Progressive Lossless Image Compression Using Image Decomposition and Context Quantization

by

Hui Zha

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2007

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Hui Zha

Abstract

Lossless image compression has many applications, for example, in medical imaging, space photograph and film industry. In this thesis, we propose an efficient lossless image compression scheme for both binary images and gray-scale images. The scheme first decomposes images into a set of progressively refined binary sequences and then uses the context-based, adaptive arithmetic coding algorithm to encode these sequences. In order to deal with the context dilution problem in arithmetic coding, we propose a Lloyd-like iterative algorithm to quantize contexts. Fixing the set of input contexts and the number of quantized contexts, our context quantization algorithm iteratively finds the optimum context mapping in the sense of minimizing the compression rate. Experimental results show that by combining image decomposition and context quantization, our scheme can achieve competitive lossless compression performance compared to the JBIG algorithm for binary images, and the CALIC algorithm for gray-scale images. In contrast to CALIC, our scheme provides the additional feature of allowing progressive transmission of gray-scale images, which is very appealing in applications such as web browsing.

Acknowledgments

I would like to thank my supervisor and mentor Professor En-hui Yang for his academic guidance, financial support, and patience throughout my study in the University of Waterloo.

I would like to thank Professor En-hui Yang, Professor Weihua Zhuang, Professor Amir K. Khandani and Professor Xuemin Shen for offering me excellent courses at the University of Waterloo.

I would like to thank Dr. Dake He for his advices and discussions on my research and thesis.

I would like to thank Professor Liang-liang Xie, Professor Zhou Wang for being my thesis readers, and for their suggestions on my thesis.

I would like to thank my friends and colleagues in the Multimedia Communication laboratory at the University of Waterloo. They are Dr. Guixing Wu, Dr. Haiquan Wang, Dr. Xiang Yu, Dr. Xudong Ma, Lin Zheng, Jiao Wang, Jin Meng, Yuhan Zhou, Abir Mukherjee, and Jingming Xu.

I would like to thank my friends Jing Wang and Guixing Wu, Jing Jiang and Chao Yan for their warm helps anytime I need.

Finally, I would like to thank Zhenmei Gu and Dake He for their sincere support. I would like to my parents, my parents-in-law, and my wife Daan for their deep love to me all the time.

Contents

List of Tables

List of Figures

Acronyms

CALIC Context-based, Adaptive, Lossless Image Codec.

GAP Gradient-Adjusted Prediction.

HDTV High-Definition Television.

JBIG Joint Bi-level Image Experts Group.

JPEG Joint Photographic Experts Group.

JPEG-LS JPEG-Lossless.

MCECQ Minimum Conditional Entropy Context Quantizer.

MED Median Edge Detector.

MSE Mean-Squared-Error.

PSNR Peak-Signal-To-Noise-Ratio.

WWW World Wide Web.

Chapter 1

Introduction

1.1 Digital Images

A two-dimensional (2-D) digital image consists of a finite number of rows and columns of *pixel*s (short for **pic**ture **el**ements). Let H denote the height or number of rows and W denote the width or number of columns of an image. A 2-D digital image can be represented by a two-dimensional matrix:

$$\mathbf{I} = \{I(i,j)\} = \begin{pmatrix} I(0,0) & I(0,1) & \cdots & I(0,W-1) \\ I(1,0) & I(1,1) & \cdots & I(1,W-1) \\ \vdots & \vdots & \ddots & \vdots \\ I(H-1,0) & I(H-1,1) & \cdots & I(H-1,W-1) \end{pmatrix}, \tag{1.1}$$

where the first coordinate i $(0 \leq i < H)$ denotes the row index along the vertical direction, the second coordinate j $(0 \leq j < W)$ denotes the column index along the horizontal direction, and $I(i,j)$ denotes the value of the pixel located in row i and column j, or more concisely, at position (i,j). As a convention, we use term "image $\mathbf{I}$" to denote the image with pixel value matrix $\mathbf{I}$, and use the regular I to denote the pixel values of image $\mathbf{I}$.

