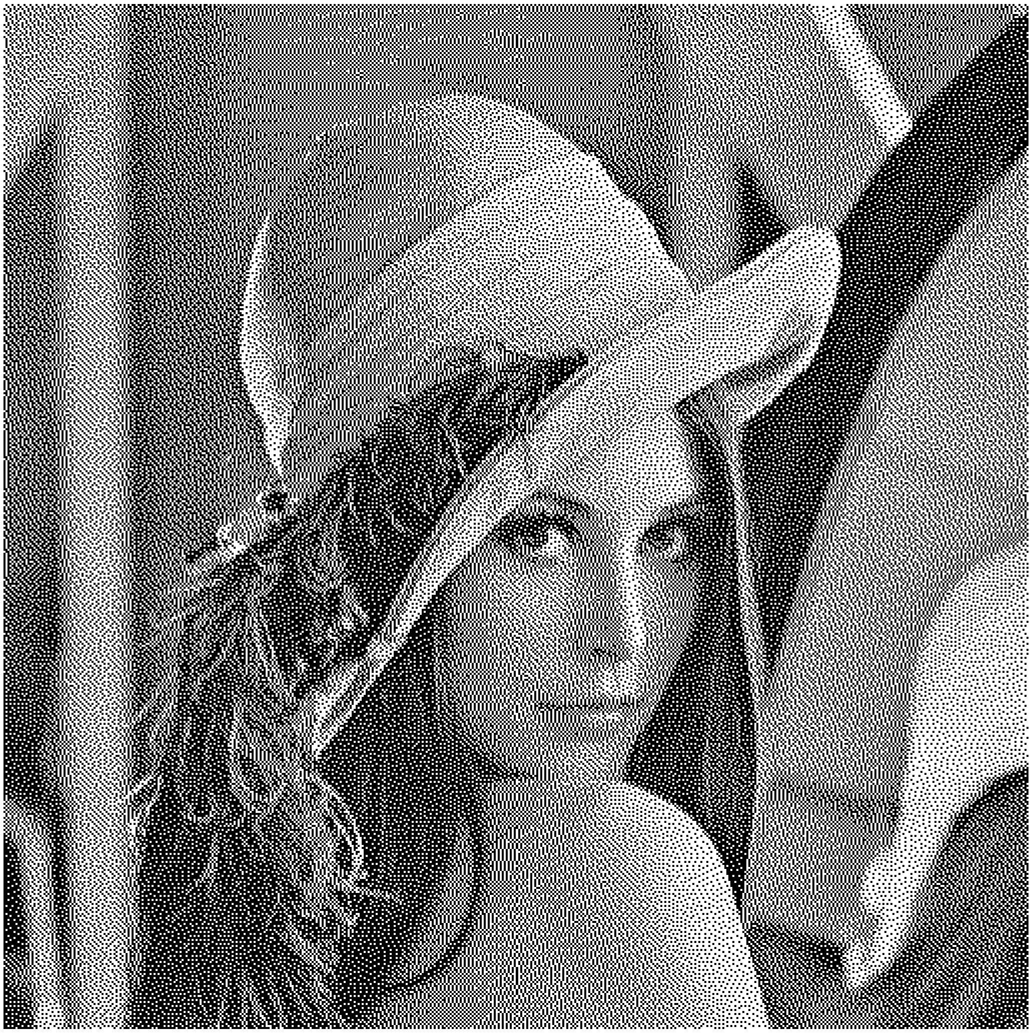


Figure 8-4 Lossless Lena Halftone

The value of *N* was chosen to be 6. The losslessly coded 6x6 halftone is presented in Figure 8-5. This is the image that is used as the starting point for the high rate halftone coding method. The equivalent grayscale image for this image was used as the reference for measuring MSE distortion, so the distortion for this image is considered to be 0.

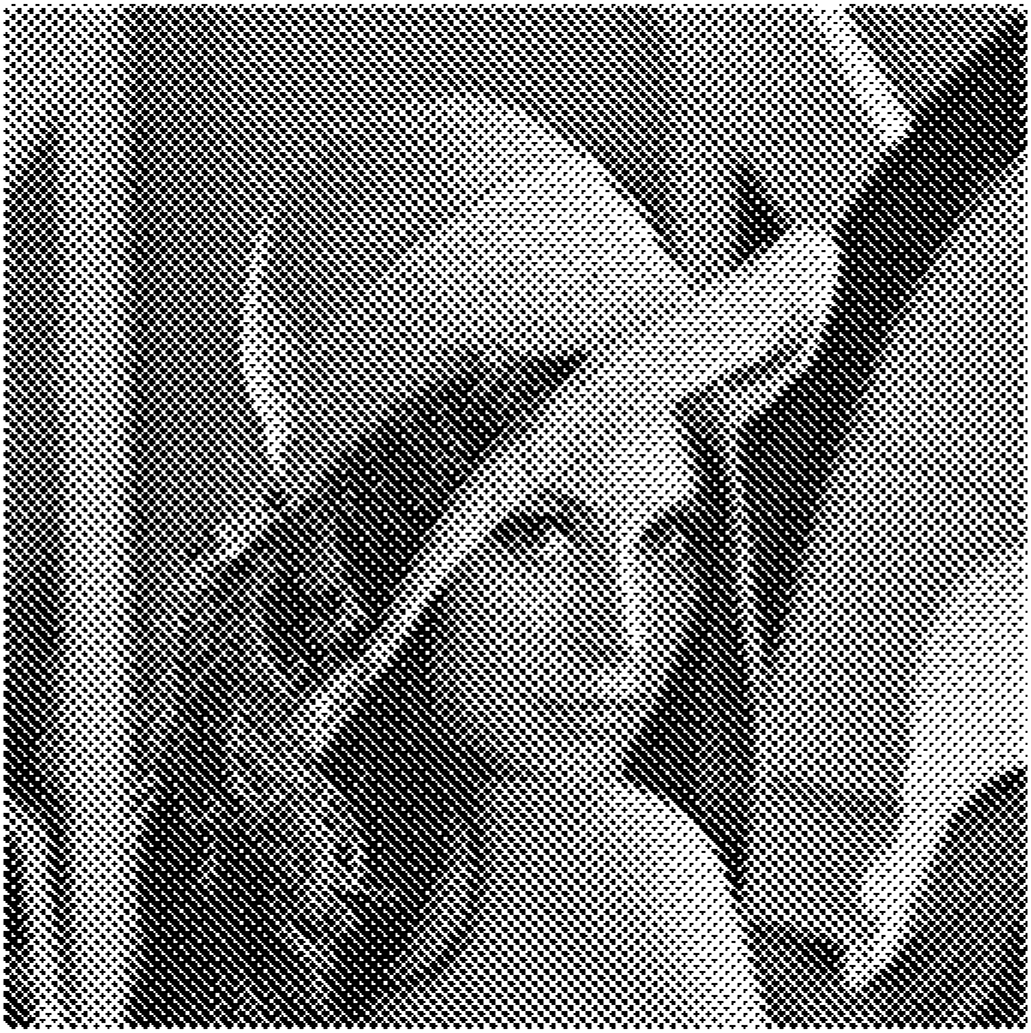


Figure 8-5  Lossless 6x6 Halftone of Lena: High Rate Starting Image

The starting point for the low rate method is an image with halftone a pattern size of 2*N*, with only the mean grayscale vectors.  This image is the equivalent of a Lossless 12x12 Halftone, and is illustrated in Figure 8-6.

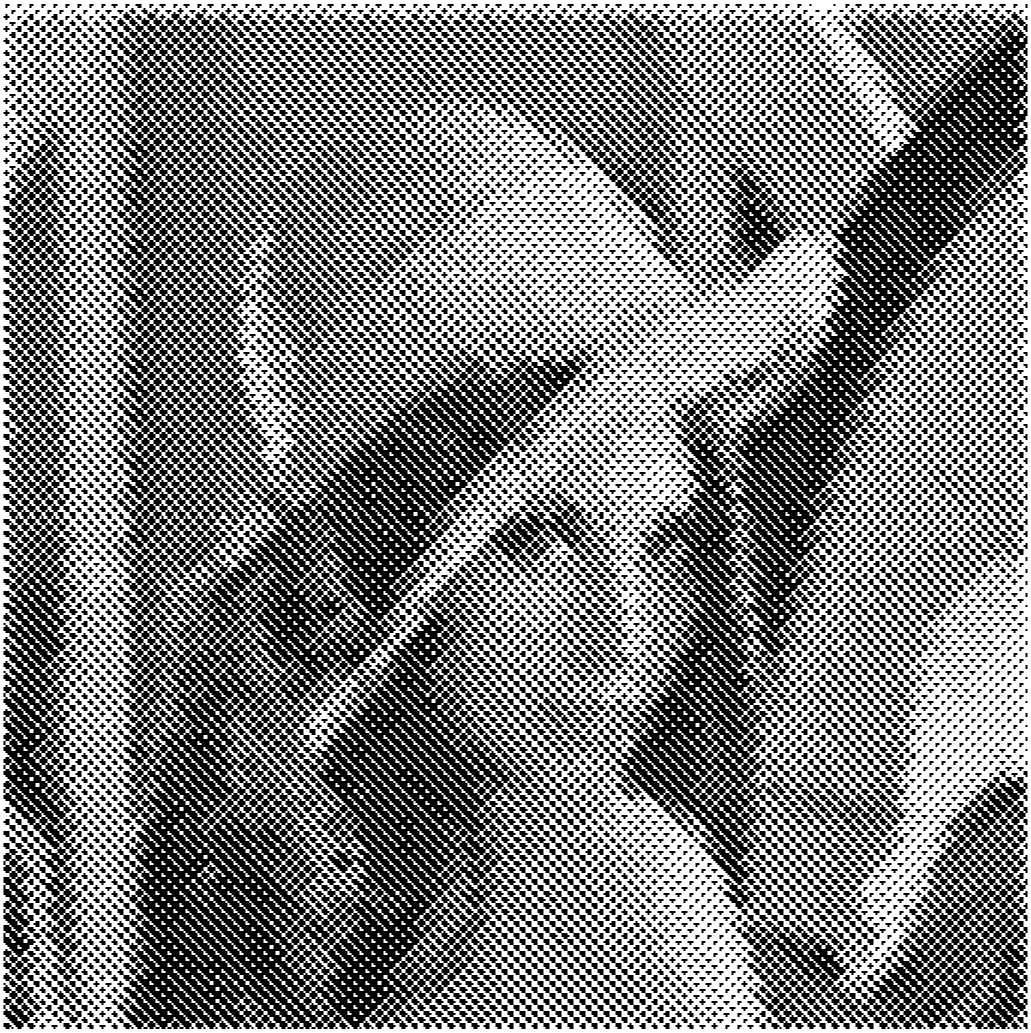


Figure 8-6 Equivalent of Lossless 12x12 Halftone: Low Rate Starting Image

The halftone coding methods were applied to the image Lena, and numerous images were generated, with each image corresponding to a different point on the Rate-Distortion curve. The curve that was generated is illustrated in Figure 8-7, and several of those images are included in Figure 8-8.

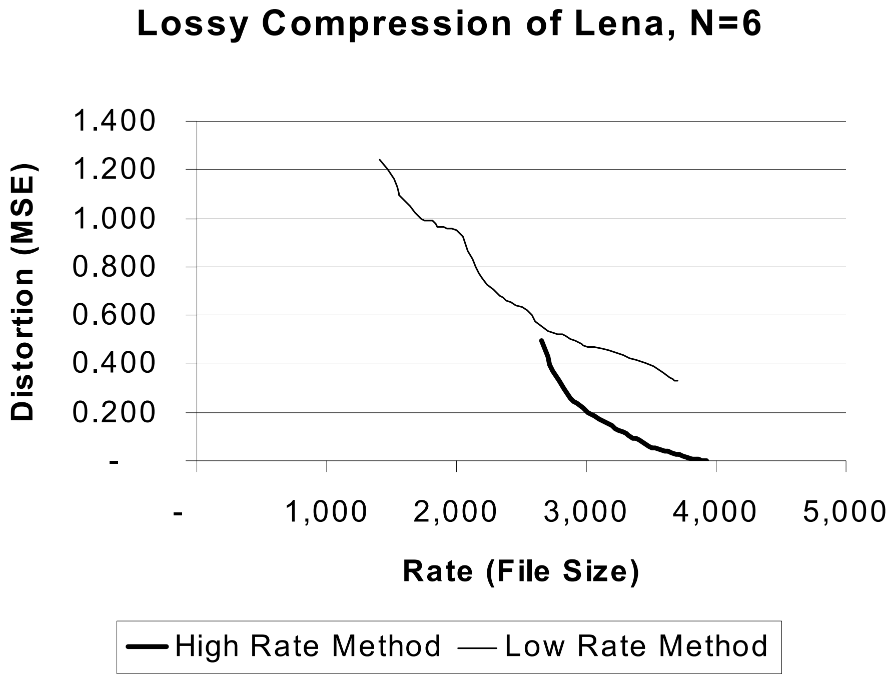
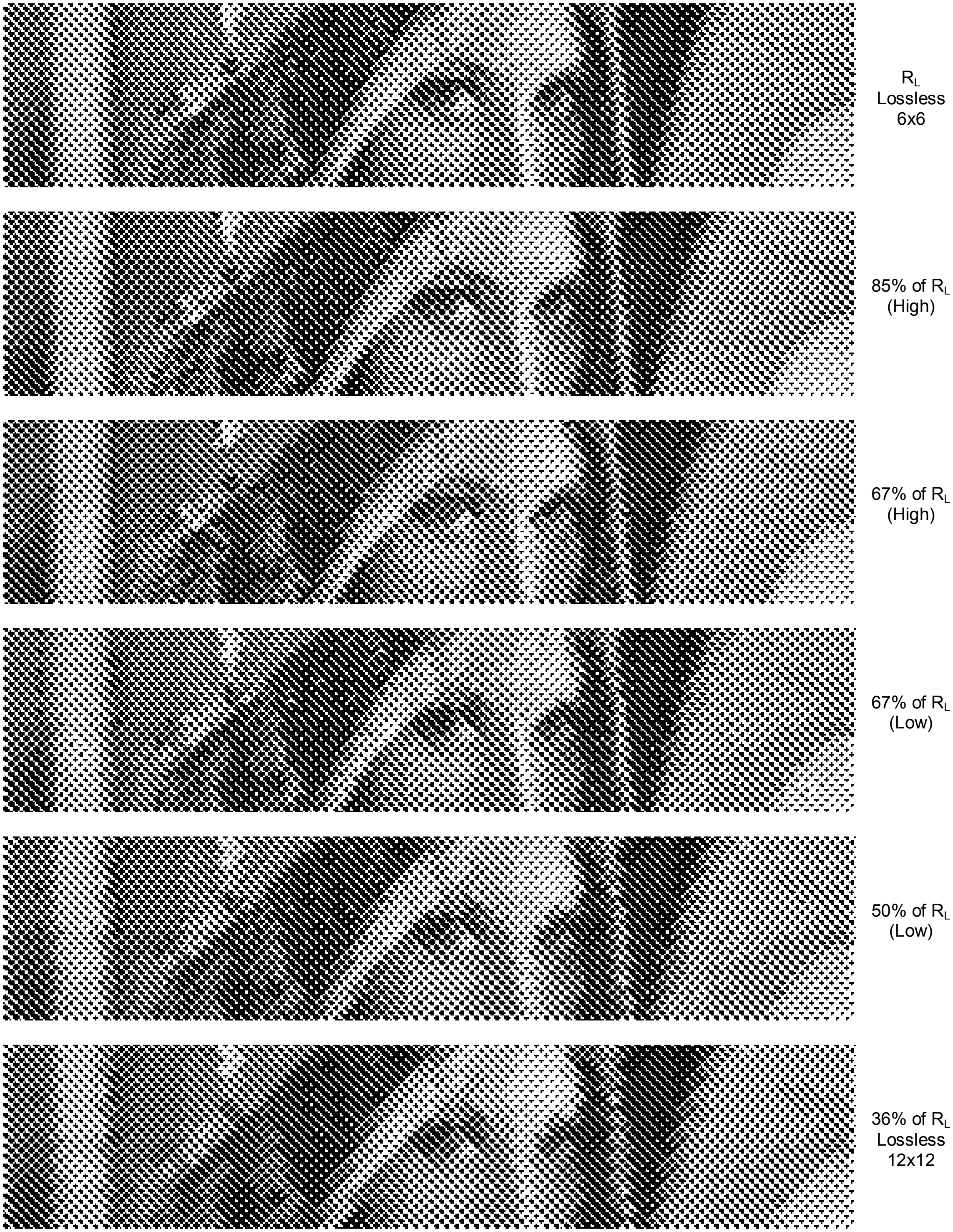


Figure 8-7 Rate-Controlled Halftone Coding of the Image Lena

From the graph in Figure 8-7, the general behaviour described in Figure 8-1 can be observed. It can be seen how the high rate method has the same problem that the generic method has, which is that it eventually saturates and can not reduce the rate further. For this particular image, it cannot even extend far enough to reach the low rate method. The low rate method works very well in the range it was designed for, namely the low rate region of the curve. In the high rate region of the curve, the low rate method has a very large dictionary, where the actual halftone patterns become a significant amount of the bitstream.

The images in Figure 8-8 give a visual interpretation of the rate-distortion curve. The first three images are all produced from the high rate method, and will suffer from similar artifacts as the generic method, namely the smoothing and contouring of the image. The last three images are all produced from the low rate method, and suffer from the block artifacts that are common with low rate VQ approaches. The middle two images show the two methods at the same rate, where the two lines on the curve nearly touch. In the MSE sense, the high rate image is of better quality than the low rate, but as is often the case, subjective quality may differ from the MSE.

Overall, the proposed method has demonstrated remarkable performance. The results from the curve in Figure 8-7 and corresponding images in Figure 8-8 clearly demonstrate the method’s ability to generate a halftoned image at any target rate, with a minimal amount of distortion.



$R_L$
Lossless
6x6

85% of $R_L$
(High)

67% of $R_L$
(High)

67% of $R_L$
(Low)

50% of $R_L$
(Low)

36% of $R_L$
Lossless
12x12

Figure 8-8  Lena Halftone, Coded at Rates Relative to the Lossless 6x6.

# CHAPTER 9. RATE CONTROL IN COMPOUND IMAGES

In each of the previous four chapters, a method was proposed that can achieve rate control for a different region type in JBIG2. In Chapter 4, a method was proposed for segmenting a compound image into different regions. In a practical system, a method would be required for compound images that would apply some or all of the above methods to the segmented regions of the image. A method for compressing compound documents is explained in this chapter, which consists of two sections. The first section describes the method, while the second section presents the results.

## 9.1 PROPOSED METHOD

The first step in the proposed method is to determine the lossless compression rate for the image. At this point, halftone coding is not considered, and all regions are coded either as a text region (with lossless SPM) or as a lossless generic region. Once the lossless rate *($R_{LOSSLESS}$)* has been obtained, it is compared against the target rate *($R_{TARGET}$)*. If $R_{TARGET}$

is greater than $R_{LOSSLESS}$, then the objective has been accomplished. Otherwise, the additional compression factor ($k_{TARGET}$) is determined as

$$k_{TARGET} = \frac{R_{TARGET}}{R_{LOSSLESS}} .$$

The next step in the proposed method is to determine the target rate for each region. The most straightforward method of determining the target rate for each region is to simply multiply the lossless rate for that region by the compression factor *($k_{TARGET}$)*. In other words, if there are *(i)* regions, and the lossless rate for each region is *($R_{Li}$),* then the target rate for each region *($R_{Ti}$)* is simply

$$R_{Ti} = k_{TARGET} \cdot R_{Li} .$$

The problem with this *uniform* approach is that it does not consider the distortion of the image. The uniform approach presumes that all of the rate-distortion curves have the same uniform shape. If all of the regions are of the same type and are using the same distortion measure, then target values can be easily calculated. However, this is rarely the case, and so a method of *weighing* the distortion in each region is required.

A weighing function *($w(k)$)* is defined for an image as a function of the compression factor *($k_{TARGET}$)*. If the aforementioned uniform approach is taken, then all images would have the weighing function $w(k) = k_{TARGET},$ illustrated in Figure 9-1.

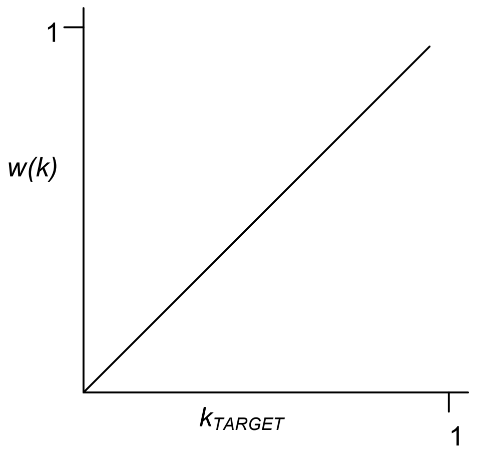


Figure 9-1  Uniform Weighing Function

If non-uniform weighing functions are used, then the target rate for each region *($R_{Ti}$)* is calculated as

$$R_{Ti} = R_{TARGET} \cdot \frac{R_{Li} \cdot w_i(k_{TARGET})}{\sum_j R_{Lj} \cdot w_j(k_{TARGET})}$$

So if a particular region has a higher weighting than the other regions at a specific value of $k_{TARGET}$, then more compression will be applied to it, relative to the other regions. The weighting functions are designed so that the distortion is distributed more evenly across regions of different types.

The sample weighing functions in Figure 9-2 are slightly exaggerated to help to illustrate this behaviour. Text images have small increases in distortion at high rates, but large increases in distortion when substitution errors start to appear.  To compensate for this behaviour, the weighing function forces a text region to be slightly more compressed at higher rates, and to achieve proportionately much less compression at lower rates.  For line-art images, generic bit-flipping can achieve good compression rates, but eventually a saturation point is reached, and halftone coding is required. The weighting function for line-art images reflects this relationship, and will ideally prevent halftone coding from being introduced until the target bitrates are at the same level where substitutions start to occur in text regions. Halftone regions are much better at absorbing distortion than other types of regions, and as such, have larger weighing functions. Stochastic halftones are especially well suited for higher compression rates, since their lossless coding rate is usually quite high.