By using raster scanning order, an image $\mathbf{I}$ of size $H \times W$ can be transformed into a discrete one-dimensional sequence

$$X = \{X_k\}_{k=1}^{n} = \{X_1, X_2, \cdots, X_n\}, \tag{1.2}$$

where $n = H \times W$ is the number of pixels of image $\mathbf{I}$ and $1 \leq k \leq n$. The conversion between the one-dimensional coordinate k and the two-dimensional coordinate (i,j) is given by

$$k = i \times W + j + 1 \tag{1.3}$$

and

$$i = \text{the quotient of } \frac{k-1}{W} \tag{1.4a}$$

$$j = \text{the remainder of } \frac{k-1}{W}. \tag{1.4b}$$

The number of bits used to represent a single pixel is called *bit depth* or *color depth*. If an image is b bits per pixel (bpp), it is also called a b-bit image. A b-bit image can represent 2^b different gray levels or colors. Three typical types of images involved in this thesis are:

1. **1-bit *bi-level* images**. A bi-level image contains two different colors which are usually black and white. A bi-level image is called in the *standard* form in this thesis if the image uses bit 0 to represent one of the two colors and uses bit 1 to represent the other color. Since bi-level images in other forms can be converted into the standard form, we only consider the bi-level images in their standard form. Bi-level images are also referred to as *binary images* or black and white images in the literature.

2. **8-bit *gray-scale*[1] images**. An 8-bit gray-scale image contains up to 256 different shades of gray with values from 0 to 255. Sometimes one can interpret the pixel values of gray-scale images as indices to colors in a color palette. In this thesis, pixel values of gray-scale images are regarded as shades of gray from black to white.

3. **24-bit true color images**. A 24-bit true color image in RGB color space consists of three gray-scale components of red, green and blue, each of which is represented by 8 bits. An image in RGB space can be converted into other color spaces such as YCbCr and YUV [1, 2]. In YCbCr or YUV space, luminance (brightness) information is stored as a single component Y, and chrominance (color) information is stored as two different components: C_b and C_r for YCbCr space; U and V for YUV space.

1.2 Motivations

Uncompressed images normally require a large amount of storage capacity and transmission bandwidth. For example, a 24-bit true color high-definition television

[1]Gray-scale is also referred to as grayscale, gray scale, or gray-level in the literatures.

(HDTV) image with size 1920×1080 needs approximately 6 megabytes (6M bytes) of storage space. On the other hand, various types of redundancy exist in images, such as temporal redundancy, spatial redundancy (or interpixel redundancy), coding redundancy, spectral redundancy and psychovisual redundancy. The primary goal of image compression is to minimize the number of bits required to represent the original images by reducing the redundancy in images, while still meeting the user-defined quality requirements. The core issue in image compression is to design efficient and effective compression schemes.

In terms of reconstruction ability, image compression schemes can be broadly classified into two major categories: *lossless* image compression and *lossy* image compression. Lossless image compression refers to the compression in which the original image can be fully reconstructed from the compressed image, *i. e.*, no loss of any information during the compressing. Lossy image compression, on the contrary, allows some kind of difference or distortion between the reconstructed image and the original image, and the original image can not be fully recovered.

Lossless image compression has many applications such as medical imaging, space photograph, and film industry. In practice, medical images must be represented flawlessly for medical professionals to make clinical diagnosis with accuracy. Any minor distortion or errors introduced by lossy compression may lead to serious consequences to patients. In space exploration, astronomical images obtained by the satellites are often compressed and transmitted back to the earth for later processing such as object identification and feature extraction. These processing procedures may "amplify" the distortion caused by lossy compression and thus produce false results due to the distortion. In such cases, lossy compression is not appropriate because original astronomical images are very difficult and expensive to be obtained again. Finally, in film industry, lossless compression also has a huge market in archiving the films in order to save storage and meanwhile maintain the original high quality for future editing and re-compressing.