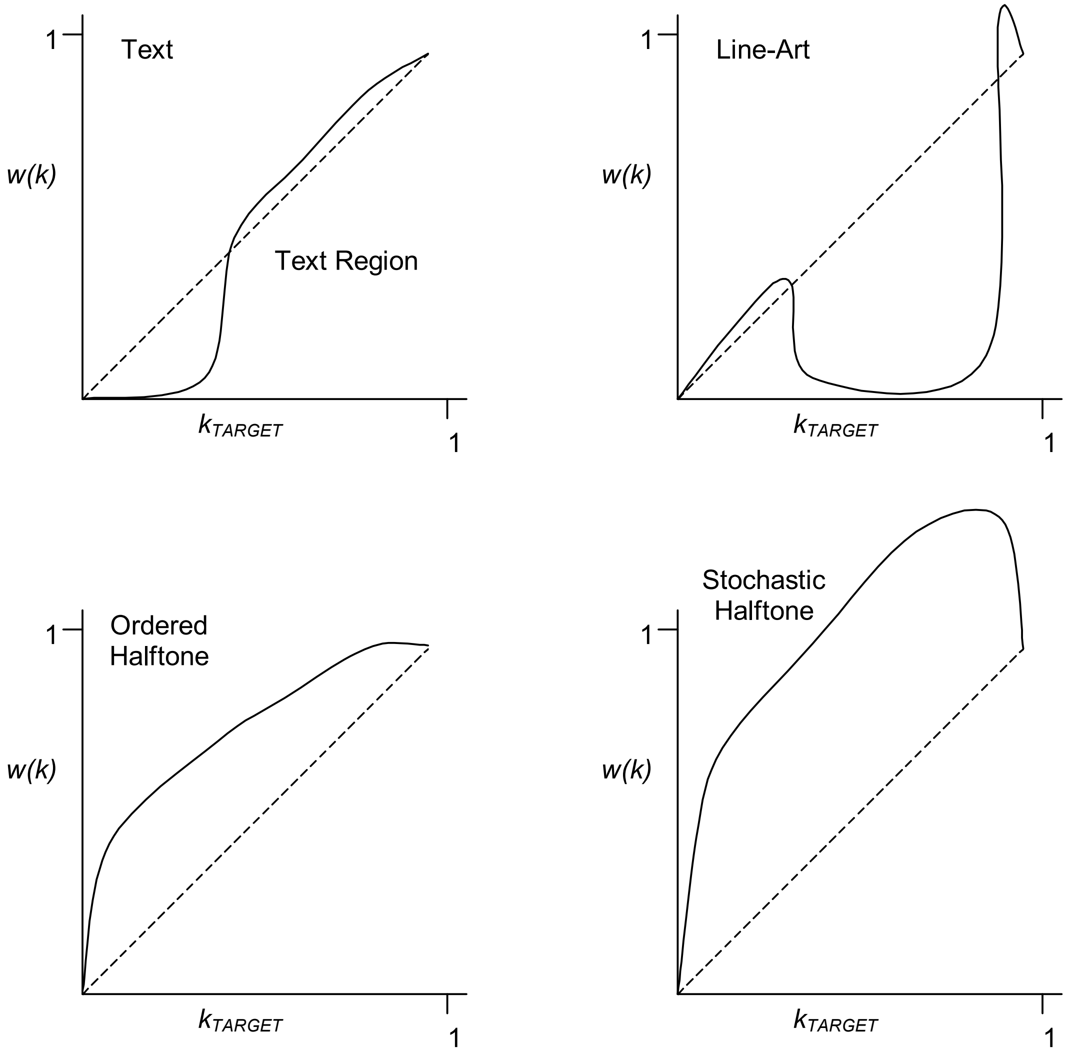


Figure 9-2 Non-Uniform Weighing Functions for Different Image Types

The curves of Figure 9-2 are general approximations, and for a practical system, more accurate information would be required. One approach would be to generate weighing functions based upon the results of training sets. Alternatively, the encoder would use the results obtained from a preliminary compression pass to determine the key characteristics of the regions. For example, the point at which saturation occurs in generic regions, and for text regions the point where substitution errors start to escalate. The encoder could then reiterate with this preliminary information and find target rates that would minimize distortion across the entire image.

## 9.2 RESULTS

To demonstrate the proposed method, a compound document was generated. For these results, the target compression rates are given as a percentage of the lossless rate. The target rates could have just as easily been chosen as a certain number of bytes. The original source image Alice is illustrated in Figure 9-3. This image was specifically generated to be predominantly text, and have a stochastic halftone and a traditional line-art image. The image was printed on a 600dpi printer, and then scanned as a bi-level image at 200dpi. The original scanned image is illustrated in Figure 9-4, and clearly shows the distortion introduced by the scanner.

The image was segmented into three regions and compressed losslessly: one text region and two non-text regions. The stochastic halftone compressed very poorly, and had a lossless compression rate that was actually larger than the text region. The image was then compressed uniformly at 90% of the lossless rate, which is illustrated in Figure 9-5. At this rate, the image is perceptually lossless. The image was then further compressed to 70%, which is illustrated in Figure 9-6. At this rate, there is visible distortion in the line-art region. Since the generic bit-flipping method had saturated for the line-art image, the image had been coded as halftone region.

Figure 9-7 illustrates the image at 50% compression, and there are visible artifacts in every region of the image. The line-art image has gotten progressively worse, since the image is coded with the low-rate halftone method. The text region is now clearly using the CSM method, and some small substitution errors are visible, especially the substitution of the letter 'c' for 'e'. The stochastic halftone has now been coded with a small halftone pattern, using the high-rate method. There is an unusual diagonal bit-flipping artifact, most likely exploiting some pattern introduced by the scanner.

Figure 9-8 illustrates the image at 30% compression, at which point all three regions are represented by halftones, all at different resolutions. The text region is not legible, and the line art region is very blocky. However, since the lossless rate for the stochastic halftone was so high, even at 30% of its' rate it looks quite good. This clearly demonstrates the problem with the uniform approach. For comparison, a non-uniform approach was used to produce Figure 9-9, which obviously looks much better than the previous Figure.

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, “and what is the use of a book,” thought Alice “without pictures or conversation?”

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

There was nothing so VERY remarkable in that; nor did Alice think it so VERY much out of the way to hear the Rabbit say to itself, “Oh dear! Oh dear! I shall be late!” (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

In another moment down went Alice after it, never once considering how in the world she was to get out again.

The rabbit-hole went straight on like a tunnel for some way, and then dipped suddenly down, so suddenly that Alice had not a moment to think about stopping herself before she found herself falling down a very deep well.

Either the well was very deep, or she fell very slowly, for she had plenty of time as she went down to look about her and to wonder what was going to happen next. First, she tried to look down and make out what she was coming to, but it was too dark to see anything; then she looked at the sides of the well, and noticed that they were filled with cupboards and book-shelves; here and there she saw maps and pictures hung upon pegs. She took down a jar from one of the shelves as she passed; it was labelled “ORANGE MARMALADE”, but to her great disappointment it was empty: she did not like to drop the jar for fear of killing somebody, so managed to put it into one of the cupboards as she fell past it.

“Well!” thought Alice to herself, “after such a fall as this, I shall think nothing of tumbling down stairs! How brave they’ll all think me at home! Why, I wouldn’t say anything about it, even if I fell off the top of the house!” (Which was very likely true.)

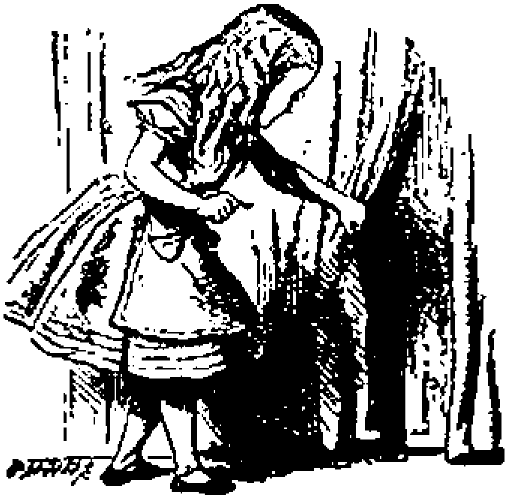
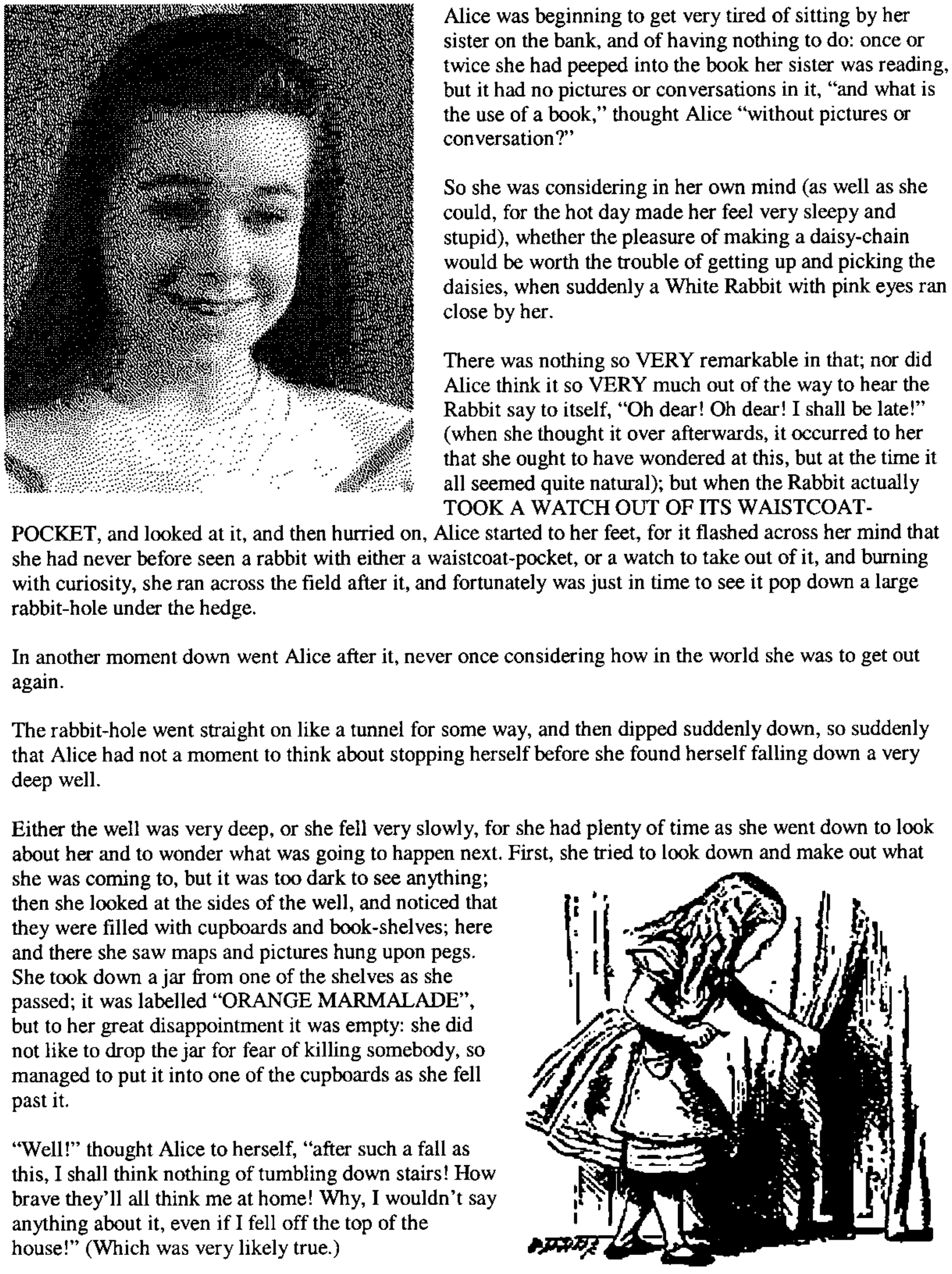


Figure 9-3 Original Source for Alice Image



Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice "without pictures or conversation?"