Recently, *progressive image compression* or *progressive image transmission* [3, 4, 5, 6, 7] has received significant attention due to the popularity of applications such as web browsing. When one views a large set of images on the World Wide Web (WWW) over relatively slow network links, he/she would like to quickly get a coarse recognition of what an image will be, and then choose to continue or stop the transmission of that image according to his/her interest. This goal can be achieved by progressive image compression which allows gradually enhancing an image via a finite number of sequences. When the first sequence or the first few sequences, which are supposed to take very little space, are transmitted to the decoder, the decoder

can rapidly build up an approximate image with lower quality or smaller size, and then progressively refine the image with more details by decoding the rest of the sequences. The process continues until lossless recovery is accomplished. Kieffer and Yang [8] generalized such kind of coding as "refinement source coding". Image refinement can take many different fashions, for example, pixel value refinement, image size refinement, spatial resolution refinement, and image objects or layers refinement. Main progressive image compression schemes include down-sampling, vector quantization, wavelet transform coding and pyramid coding, *etc.* Due to its appealing viewing on-the-fly feature, progressive image compression is included as a part of recent image compression standards such as JPEG (Joint Bi-level Image Experts Group) [9], JBIG2 [10, 11], JPEG (Joint Photographic Experts Group) [12, 13] and JPEG 2000 [14, 15, 16].

In recent years, there have been many research efforts in lossless compression of bi-level images [9, 10] and gray-scale images [17, 18, 19, 20, 21, 22, 23, 24, 25]. However, existing gray-scale image compression schemes either do not provide the progressive image transmission feature, or with this feature, but the compression performance is not as good as CALIC (Context-based, Adaptive, Lossless Image Codec) [17], which is widely accepted as the state-of-the-art scheme for gray-scale image compression. In addition, existing gray-scale image compression schemes usually do not perform well in compressing bi-level images. Therefore, this thesis will emphasize on the development of an efficient scheme for both lossless compression of bi-level images and progressive lossless compression of gray-scale images. We employ two methods to achieve our objective: context quantization and image decomposition, which will be discussed in Chapter 3 and Chapter 4, respectively. Roughly speaking, context quantization is to reduce the number of contexts in order to deal with the context dilution problem existed in context-based arithmetic coding; image decomposition is to provide the progressive feature. By elegantly combining these two methods in the same scheme, we obtain competitive lossless compression performance, compared to JBIG [9] for binary images, and to CALIC for gray-scale images. Moreover, we offer the progressive image transmission feature in contrast to CALIC.

1.3 Performance Measure

Appropriate performance metrics are required to evaluate the performance of a specific image compression scheme. An image compression algorithm can be evalu-

ated in many different ways according to different requirements. Major compression metrics include compression ratio, compression speed, computing complexity, memory and storage complexity, objective and subjective quality of the reconstructed image, *etc.*

The most common metric of performance measure of an image compression scheme is the *compression ratio*, which is defined by

$$\text{compression ratio} = \frac{\text{original image size in bits}}{\text{compressed image size in bits}}, \tag{1.5}$$

i. e., the ratio of the number of bits to represent the original image data to the number of bits to represent the compressed image data.

The original image size used in this thesis does not include any image format overhead or byte alignment overhead. For an image of size $W \times H$ and bit depth b, the original image size is simply calculated by the following formula

$$\text{original image size in bits} = W \times H \times b, \tag{1.6}$$

On the contrary, the compressed image size counts all header or tail overhead needed to reconstruct the original image. For example, if an image of size 512×512 with 8 bits per pixel is compressed to 8192 bytes, the compression ratio will be $32 : 1$ or 32.

Another way to evaluate image compression performance is to use *compression rate* in the unit of *bits per sample* or *bits per pixel* (bpp), which is defined as the average number of bits used to represent a single sample (pixel). The compression rate is given by

$$\text{compression rate} = \frac{\text{compressed image size in bits}}{\text{number of pixels}}. \tag{1.7}$$

In the above example, we say that the original rate is 8 bpp and the compression rate is $\frac{8192 \times 8}{512 \times 512} = 0.25$ bpp.

We note that a larger value of compression ratio or a smaller value of compression rate indicates better compression performance of a compression scheme.

Although reconstruction quality is irrelevant to the lossless compression schemes, it is important to lossy compression. For lossy image compression, we can not justify a scheme by considering the compression ratio or compression rate alone, because the scheme which gives higher compression ratio may result in worse reconstruction

quality. The most widely used objective quality metric for lossy image compression is the peak-signal-to-noise-ratio (PSNR) , which is given by

$$\text{PSNR(dB)} = 10 \log_{10} \left(\frac{\phi_{\max}^2}{\text{MSE}} \right), \tag{1.8}$$

where $\phi_{\max}$ is the peak (maximum) pixel value of the image. For 8-bit gray-scale images, we have $\phi_{\max} = 255$. The mean-squared-error (MSE) denotes the *mean-squared-error* which is defined by

$$\text{MSE} = \frac{1}{WH} \sum_{i=1}^{H-1} \sum_{j=1}^{W-1} [I(i,j) - \hat{I}(i,j)]^2, \tag{1.9}$$

where $I(i,j)$ and $\hat{I}(i,j)$ refer to the pixel values of the original image and the reconstructed image, respectively, both at position (i,j).

In this thesis, we will use compression rate as the major measure of lossless compression performance.

1.4 Organization

The rest of the thesis is organized as follows. In Chapter 2, we briefly review the information theory preliminaries and major lossless image compression schemes in the literature.

In Chapter 3, we propose a Lloyd-like iterative algorithm for context quantization in order to address the context dilution problem in context-based arithmetic coding. Fixing the set of input contexts and the size of quantized contexts, the proposed context quantization algorithm iteratively finds the optimum context mapping in the sense of minimizing the compression rate. We also discuss two algorithm implementation problems: the initial codebook generation problem and the empty cell problem.

In Chapter 4, we propose an image decomposition method based on color splitting. The method decomposes an image into a set of binary sequences which are suitable for context quantization and arithmetic coding. Meanwhile, the decomposition enables the progressive transmission feature in our proposed image compression scheme.

In Chapter 5, we present the experimental results of our proposed scheme and other representative schemes on the compression of binary images and gray-scale

images. We show that the proposed scheme achieves competitive lossless compression performance on both binary and gray-scale images compared to those widely accepted state-of-the-art compression schemes.

Finally, in Chapter 6, we summarize the thesis and suggest the future work.

Chapter 2

Preliminaries and Literature Review

In this chapter, we will first review some basic information theory concepts which will be used for later chapters. The second part of this chapter includes the literature review of several representative lossless image compression schemes.

2.1 Preliminaries

From the information theory point of view, image compression belongs to the area of *source coding* which is originated to Claude E. Shannon in his fundamental paper "A mathematical theory of communication" in 1948 [26]. In this section, we will briefly review the information theory preliminaries to please our discussion on the image compression.

Throughout this thesis, we shall use the following notation. Let $\mathcal{A}$ be a finite alphabet[1] and let $|\mathcal{A}|$ stand for the cardinality of $\mathcal{A}$. We consider only the non-trivial cases where $|\mathcal{A}| \geq 2$. A sequence from $\mathcal{A}$ is called an $\mathcal{A}$-sequence. For any $\mathcal{A}$-sequence x, $|x|$ denotes the length of x. Let $\mathcal{A}^n$ denote the set of all sequences of length n from $\mathcal{A}$. For brevity, an $\mathcal{A}$-sequence $X_m X_{m+1} \cdots X_n \in \mathcal{A}^{n-m+1}$ may be written as X_m^n, where m and n are positive integers satisfying $m < n$. Besides, all logarithms are to the base 2 unless otherwise specified.

[1] In this thesis, we are concerned only with sources with finite alphabets because digital images are generally with finite alphabets.

2.1.1 Information Sources

In the source coding theory, data sequences to be encoded are generally assumed to be emitted from some information source.

Definition 2.1 (Information Source)**.** A (discrete) alphabet $\mathcal{A}$ *information source* is defined as a discrete time random process $X = \{X_i\}_{i=1}^{\infty} = \{X_1, X_2, \cdots\}$ associated with a probability space $(\mathcal{A}^{\infty}, \mathfrak{F}, P_X)$, where $\mathfrak{F}$ is called a *σ-field* and P_X is a *probability measure* on $(\mathcal{A}^{\infty}, \mathfrak{F})$ [27].