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

There was nothing so VERY remarkable in that; nor did Alice think it so VERY much out of the way to hear the Rabbit say to itself, "Oh dear! Oh dear! I shall be late!" (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

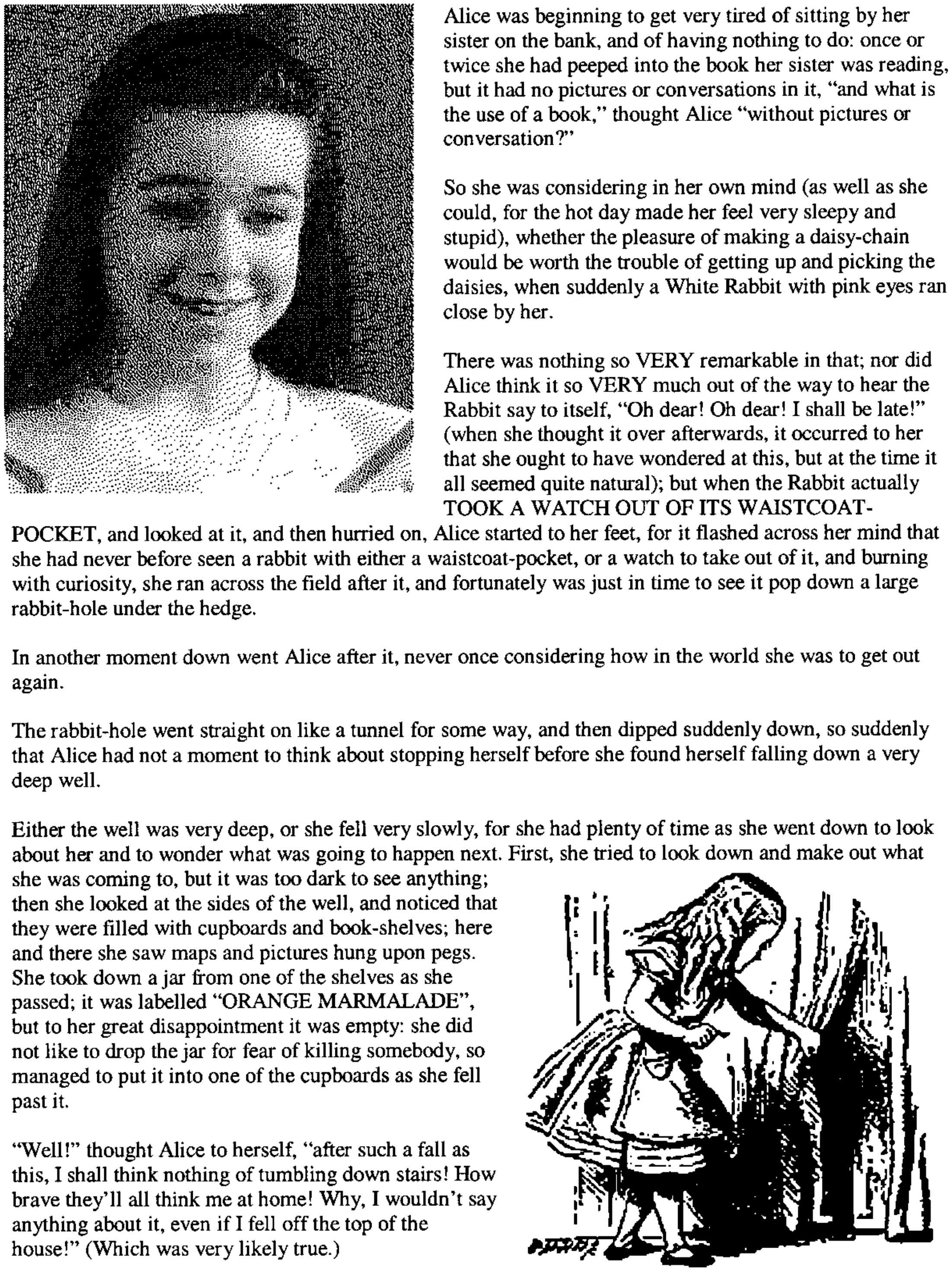
In another moment down went Alice after it, never once considering how in the world she was to get out again.

The rabbit-hole went straight on like a tunnel for some way, and then dipped suddenly down, so suddenly that Alice had not a moment to think about stopping herself before she found herself falling down a very deep well.

Either the well was very deep, or she fell very slowly, for she had plenty of time as she went down to look about her and to wonder what was going to happen next. First, she tried to look down and make out what she was coming to, but it was too dark to see anything; then she looked at the sides of the well, and noticed that they were filled with cupboards and book-shelves; here and there she saw maps and pictures hung upon pegs. She took down a jar from one of the shelves as she passed; it was labelled "ORANGE MARMALADE", but to her great disappointment it was empty: she did not like to drop the jar for fear of killing somebody, so managed to put it into one of the cupboards as she fell past it.

"Well!" thought Alice to herself, "after such a fall as this, I shall think nothing of tumbling down stairs! How brave they'll all think me at home! Why, I wouldn't say anything about it, even if I fell off the top of the house!" (Which was very likely true.)

Figure 9-4 Original Scanned Alice Image (200dpi). $R_{LOSSLESS}$ [ 6.2 : 1 ]



Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice "without pictures or conversation?"

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

There was nothing so VERY remarkable in that; nor did Alice think it so VERY much out of the way to hear the Rabbit say to itself, "Oh dear! Oh dear! I shall be late!" (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

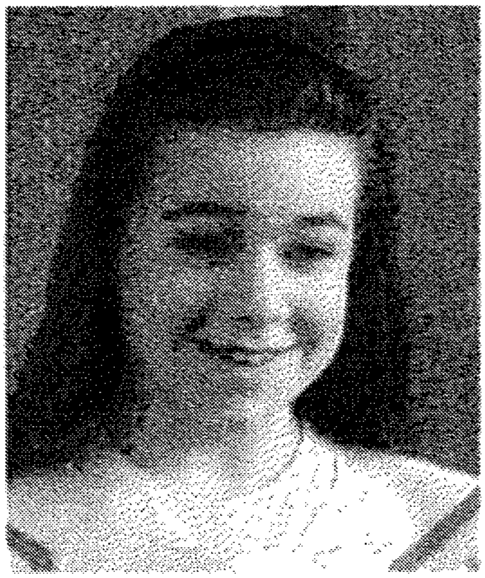
In another moment down went Alice after it, never once considering how in the world she was to get out again.

The rabbit-hole went straight on like a tunnel for some way, and then dipped suddenly down, so suddenly that Alice had not a moment to think about stopping herself before she found herself falling down a very deep well.

Either the well was very deep, or she fell very slowly, for she had plenty of time as she went down to look about her and to wonder what was going to happen next. First, she tried to look down and make out what she was coming to, but it was too dark to see anything; then she looked at the sides of the well, and noticed that they were filled with cupboards and book-shelves; here and there she saw maps and pictures hung upon pegs. She took down a jar from one of the shelves as she passed; it was labelled "ORANGE MARMALADE", but to her great disappointment it was empty: she did not like to drop the jar for fear of killing somebody, so managed to put it into one of the cupboards as she fell past it.

"Well!" thought Alice to herself, "after such a fall as this, I shall think nothing of tumbling down stairs! How brave they'll all think me at home! Why, I wouldn't say anything about it, even if I fell off the top of the house!" (Which was very likely true.)

Figure 9-5 Lossy Alice, Uniform Weighting, 90% of $R_{LOSSLESS}$ [ 6.9 : 1 ]



Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice "without pictures or conversation?"

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

There was nothing so VERY remarkable in that; nor did Alice think it so VERY much out of the way to hear the Rabbit say to itself, "Oh dear! Oh dear! I shall be late!" (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

In another moment down went Alice after it, never once considering how in the world she was to get out again.

The rabbit-hole went straight on like a tunnel for some way, and then dipped suddenly down, so suddenly that Alice had not a moment to think about stopping herself before she found herself falling down a very deep well.

Either the well was very deep, or she fell very slowly, for she had plenty of time as she went down to look about her and to wonder what was going to happen next. First, she tried to look down and make out what she was coming to, but it was too dark to see anything; then she looked at the sides of the well, and noticed that they were filled with cupboards and book-shelves; here and there she saw maps and pictures hung upon pegs. She took down a jar from one of the shelves as she passed; it was labelled "ORANGE MARMALADE", but to her great disappointment it was empty: she did not like to drop the jar for fear of killing somebody, so managed to put it into one of the cupboards as she fell past it.

"Well!" thought Alice to herself, "after such a fall as this, I shall think nothing of tumbling down stairs! How brave they'll all think me at home! Why, I wouldn't say anything about it, even if I fell off the top of the house!" (Which was very likely true.)