An alphabet $\mathcal{A}$ source $X = \{X_i\}_{i=1}^{n}$ of a finite length n is totally characterized by the joint distribution

$$P_X(X_1^n = x_1^n) = \Pr(X_1 = x_1, X_2 = x_2, \cdots, X_n = x_n) = p(x_1^n), \tag{2.1}$$

where $x_1^n \in \mathcal{A}^n$. Note that we denote $P_X(X = x)$ by the probability mass function $p(x)$ for convenience.

Definition 2.2 (Stationary Source)**.** An alphabet $\mathcal{A}$ source is said to be a *stationary source* if the joint distribution of any subset of the random sequence $X = \{X_i\}_{i=1}^{\infty}$ satisfies

$$\Pr\{X_1^n = x_1^n\} = \Pr\{X_{1+m}^{n+m} = x_1^n\} \tag{2.2}$$

for all positive integers m and all $x_1^n \in \mathcal{A}^n$.

Definition 2.3 (I.I.D. Source)**.** An alphabet $\mathcal{A}$ source is said to be an *independent and identically distributed (i.i.d.) source* if

$$\Pr\{X_n = x_n | X_1^{n-1} = x_1^{n-1}\} = p(x_n) \tag{2.3}$$

for all $n \geq 1$, where the probability mass function $p : \mathcal{A} \to [0, 1]$ satisfies $\sum_{a \in \mathcal{A}} p(a) = 1$, and the initial symbol $X_0 = x_0 \in \mathcal{A}$ is assumed known.

Definition 2.4 (Markov Source)**.** Let k be a positive integer. An alphabet $\mathcal{A}$ source is said to be a kth-order *Markov source* if

$$\Pr\{X_n = x_n | X_1^{n-1} = x_1^{n-1}\} = p(x_n | x_{n-k}^{n-1}) \tag{2.4}$$

for all $n > k$, where the transition probability function $p : \mathcal{A}^k \times \mathcal{A} \to [0, 1]$ satisfies $\sum_{a_{k+1} \in \mathcal{A}} p(a_{k+1} | a_1^k) = 1$ for all $a_1^k \in \mathcal{A}^k$, and the initial symbol $X_0 = x_0 \in \mathcal{A}$ is assumed known.

Definition 2.5 (Finite-State Source)**.** Let $\mathcal{C} = \{c_1, c_2, \cdots, c_{|\mathcal{C}|}\}$ be a finite set of size $|\mathcal{C}| = m$ where each element $c \in \mathcal{C}$ is called a *state*. Let $S = \{S_i\}_{i=1}^n = \{S_1, S_2, \cdots, S_n\}$ be a random sequence which takes values in $\mathcal{C}$ and corresponds to an alphabet $\mathcal{A}$ source $X = \{X_i\}_{i=1}^n$. Source X is said to be a *finite-state source with m states*

$$\Pr\{X_1^n = x_1^n, S_1^n = s_1^n\} = \prod_{i=1}^n p(x_i, s_i|s_{i-1}) \tag{2.5}$$

for all $n \geq 1$ and all $x_1^n \in \mathcal{A}^n$, where the transition probability function $p : \mathcal{C} \times (\mathcal{A} \times \mathcal{C}) \to [0, 1]$ satisfies $\sum_{a\in\mathcal{A}} \sum_{c\in\mathcal{C}} p(a, c|c') = 1$ for all $c' \in \mathcal{C}$, and the initial state $S_0 = s_0 \in \mathcal{C}$ is assumed known.

Remark 2.1. It is easy to verify that i.i.d. sources are special cases of Markov sources and finite-state sources.

Remark 2.2. For the finite alphabet case, finite-state sources are more general than kth-order Markov sources in the sense as follows.

1. A kth-order Markov source with a finite alphabet belongs to the class of finite-state sources.

2. A kth-order Markov source takes the prefix sequence x_{i-k}^{i-1} of x_i as its state s_i' at time instant i, while the state s_i of a finite-state source is an abstract state which can be any function of the prefix sequence x_1^{i-1} at time instant i and the previous state s_{i-1}:

$$s_i = f(x_1^{i-1}, s_{i-1}),\ i = 1, 2, \cdots, n, \tag{2.6}$$

   where the initial state $s_0 \in \mathcal{C}$ is assumed known and the state function f is defined as the mapping

$$f : \mathcal{A}^{i-1} \times \mathcal{C} \to \mathcal{C}. \tag{2.7}$$

   In that sense, the abstract state s_i of a finite-state source takes the state $s_i' = x_{i-k}^{i-1}$ as a special case.

Markov sources and finite-state sources are widely adopted to model data sources in data compression. In such cases, the prefix sequences (given a specific sequence scanning order) or states are often called *contexts*. In this thesis, we will use finite-state sources to characterize images due to the generality of finite-state sources for most situations in practice.

2.1.2 Shannon Entropy and Entropy Rate

Let X and Y be two discrete random variables with alphabet $\mathcal{A}$ and $\mathcal{B}$, respectively.

Definition 2.6 (Shannon Entropy)**.** The *Shannon entropy* or simply *entropy* $H(X)$ of X is defined by

$$H(X) = -\sum_{x\in\mathcal{A}} p(x)\log p(x). \tag{2.8}$$

The unit of entropy is bits per symbol if we take the logarithm to the base 2.

Definition 2.7 (Joint Entropy)**.** The *joint entropy* $H(X,Y)$ of two discrete random variables (X,Y) with a joint probability distribution $p(x,y)$ is defined as

$$H(X,Y) = -\sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p(x,y)\log p(x,y). \tag{2.9}$$

Definition 2.8 (Conditional Entropy)**.** For two discrete random variables (X,Y) with joint distribution $p(x,y)$, the *conditional entropy* $H(X|Y)$ is defined as

$$\begin{aligned} H(X|Y) &= -\sum_{y\in\mathcal{B}} p(y)H(X|Y=y) && (2.10)\\ &= -\sum_{y\in\mathcal{B}} p(y)\sum_{x\in\mathcal{A}} p(x|y)\log p(x|y) && (2.11)\\ &= -\sum_{y\in\mathcal{B}}\sum_{x\in\mathcal{A}} p(x,y)\log p(x|y). && (2.12) \end{aligned}$$

An important result about conditional entropy is given by the following theorem [28].

Theorem 2.1 (Conditioning reduces entropy, Theorem 2.6.5 in [28])**.** *For any two random variables X and Y, we have*

$$H(X|Y) \le H(X) \tag{2.13}$$

with equality if and only if X and Y are independent.

Remark 2.3. Theorem 2.1 implies that if a source can be modeled as a Markov source or a finite-state source, we can reduce the source entropy by conditioning on some states or contexts correlated with the source. However, in practice, too many states or contexts will affects the compression efficiency due to the storage problem and context dilution problem [29, 17].

Definition 2.9 (Entropy Rate)**.** The *entropy rate* of a source $X = \{X_i\}_{i=1}^{\infty}$ is defined as

$$H(\mathcal{X}) = \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, \ldots, X_n) \tag{2.14}$$

when the limit exists.

Remark 2.4. In the data compression scenario, the entropy rate is the theoretical data compression limit.

For a stationary source X, we have [28],

$$\lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, \ldots, X_n) = \lim_{n\to\infty} H(X_n|X_1^{n-1}), \tag{2.15}$$

where $H(X_n|X_1^{n-1})$ is the conditional entropy of X_n given X_1^{n-1} and we use the convention that $H(X_1|X_0) = H(X_1)$.