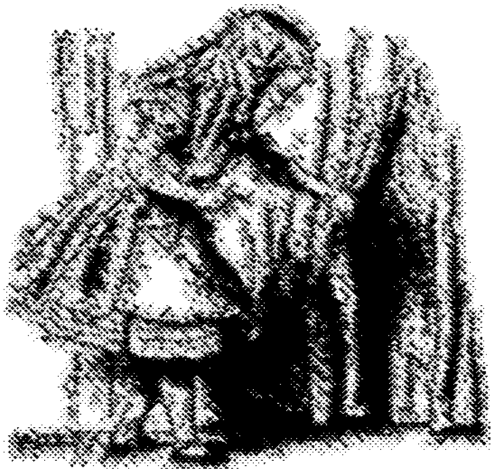
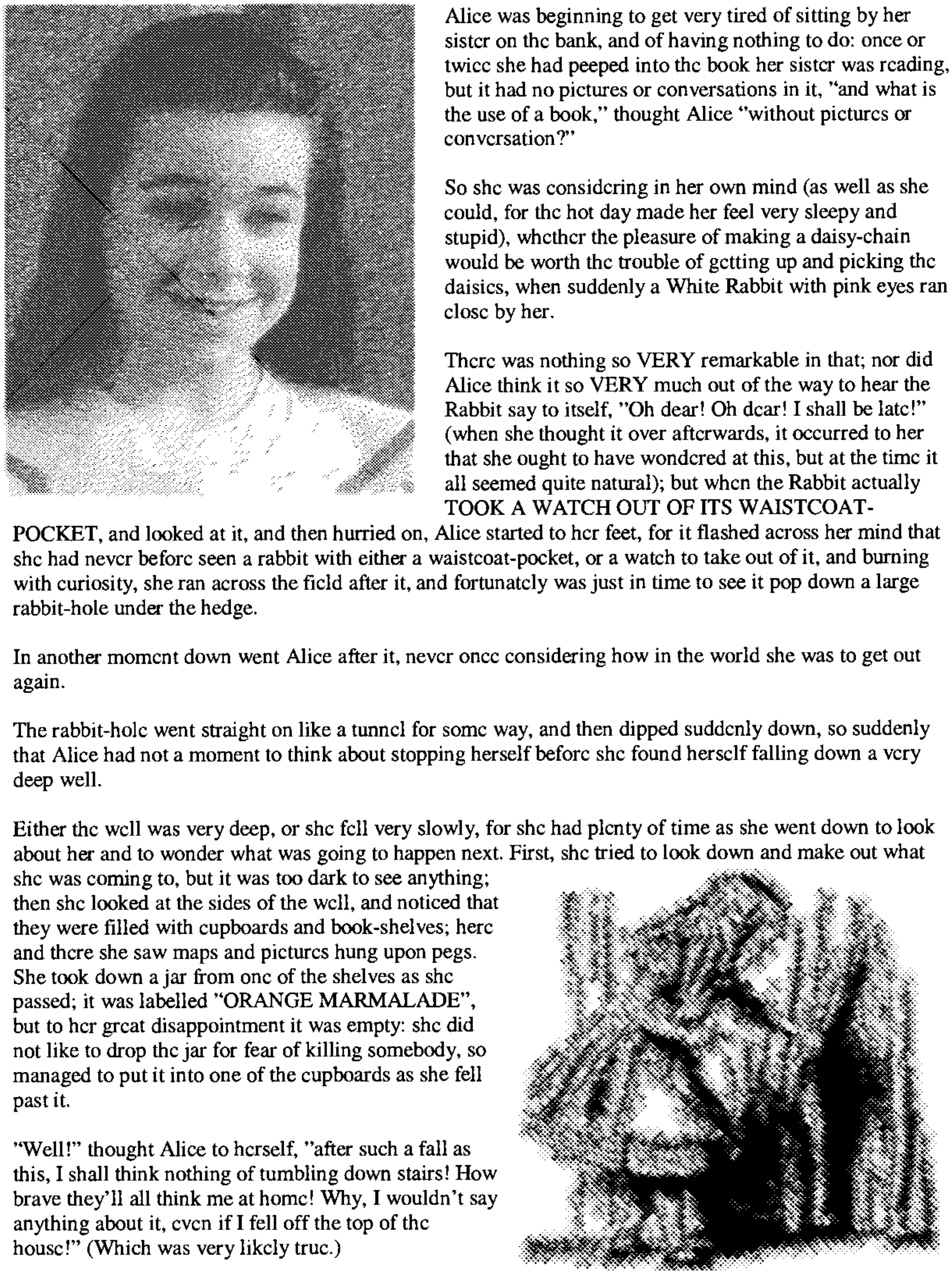


Figure 9-6 Lossy Alice Image, Uniform Weighting, 70% of $R_{LOSSLESS}$ [ 8.8 : 1 ]



Alice was beginning to get very tired of sitting by her sistcr on thc bank, and of having nothing to do: once or twicc she had peeped into thc book her sistcr was rcading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice "without picturcs or convcrsation?"

So shc was considcring in her own mind (as well as she could, for thc hot day made her feel very sleepy and stupid), whcthcr the pleasure of making a daisy-chain would be worth thc trouble of gctting up and picking thc daisics, when suddenly a White Rabbit with pink eyes ran closc by her.

Thcrc was nothing so VERY remarkable in that; nor did Alice think it so VERY much out of the way to hear the Rabbit say to itself, "Oh dear! Oh dcar! I shall be latc!" (when she thought it over aftcrwards, it occurred to her that she ought to have wondcred at this, but at the timc it all seemed quite natural); but whcn the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on, Alice started to hcr feet, for it flashed across her mind that shc had nevcr beforc seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the ficld after it, and fortunatcly was just in time to see it pop down a large rabbit-hole under the hedge.

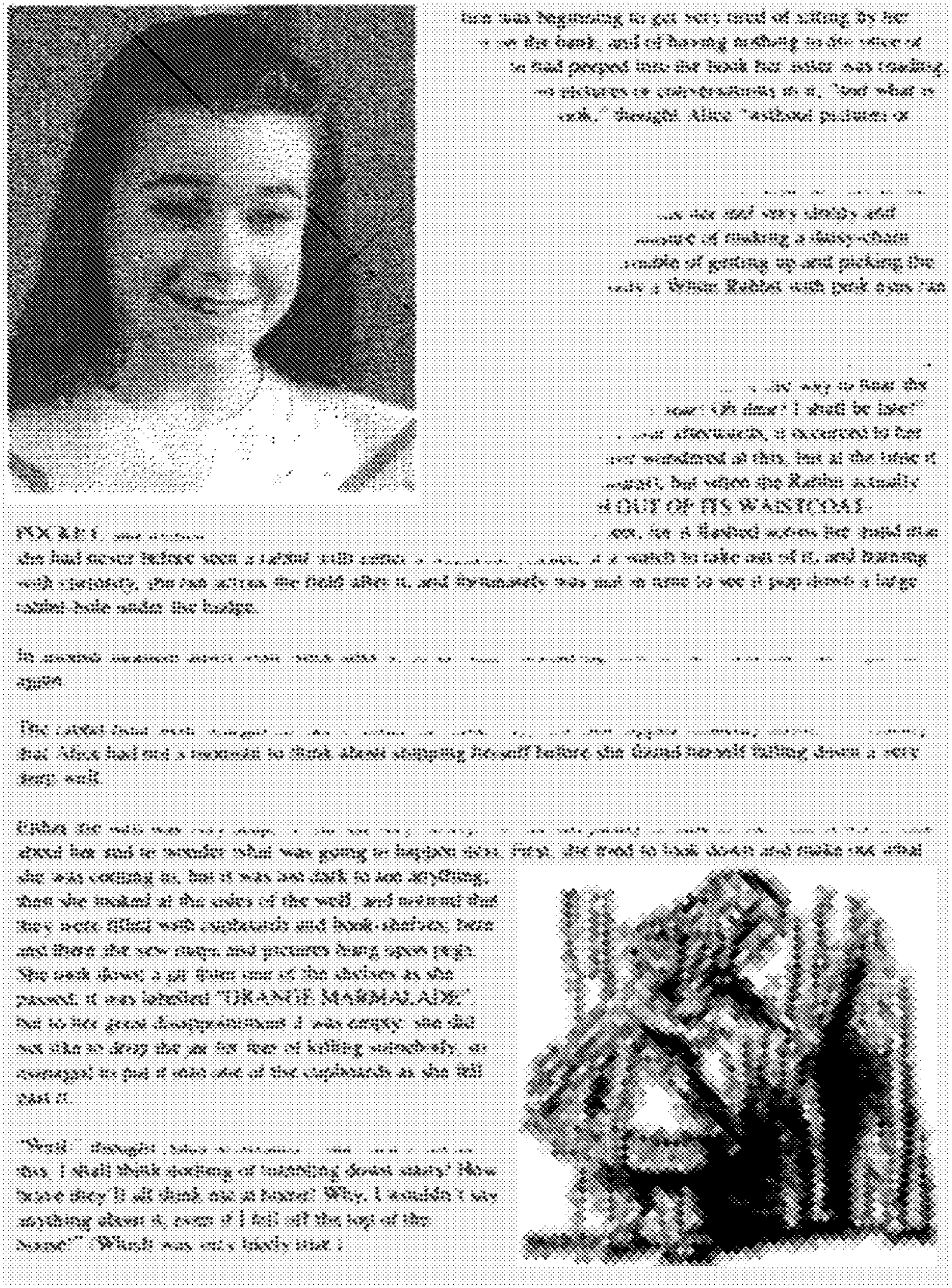
In another momcnt down went Alice after it, nevcr oncc considering how in the world she was to get out again.

The rabbit-holc went straight on like a tunncl for somc way, and then dipped suddcnly down, so suddenly that Alice had not a moment to think about stopping herself beforc shc found hersclf falling down a vcry deep well.

Either thc wcll was very deep, or shc fcll very slowly, for shc had plcnty of time as she went down to look about her and to wonder what was going to happen next. First, shc tried to look down and make out what shc was coming to, but it was too dark to see anything; then shc looked at the sides of the wcll, and noticed that they were filled with cupboards and book-shelves; herc and thcre she saw maps and picturcs hung upon pegs. She took down a jar from onc of the shelves as shc passed; it was labelled "ORANGE MARMALADE", but to hcr grcat disappointment it was empty: shc did not like to drop thc jar for fear of killing somebody, so managed to put it into one of the cupboards as she fell past it.

"Well!" thought Alice to hcrself, "after such a fall as this, I shall think nothing of tumbling down stairs! How brave they'll all think me at homc! Why, I wouldn't say anything about it, cvcn if I fell off the top of thc housc!" (Which was very likcly truc.)

Figure 9-7 Lossy Alice Image, Uniform Weighting, 50% of $R_{LOSSLESS}$ [ 12.4 : 1 ]



…ice was beginning to get very tired of sitting by her
… on the bank, and of having nothing to do: once or
…he had peeped into the book her sister was reading,
…no pictures or conversations in it, "and what is
…ook," thought Alice "without pictures or

… as she could, for the hot day made her feel very sleepy and
…easure of making a daisy-chain
…rouble of getting up and picking the
…enly a White Rabbit with pink eyes ran

… the way to hear the
…ear! Oh dear! I shall be late!"
… over afterwards, it occurred to her
…ave wondered at this, but at the time it
…atural); but when the Rabbit actually
…H OUT OF ITS WAISTCOAT-

POCKET, … …ent, for it flashed across her mind that she had never before seen a rabbit with either a … …, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