In light of the chain rule for entropy [28]

$$H(X_1, X_2, \ldots, X_n) = H(X_1) + H(X_2|X_1) + \cdots + H(X_n|X_1^{n-1}), \tag{2.16}$$

we have

$$\begin{aligned} H(\mathcal{X}) &= \lim_{n\to\infty} \frac{1}{n} (H(X_1) + H(X_2|X_1) + \cdots + H(X_n|X_1^{n-1})) \\ &= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} H(X_i|X_1^{i-1}) \qquad (2.17) \\ &= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} -\log p(x_i|x_1^{i-1}). \qquad (2.18) \end{aligned}$$

Remark 2.5. The significance of Equation (2.17) lies in that we can use the time average of the conditional entropies given all the past sequences to asymptomatically achieve the entropy rate of a stationary source. Thus, to encode an $\mathcal{A}$-sequence $X = \{X_1, X_2, \cdots\}$, from Equation (2.18) we shall assign a codeword with length $-\log p(x_i|x_1^{i-1})$ to encode the sample X_i. The (average) compression rate $r = -\frac{1}{n} \log \prod_{i=1}^{n} p(x_i|x_1^{i-1})$ in bits per symbol can be achieved by using an adaptive arithmetic coding algorithm when n is large enough.

2.1.3 Relative Entropy

Definition 2.10. The *relative entropy* or *Kullback Leibler distance* (K-L distance) between two probability mass functions $p(x)$ and $q(x)$ is defined as [28]

$$D(p\|q) = \sum_{x\in\mathcal{A}} p(x) \log \frac{p(x)}{q(x)}. \tag{2.19}$$

Remark 2.6. We see that $D(p\|q)$ is a measure of the distance or the "distortion" between two distributions p and q.

If $X \sim p(x)$, from Equation (2.19), we have

$$\begin{aligned} D(p\|q) &= -\sum_{x\in\mathcal{A}} p(x)\log q(x) - \left(-\sum_{x\in\mathcal{A}} p(x)\log p(x)\right) \\ &= H_q(X) - H_p(X), \end{aligned} \tag{2.20}$$

where $H_p(X)$ is the entropy of X and $H_q(X) = -\sum_{x\in\mathcal{A}} p(x)\log q(x)$ is the average codeword length if we assign a codeword of $-\log q(x)$ bits to represent symbol x.

Remark 2.7. From Equation (2.20), we see that if the true distribution of a random variable X is p and we construct a code for distribution q instead, we would need $D(p\|q)$ more bits on the average to represent X.

2.2 Literature Review

Most image compression schemes are based on the framework of predictive coding and context modeling followed by entropy coding [30]. Some heuristic prediction methods have been developed for gray-scale images, for example, the median edge detector (MED) used in JPEG-LS standard [31] standardized by the JPEG [13], the gradient-adjusted prediction (GAP) used in CALIC algorithm [17], *etc.* Recently, context modeling for image compression has been received more and more attention [17]. Combined with prediction, context modeling for the predictive residuals (or errors) is performed based on selected context templates of causal pixels. Finally *arithmetic coding* [32, 33] or *huffman coding* [34] is used to encode those predictive residuals conditioned on the constructed contexts.

From the information theory point of view, prediction is essentially a context modeling technique. A challenging issue for image compression is how to perform context modeling efficiently and effectively. On one hand, theoretically conditioning on more information will reduce the source entropy if the conditioned information has correlations to the sequence to be encoded (see Theorem 2.6.5 in [28]). Thus we tend to use as many contexts as possible. On the other hand, too many contexts will cause two main problems [29]: the storage problem and the *context dilution* problem. The first problem is that a large number of contexts requires a large amount of memory/disk storage. The second problem is that the conditional probabilities can hardly be accurately estimated during the model learning process when

statistics are spread over too many contexts [35]: in practice available sample pixels are insufficient to build a high order context model with reasonably steady statistic states. The two problems, also referred as to the "modeling cost", limit the practical use of high order context models.

There are two approaches to reduce the modeling cost for image compression. The first approach is predictive coding which predicts the current encoding pixel value based on neighboring pixels in a linear or non-linear manner. Normally the prediction utilizes some *a priori* knowledge of images such as the smooth structure in certain areas of images; thus the modeling cost associated with the model learning process is reduced. The prediction parameters can be chosen empirically or obtained by optimization processes in some defined sense, *e. g.*, minimizing the MSE of prediction values.