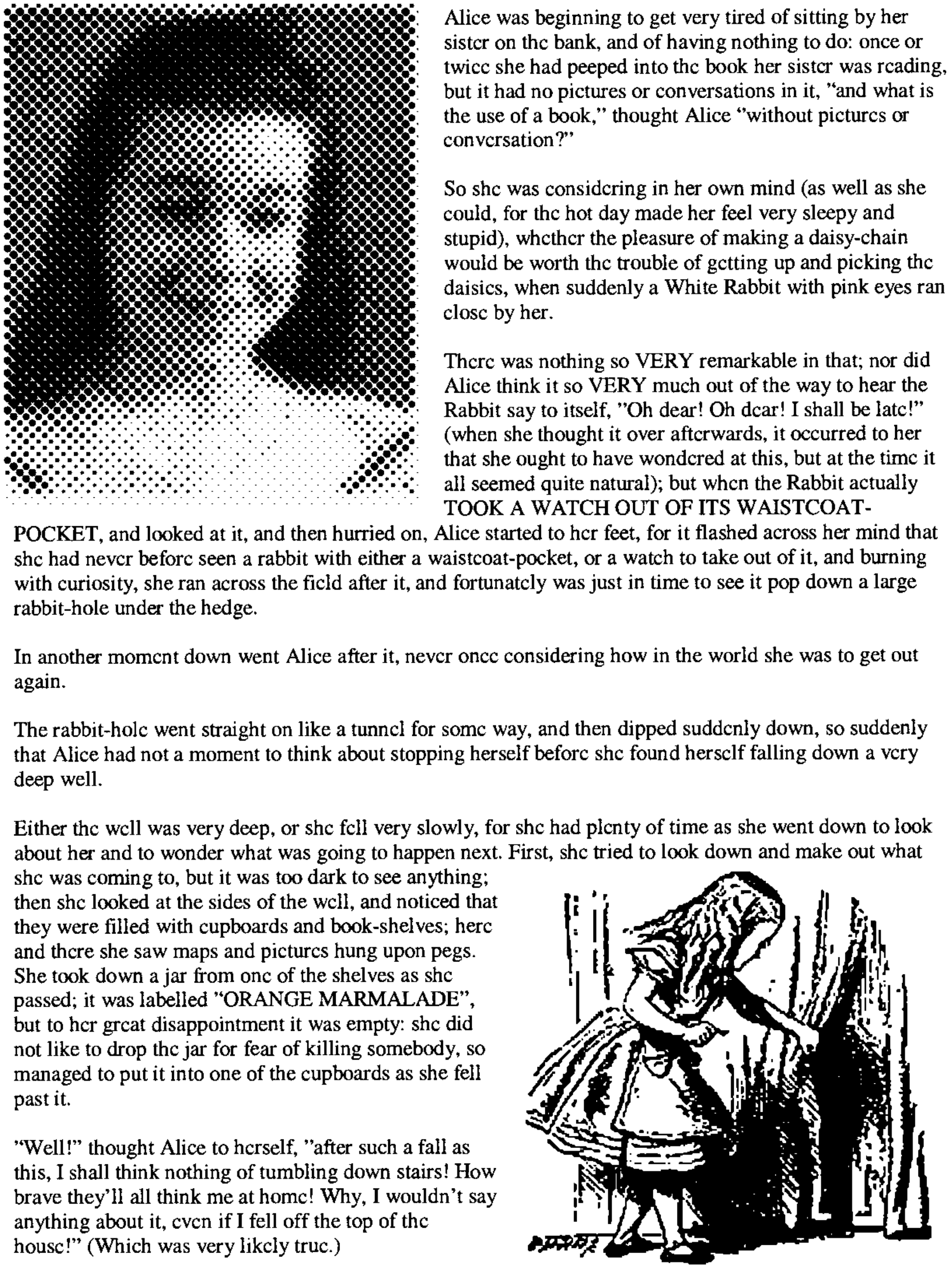
In another moment down went Alice after it, … …
again.

The rabbit-hole went straight on … … …, so suddenly that Alice had not a moment to think about stopping herself before she found herself falling down a very deep well.

Either the well was very deep, … … … about her and to wonder what was going to happen next. First, she tried to look down and make out what she was coming to, but it was too dark to see anything; then she looked at the sides of the well, and noticed that they were filled with cupboards and book-shelves; here and there she saw maps and pictures hung upon pegs. She took down a jar from one of the shelves as she passed; it was labelled "ORANGE MARMALADE", but to her great disappointment it was empty: she did not like to drop the jar for fear of killing somebody, so managed to put it into one of the cupboards as she fell past it.

"Well!" thought Alice … … … this, I shall think nothing of tumbling down stairs! How brave they'll all think me at home! Why, I wouldn't say anything about it, even if I fell off the top of the house!" (Which was very likely true.)

Figure 9-8  Lossy Alice Image, Uniform Weighting, 30% of $R_{LOSSLESS}$ [ 20.6 : 1 ]



Alice was beginning to get very tired of sitting by her sistcr on thc bank, and of having nothing to do: once or twicc she had peeped into thc book her sistcr was rcading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice "without picturcs or convcrsation?"

So shc was considcring in her own mind (as well as she could, for thc hot day made her feel very sleepy and stupid), whcthcr the pleasure of making a daisy-chain would be worth thc trouble of gctting up and picking thc daisics, when suddenly a White Rabbit with pink eyes ran closc by her.

Thcrc was nothing so VERY remarkable in that; nor did Alice think it so VERY much out of the way to hear the Rabbit say to itself, "Oh dear! Oh dcar! I shall be latc!" (when she thought it over aftcrwards, it occurred to her that she ought to have wondcred at this, but at the timc it all seemed quite natural); but whcn the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on, Alice started to hcr feet, for it flashed across her mind that shc had nevcr beforc seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the ficld after it, and fortunatcly was just in time to see it pop down a large rabbit-hole under the hedge.

In another momcnt down went Alice after it, nevcr oncc considering how in the world she was to get out again.

The rabbit-holc went straight on like a tunncl for somc way, and then dipped suddcnly down, so suddenly that Alice had not a moment to think about stopping herself beforc shc found hersclf falling down a vcry deep well.

Either thc wcll was very deep, or shc fcll very slowly, for shc had plcnty of time as she went down to look about her and to wonder what was going to happen next. First, shc tried to look down and make out what shc was coming to, but it was too dark to see anything; then shc looked at the sides of the wcll, and noticed that they were filled with cupboards and book-shelves; herc and thcre she saw maps and picturcs hung upon pegs. She took down a jar from onc of the shelves as shc passed; it was labelled "ORANGE MARMALADE", but to hcr grcat disappointment it was empty: shc did not like to drop thc jar for fear of killing somebody, so managed to put it into one of the cupboards as she fell past it.

"Well!" thought Alice to hcrself, "after such a fall as this, I shall think nothing of tumbling down stairs! How brave they'll all think me at homc! Why, I wouldn't say anything about it, cvcn if I fell off the top of thc housc!" (Which was very likcly truc.)

Figure 9-9 Lossy Alice Image, Non-Uniform Weighting, 30% of $R_{LOSSLESS}$ [20.6:1]

# CHAPTER 10. CONCLUSIONS

This chapter contains the final conclusions. In the first section, the major contributions are summarized. In the last section, there are suggested areas of further research.

## 10.1 SUMMARY

The objective of the research was to develop a method for using JBIG2 to compress a bi-level image to any rate, with the minimum amount of distortion. While no single technique was developed to achieve this goal, several different strategies have been proposed, that when used together can produce equivalent results. In this section, the various components of the research are summarized.

In Chapter 3, the problems of measuring distortion in bi-level images are addressed. Although some of the distortion measures presented are novel for bi-level image applications, the majority of the material has been presented before. However, this chapter is still very important as it highlights a concept that has been largely overlooked in the literature.

In Chapter 4, an image segmentation method is presented. Unlike many other segmentation methods, this method has been specifically designed for bi-level compression applications, with the simple classifications types of text and non-text. The method spatially reduces the image with a technique known as smearing, and then analyzes the resulting clusters for non-text characteristics. The method presented is novel and produces excellent results.

In Chapter 5, the task of controlling the rate of a generic region is addressed. The basic approach to reducing the rate, known as bit-flipping, is not novel. However, the basic method is not very accurate, and several novel enhancements to the base method have been proposed, including adjustments for the characteristics of the MQ Coder. The material presented in Chapter 6 is the application of the bi-flipping method to refinement regions, which is the first time that this straightforward approach has been formally proposed.

In Chapter 7, a method for controlling the rate of a text region is proposed. This material is the cornerstone of the research, and is the first comprehensive solution to this problem. By combining both the CSM and SPM approaches, both high and low bitrates can be accommodated, and by providing a framework for incrementally changing dictionary sizes, accurate rate control is achieved. The proposed method is especially robust as it finds the image with a minimal amount of distortion, for any measure of distortion.

In Chapter 8, a method for controlling the rate of a halftone region is proposed. The method uses two different and novel approaches to converge at the target rate from either direction. In the high rate method, a multi-level bit-flipping method is used on the reduced grayscale image. In the low rate method, vector quantization is used to select a fewer number of halftone patterns, at a reduced resolution.

In Chapter 9, a method for determining the individual target rates for a compound image is proposed. The method is a unique, straightforward solution to a complicated problem. Chapter 9 brings all of the other components together, completing a comprehensive approach to achieving rate control for bi-level image compression.

## 10.2 FURTHER RESEARCH

Although the primary goal of the research has been reached, there are still numerous areas of research that can be explored and expanded. In the following sections, different avenues of further research are suggested.

### 10.2.1 DISTORTION MEASUREMENTS

As Chapter 3 discusses, determining the minimum amount of distortion is a difficult task, and can be very subjective, especially for text images. Ideally, the research would have produced a method that would be able to generate the optimal image at a given rate for any desired distortion measure. Academically, this flexibility would have been very beneficial, allowing for accurate comparisons between the proposed method and other JBIG2 implementations, in addition to comparing JBIG2 against other compression schemes. While this goal was met for text based regions, it was not met for generic or halftone regions, and so a more comprehensive approach could be developed.

### 10.2.2 MULTIPLE BIT-FLIPPING

The generic bit-flipping algorithms used in this research are all based upon flipping one bit, and analyzing the distortion that is introduced by flipping just one bit. As Section 5.1 explains, flipping that one bit can have a considerable consequences, and the rate analysis can be quite complicated. Consequently, the analysis for flipping two or more bits is even more complicated, but simplifications can be made so that the analysis is only slightly worse. However, the drawback of flipping two bits simultaneously is not the more complicated rate analysis, but rather the task of finding the two ideal flip candidates, which is a computational nightmare. If a clever method was developed that could analyze an image and find multiple simultaneous flip candidates, then it would be a feasible solution to improve generic compression.

### 10.2.3 SHARED DICTIONARIES

In the proposed method for achieving rate control in text images, each text region is considered separately. If a compound image has multiple text regions, then for each region there is a symbol dictionary segment and a text region segment. However, as Section 2.5.5 explains, those individual dictionaries could refer to common dictionaries, or to each other. If any of the text regions have similar symbols then some significant compression gains could be achieved by sharing those symbols. The proposed method for achieving rate control does not consider this sharing of symbols. Including a shared symbol analysis would significantly increase the complexity of the method, but with potentially large compression gains.

### 10.2.4 MORE FLEXIBLE HALFTONE PARAMETERS

Some of the halftone parameters that are used in the proposed method are fixed, to reduce the complexity of the method. However, in some circumstances it is possible that better rate-distortion performance could be achieved with different parameters. For example, to reduce distortion the halftone grid angle is set to 45º, but for a particular image there may be an enormous compression improvement if it is set to 0º, or even 31.4º, which could more than compensate for any increase in distortion. A more flexible halftone coding method combined with a more sophisticated halftone analysis would produce better results than the proposed method.

### 10.2.5 COMPOUND IMAGES

In Chapter 9, a method is proposed to control the rate of compound images. To compensate for different distortion measures between regions, and different region types, a weighting curve was used. In the proposed method, only a rudimentary approach was used to estimate those weighting curves. In a more robust implementation, those weighting curves would be more accurately determined.

### 10.2.6 MULTIPLE PAGES

One of the key advantages of using JBIG2 is the ability to compress more than one page of information, and to share information between pages. Since the research was targeted at a single bi-level image, this advantage of JBIG2 has been largely ignored. If rate control for a multi-page document is desired, then the analysis becomes more difficult. If the rate for each page is determined uniformly, then it could produce the same undesirable effects of uniform region weighting that are illustrated in Figure 9-8. Instead, a comprehensive analysis of all of the compound pages would be required to minimize distortion across the entire document. Such an analysis becomes even more complicated if symbol dictionaries are shared across pages, which was discussed in Section 10.2.3. Clearly, a multi-page approach would increase the complexity and scope of the proposed method considerably.

# GLOSSARY

| Expression | Brief Description | Location |
|---|---|---|
| Arithmetic Coding | An advanced method of compressing data with prediction. | Appendix A |
| Artifact | Part of an image where distortion that is clearly visible. | 2.3 |
| BER | Bit Error Rate. The number of bits in an image that are incorrect, usually measured as a percent (%). | 3.1 |
| Bi-Level | Refers to an image that has only 2 colours. | 2.1 |
| Bitstream | A sequence of bits that are the output of a compression system. | 2.5 |
| Clustered Dot | A type of ordered halftone where all of the pixels in the patterns are clustered. | 2.2.2 |
| Context | An arrangement of pixels that are used to model image behaviour and help predict the value of a pixel. | 2.5.3 |
| CSF | Contrast Sensitivity Function. Used to measure distortion in an image. | 3.3.2 |
| CSM | Combined Symbol Matching. A method of lossy text coding where symbols are replaced by other, similar symbols. | 7.1 |
| Decode | The process of generating an uncompressed image from a compressed bitstream. | 2.3 |
| Distortion | The difference between the two images. | 2.3 |
| DPI | Dots Per Inch. Used to describe the resolution of an image. | 2.1 |
| Encode | The process of generating a compressed bitstream from an uncompressed image. | 2.2.2 |
| Error Diffusion Halftone | A stochastic halftone. | 2.2.2 |
| Facsimile Machine | A device that scans a page and then transmits the scanned image over a phone line to another facsimile machine for printing. | 1.1 |
| Generic Coding | A method of bi-level image coding that is robust for almost all image types. | 2.5.3 |
| Generic Regions | The term used to describe bi-level regions that are not halftones or textual. | 2.2.3 |

| | | |
|---|---|---|
| Gray Coding | A method used to reorder the traditional method of assigning binary codes to integers. In Gray coding, sequential integers differ in their binary representations by only one bit. | |
| Halftone | A bi-level approximation of a multi-level image. | 2.2.2 |
| Hamming Distance | The number of bits that are different between two sequences of bits or images. | 3.1 |
| Huffman Coding | A fast method of compressing data that assigns different code lengths to each data element. | 2.4.1 |
| HVS | Human Visual System. Used to describe the physical properties of the way the Human eyes and brain work. | 3.3.2 |
| ISO | International Standards Organization. | 2.4 |
| ITU | International Telecommunication Union. | 2.4 |
| ITU-T | Telecommunication standardization sector of the ITU. | 2.4 |
| JBIG | Joint Bi-level Image Experts Group. | 2.4 |
| Lossless Compression | Refers to compression methods where the reconstructed data is exactly the same as the original. | 2.3 |
| Lossy Compression | Refers to compression methods where there are differences between the reconstructed data and the original. | 2.3 |
| LPS | Least Probable Symbol(s). Used to refer to either 1 or 0, which ever is the least probable under the circumstances. | 2.5.1 |
| Magnafax Telecopier | The name of the first commercial facsimile machine. | 1.1 |
| MB | 1 MB = 1 MegaByte = 1,000,000 Bytes. | |
| MFBF | Martins-Forchhammer Bit Flipping. A method of increasing compression with generic coding. | 5.2 |
| MH | Modified Huffman. A method of compressing bi-level images. | 2.4.1 |
| MMR | Modified Modified Read. A method of compressing bi-level images. A modified version of MR coding. | 2.4.1 |
| MPS | Most Probable Symbol(s). Used to refer to either 1 or 0, which ever is the most probable under the circumstances. | 2.5.1 |
| MQ | Modified Q-Coder. The arithmetic coder used in the JBIG2 standard. | 2.4.1 |
| MR | Modified Read. A method of compressing bi-level images. A modified version of MH coding. | 2.4.1 |
| MSE | Mean Squared Error. Refers to a measurement of the differences between two images. | 3.1 |

| | | |
|---|---|---|
| Multi-Level | Refers to an image that has more than 2 colours. | 2.1 |
| OCR | Optical Character Recognition. When a bi-level image of text is converted to a textual data format of letters and words. | 3.2.4 |
| Ordered Halftone | A type of halftone where all of patterns organized in a grid pattern. | 2.2.2 |
| Pantelegraph | One of the first machines to transmit bi-level images over telegraph lines. | 1.1 |
| Pattern | A single element of a pattern dictionary, which is used in halftone coding and usually represents one shade of gray. | 2.5.7 |
| Pixel | Picture Element. The smallest element of an image. | 2.1 |
| PSNR | Peak Signal-to-Noise Ratio. | 3.1 |
| QM | The QM arithmetic coder is used in JBIG1 and JPEG. | 2.4.2 |
| Raster Scan Order | Raster Scan Order describes the way that fax machines (and humans) read a fax, starting at the top, going left to right for each row, and then moving down a row. | 2.4.1 |
| Rate Control | The ability to compress data to any rate or size. | 2.3 |
| Refinement Coding | A method of bi-level image coding where a previously coded image is used as a basis to better predict another image, which is usually very similar. | 2.5.4 |
| Renormalization | A term to describe when an arithmetic coder produced an output bit, so that it can maintain the maximum precision. | 2.5.1 |
| Representative Symbol | The symbol that is selected from a set of symbols to represent the entire set in the image. | 7.3 |
| Saturation | An expression used to describe the point in a compression system when the encoder cannot compress the data any further. | 5.4 |
| Smearing | A method of reducing an image that results in the reduced image appearing darker and smeared. | 4.2 |
| SPM | Soft Pattern Matching. Used to describe a text coding method that exploits similarities between symbol with refinement coding. | 7.1 |
| Stochastic Halftones | A type of halftone where no grid is used to generate the image. Instead, dots are seemingly randomly arranged to make artifacts appear as high frequency noise. | 2.2.2 |
| Symbol | A single element of a symbol dictionary, which is used in text coding and usually represents one character or letter. | 2.5.5 |
| Template | Used to describe the shape of a context. | 2.5.3 |

| | | |
|---|---|---|
| VQ | Vector Quantization | 8.4 |
| WXOR | Weighted XOR.  The WXOR is measured by adding all of the XORs in the adjacent 9 pixels. | 3.2.1 |
| XOR | Exclusive Or.  A Boolean function that returns 1 if both inputs are different. | |

# REFERENCES

[Arps94] R. Arps, T. Truong. Comparison of International Standards for Lossless Still Image Compression. *Proceedings of the IEEE,* Vol. 82, No. 6, June 1994.

[Asch74] R. Ascher, G. Nagy. A Means for Achieving a High Degree of Compaction on Scan-Digitized Printed Text. *IEEE Transactions on Computers,* Vol. C-23, 1974.

[Bott98] L. Bottou, P. Howard, Y. Bengio. The Z-Coder Adaptive Binary Coder. *Proceedings of the 1998 Data Compression Conference,* 1998.

[Case96] R. Casey, E. Lecolinet. A Survey of Methods and Strategies in Character Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 18, No. 7, July 1996.

[Dame98] N. Damera-Venkata, T. Kite, M. Venkataraman, B. Evans. Fast Blind Inverse Halftoning. *Proceedings of the 1998 International Conference on Image Processing,* Vol. 2, 1998.

[Dame00] N. Damera-Venkata, T. Kite, W. Geisler, B. Evans, A. Bovik. Image Quality Assessment Based on a Degradation Model. *IEEE Transactions on Image Processing,* Vol. 9, No. 4, April 2000.

[Dobb90] M. Dobbs. Workers of the World, Fax! *Washington Post,* Page C03, December 23, 1990.

[Fowl95] B. Fowler, R. Arps, A. Gamal, D. Yang. Quadtree Based JBIG Compression. *Proceedings of the 1995 Data Compression Conference,* 1995.

[Floy76] R. Floyd, L. Steinberg. An adaptive algorithm for spatial grayscale. *Proceedings of the Society for Image Display,* Vol. 17, No. 2, 1976.

[Fole82] J. Foley, A. Van Dam. *Fundamentals of Interactive Computer Graphics.* Reading, MA: Addison-Wesely, 1982.

[Gers91] A. Gersho, R. Gray. *Vector Quantization and Signal Compression.* Norwell, MA: Kluwer Academic Publishers, 1991.

[Grim92] G. Grimmett, D. Stirzaker. *Probability and Random Processes.* New York: Oxford University Press, 1992.

[Howa94] P. Howard, J. Vitter. Arithmetic Coding for Data Compression. *Proceedings of the IEEE,* Vol. 82, No. 6, June 1994.

[Howa96] P. Howard. Lossless and Lossy Compression of Text Images by Soft Pattern Matching. *Proceedings of the 1996 Data Compression Conference,* 1996.

[Howa97] P. Howard. Text Image Compression Using Soft Pattern Matching. *The Computer Journal,* Vol. 40, 1997.

[Howa98] P. Howard, F. Kossentini, B. Martins, S. Forchhammer, W. Rucklidge, F. Ono. The Emerging JBIG2 Standard. *IEEE Transactions on Circuit and Systems for Video Technology,* Vol. 8, No. 5, September 1998.

[Huff51] D. Huffman. A Method for the Construction of Minimum Redundancy Codes. *Proceedings of the IRE,* Vol. 40, 1951.

[Hunt95] R. Hunt. *The Reproduction of Colour.* Fountain Press, 1995.

[Hutt93] D. Huttenlocher, G. Klanderman, W. Rucklidge. Comparing Images Using the Hausdorff Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 15, No. 9, September 1993.

[Jain98] A. Jain, B. Yu. Document Representation and Its Application to Page Decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 20, No. 3, March 1998.

[John83] O. Johnsen. Coding of Two-Level Pictures by Pattern Matching and Substitution. *Bell System Technical Journal,* Vol. 62, October 1983.

[Lind80] Y. Linde, A. Buzo, R. Gray. An Algorithm for Vector Quantization Design. *IEEE Transactions on Communications,* Vol COM-28, January 1980.

[Kite98] T. Kite, N. Damera-Venkata, B. Evans, A. Bovik. A High Quality, Fast Inverse Halftoning Algorithm for Error Diffused Halftones. *Proceedings of the 1998 International Conference on Image Processing,* Vol. 2, 1998.

[Koss96] F. Kossentini, R. Ward. An Analysis-Compression Technique for Black and White Documents. *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation,* 1996.

[Lang79] G. Langdon. Arithmetic Coding. *IBM Journal of Research & Development,* Vol. 23, 1979.

[Lang81] G. Langdon, J. Rissanen. Compression of Black-White Images with Arithmetic Coding. *IEEE Transactions on Communications,* Vol. COM-29, June 1981.

[Mann74] J. Mannos, D. Sakrison. The effects of a visual fidelity criterion on the encoding of images. *IEEE Transactions on Information Theory,* Vol. 20, July 1974.

[Mart98] B. Martins and S. Forchhammer. Tree Coding of Bilevel Images. *IEEE Transactions on Image Processing,* Vol. 7, No. 4, April 1998.

[Mart99] B. Martins and S. Forchhammer. Lossless, Near-Lossless, and Refinement Coding of Bilevel Images. *IEEE Transactions on Image Processing,* Vol. 8, No. 5, May 1999.

[Mart00] B. Martins and S. Forchhammer. Halftone Coding with JBIG2. *SPIE Journal of Electronic Imaging,* Vol. 9, No. 1, January 2000.

[Mitc88] J. Mitchell, W. Pennebaker. Software Implementations of the Q-Coder. *IBM Journal of Research & Development,* Vol. 32, No. 6, 1988.

[Mohi84] K. Mohiuddin, J. Rissanen, R. Arps. Lossless Binary Image Compression Based on Pattern Matching. *International Conference on Computers, Systems & Signal Processing,* 1984.

[Mori92] S. Mori, C. Suen, K. Yamamoto. Historical Review of OCR Research and Development. *IEEE Proceedings,* Vol. 80, July 1992.

[Penn88a] W. Pennebaker, J. Mitchell, G. Langdon, R. Arps. An Overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder. *IBM Journal of Research & Development,* Vol. 32, No. 6, 1988.

[Penn88b] W. Pennebaker, J. Mitchell. Probability Estimation for the Q-Coder. *IBM Journal of Research & Development,* Vol. 32, No. 6, 1988.

[Prat80] W. Pratt, P. Capitant, W. Chen, E. Hamilton, R. Wallis. Combined Symbol Matching Facsimile Data Compression System. *Proceedings of the IEEE,* Vol. 68, July 1980.

[Roch92] J. Rocha, T. Pavlidis. A Shape Analysis Model with Applications to a Character Recognition System. *Proceedings of the 1992 IEEE Workshop on Applications of Computer Vision, 1992.*

[Sait91] T. Saitoh, T. Pavlidis. Page Segmentation without Rectangle Assumption. *Proceedings of the 11th International Conference on Pattern Recognition,* Saint Malo, France. 1991.

[Sayo96] K. Sayood. *Introduction to Data Compression.* San Francisco, CA: Morgan Kaufmann, 1996.

[Shan48] C. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal,* Vol. 27, 1948.

[Silv84] D. Silver. Facsimile Coding using Symbol-Matching Techniques. *IEE Proceedings, Part F,* Vol. 131, April 1984.

[Slat98] M. Slattery, J. Mitchell. The Qx-coder. *IBM Journal of Research & Development,* Vol. 42, No. 6, 1998.

[Sull93] J. Sullivan, R. Miller, G. Pios. Image Halftoning using a visual model in error diffusion. *Journal of the Optical Society of America,* Vol. 10, August 1993.

[T6] ITU-T Recommendation T.6. *Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus,* 1988

[T82] ITU-T Recommendation T.82. *Information Technology - Coded Representation of Picture and Audio Information - Progressive Bi-Level Image Compression,* 1993.

[T85] ITU-T Recommendation T.85. *Application Profile for Recommendation T.82 - Progressive Bi-Level Image Compression (JBIG Coding Scheme) For Facsimile Apparatus,* 1995.

[T88] ITU-T Recommendation T.88. *Information Technology — Lossy/Lossless Coding of Bi-Level Images,* 2000.

[Tomp99a] D. Tompkins, F. Kossentini. Lossless JBIG2 Coding Performance. *Proceedings of the 1999 Data Compression Conference,* 1999.

[Tomp99b] D. Tompkins, F. Kossentini. A Fast Segmentation Algorithm for Bi-level Image Compression using JBIG2. *Proceedings of the 1999 International Conference on Image Processing,* Vol. 1, 1999.

[Ulic87] R. Ulichney. *Digital Halftoning.* Cambridge: MIT Press, 1987.

[Vall99] M. Valliappan, B. Evans, D. Tompkins, F. Kossentini. Lossy Compression of Stochastic Halftones with JBIG2. *Proceedings of the 1999 International Conference on Image Processing,* Vol. 1, 1999.

[Witt94a] I. Witten, A. Moat, and T. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* New York: Van Nostrand Reinhold, 1994.

[Witt94b] I. Witten, T. Bell, H. Emberson, S. Inglis, A. Moffat. Textual Image Compression: Two-Stage Lossy/Lossless Encoding of Textual Images. *Proceedings of the IEEE,* Vol. 82, No. 6, June 1994.

# APPENDIX A. ARITHMETIC CODING

Arithmetic coding is an increasingly popular method of compressing data, especially when there are only a few symbols in the alphabet [Lang79,Lang81,Bott98,Howa94]. The original concept of arithmetic coding was hinted at by Shannon, [Shan48] but is usually credited to Peter Elias [Sayo96] and is sometimes referred to as *Elias coding*. Most practical arithmetic coding systems are based upon work by Rissanen [Riss79].

In arithmetic coding, as with most compression systems, the data to compress contains symbols ($a_i$) in an alphabet ($A$)

$$A = \{a_1, a_2, a_3 \ldots a_M\}$$

and the sequence of data

$$x_1, x_2, x_3 \ldots x_N$$

contains elements ($x_k$) which can be any symbol from the alphabet ($A$)

$$x_k = a_i \qquad\qquad a_i \in A$$

With an adaptive system, the probability that a data element ($x_k$) is a specific symbol ($a_i$) is represented as:

$$P(x_k = a_i \mid x_1, x_2 \ldots x_{k-1}) \qquad\qquad a_i \in A$$

For example, consider the following simple system:

$$A = \{1,2,3,4,5\}$$

$$P(x_k = a_i \mid x_1, x_2 \ldots x_{k-1}) = \frac{\sum_{j=1}^{k-1}(x_j = a_i)}{k-1}$$

and the (*k-1=20*) elements of data already coded are

$$x_1, x_2 \ldots x_{20} = 5,2,1,3,4,4,2,4,2,2,5,1,5,4,2,1,1,1,2,4$$

the probability that the next symbol ($x_{20}$) will be 3 may be:

$$P(x_{21} = 3 \mid x_1, x_2 \ldots x_{20}) = \frac{\sum_{j=1}^{20}(x_j = 3)}{20} = \frac{1}{20} = 0.05$$

In arithmetic coding, an interval (*R*) is used

$$R = [R_L, R_H)$$

which is updated after each element

$$\begin{aligned}
&x_k = a_i \\
&R_{k+1} = [R_{L_{k+1}}, R_{H_{k+1}}) \\
&R_{L_{k+1}} = R_{L_k} + (R_{H_k} - R_{L_k}) \cdot \sum_{j=1}^{i-1} P(x_k = a_j) \\
&R_{H_{k+1}} = R_{L_{k+1}} + (R_{H_k} - R_{L_k}) \cdot P(x_k = a_i)
\end{aligned}$$

and only the final interval ($R_{N+1}$) or a value within that interval has to be coded. Arithmetic coding is best illustrated with an example. Consider the following non-adaptive binary system:

$$\begin{aligned}
&A = \{0,1\} \\
&P(x_k = 0) = 0.3 \\
&P(x_k = 1) = 0.7 \\
&R_1 = [0.0,1.0)
\end{aligned}$$

and there are 4 elements of data to be coded:

$$x_1, x_2, x_3, x_4 = 1,0,1,1$$

For the first element of data, the interval changes as follows:

$$x_1 = 1$$
$$R_{L_2} = R_{L_1} + \left(R_{H_1} - R_{L_1}\right) \cdot P\left(x_1 = 0\right) = 0.0 + (1.0 - 0.0) \cdot (0.3) = 0.3$$
$$R_{H_2} = R_{L_2} + \left(R_{H_1} - R_{L_1}\right) \cdot P\left(x_1 = 1\right) = 0.3 + (1.0 - 0.0) \cdot (0.7) = 1.0$$
$$R_2 = \left[R_{L_2}, R_{H_2}\right) = \left[0.3, 1.0\right)$$

and then for the second element,

$$x_1 = 0$$
$$R_{L_3} = R_{L_2} = 0.3$$
$$R_{H_3} = R_{L_3} + \left(R_{H_2} - R_{L_2}\right) \cdot P\left(x_1 = 0\right) = 0.3 + (1.0 - 0.3) \cdot (0.3) = 0.51$$
$$R_3 = \left[R_{L_3}, R_{H_3}\right) = \left[0.3, 0.51\right)$$

and subsequently:

$$R_4 = \left[0.363, 0.51\right)$$
$$R_5 = \left[0.4071, 0.51\right)$$

The following figure illustrates the procedure for calculating the interval ($R$) after each element of data:

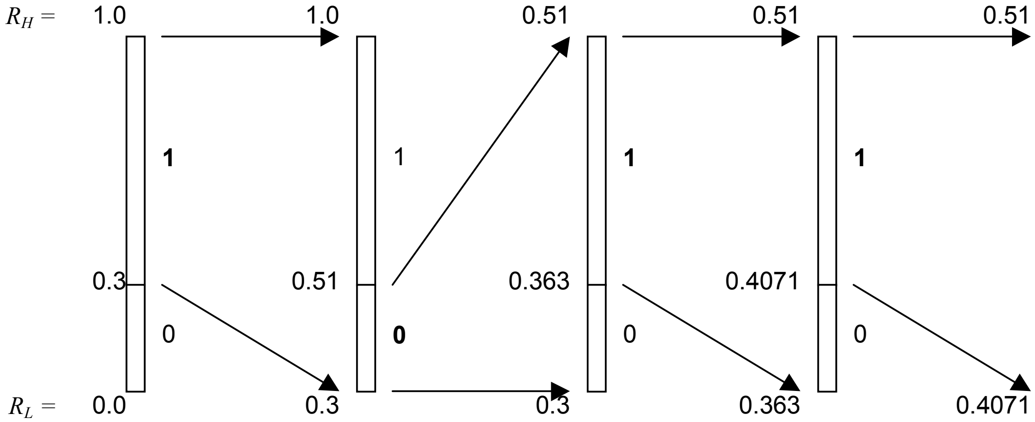


Figure A-1 A Sample Binary Non-Adaptive Arithmetic Coder

To properly code this example, a value that is within the final interval [0.4071,0.51) is chosen, which will uniquely identify this sequence. In this example, the value of 0.5 would suffice.

In an arithmetic coding system, the number of bits required to encode a symbol is

$$-P(x_i)\cdot\log P(x_i)+\varepsilon$$

where (ε) is some extra overhead imposed by the specific arithmetic coding system. For an ideal system with a large number of symbols, the value ε approaches zero, and the total number of bits required to code a sequence approaches the entropy of the system (*H*) which is calculated by:

$$H=-\sum_{j=1}^{N}P(x_i)\cdot\log P(x_i)$$

From this equation, it is clear that the coding efficiency of an arithmetic coding system depends on the probabilities of the data elements. In most systems, the exact probabilities are not precisely known, but are instead calculated estimates based upon what is known about the system. As a result, the efficiency of an arithmetic coder depends upon the quality of the probability estimates.

An ideal arithmetic coding system would require infinitely precise calculations and large quantities of memory. Naturally, practical systems must be designed within some constraints, which reduces the coding efficiency of the system. The reduction in coding efficiency depends on the characteristics of the arithmetic coding system.