Another approach is *context quantization* [36] which is similar to the vector quantization [37, 38] in signal processing and lossy data compression to some degree. The principle of context quantization is to group the original contexts into a smaller number of context sets and thus to reduce the number of contexts handled by entropy coders such as arithmetic coders. A heuristic context quantizer is used in [17], where several local pixel values and their linear combinations are formed as a vector $c = \{a_0, a_1, \cdots, a_{K-1}\}$ $(K \geq 1)$ for the current pixel to be encoded. Each of a_i $(i = 0, 1, \cdots, K-1)$ takes values from $[0, 255]$ for 8-bit gray-scale images, which makes the total number of possible contexts very large. In [17], each a_i is compared to the prediction value of the current pixel. According to the comparison result, a binary value is assigned to the corresponding component of the vector c to form a binary vector $b = \{b_0, b_1, \cdots, b_{K-1}\}$, where $b_j = 0$ or 1 for all $j = 0, 1, \cdots, K-1$. In this way, the number of possible contexts is reduced from 256^K to 2^K.

These heuristic quantization methods have yielded some good lossless compression results; however, these methods are not necessarily optimal in the sense of minimizing the compression rate. Recently, Wu *et al.* [36] present a minimum conditional entropy context quantizer (MCECQ) to minimize the conditional entropy given the conditional probability density function of the sources. They further propose to use dynamic programming to solve the optimization problem for sources with binary alphabets. In MCECQ, the problem is formulated without taking the modeling cost into account explicitly. In addition, the conditional probability density functions are assumed known, which is not the case in practice.

In the following, we briefly review several important lossless image compression schemes in binary image compression and gray-scale image compression.

Bi-level Image Compression Main image compression standards for bi-level images are JBIG [9] and JBIG2 [10] which achieve highly efficient results for facsimile documents compression.

Bi-level images usually have a lot of local structure. In some local part of the document, if most pixels in the neighborhood of the current pixel to be encoded are white, we can guess that the current pixel is also white. JBIG and JBIG2 both employ the local structure or patterns in the neighborhood of the current pixel and construct different contexts to skew the probability distribution, which is suitable for the context-based arithmetic coding. To adapt to different local structure, JBIG uses several different causal context templates to construct contexts.

For a document containing various type of areas such as symbol regions, halftone regions and generic regions, JBIG usually has worse compression performance compared to documents containing only text. To further improve the compression performance for that type of documents, the JBIG2 standard allows the encoder to select the compression technique that would provide the best performance for the type of data. This is a simple way of using *a priori* knowledge of the image to improve the compression performance. In addition, JBIG2 provides lossy image compression of bi-level images.

Gray-Scale Image compression Two major types of lossless gray-scale image compression methods are statistical modeling (or context modeling) based entropy coding method and string matching based (or grammar-based) entropy coding method. Among all the entropy coding algorithms, the context-based arithmetic coding is the most widely used algorithm in current major lossless image compression schemes due to its coding efficiency and affordable computing complexity.

The process of estimating the source statistics through statistical (or context) models is called context modeling. The models could be known in advance or be constructed during the encoding process. Arithmetic coding algorithms are very suitable to use the statistical models and yields good compression performance. The main problem is how to determine the contexts to be used. If we use a small number of contexts, we may not capture the "true" statistics of the source such that the compression performance is not good; while if we use a large number of contexts, we are encountering memory/storgage and context dilution problems. Two possible ways to deal with the problem are predictive coding and context quantization. Two representative methods are JPEG-LS (JPEG-Lossless) algorithm and CALIC algorithm which use both prediction and context quantization in a heuristic way.

The idea of grammar-based coding, in short, is to transform the data sequence into grammars by string matching mechanism and then use arithmetic coding to encode a set of generated grammars. The grammar-based coding is actually a very broad class of compression algorithms. For example, block codes and Lempel-Ziv codes are special cases of grammar-based codes. Grammar-based codes can be divided into two types: context-free and context-dependent [39]. Well designed context-free and context-dependent grammar-based codes can achieve very high compression performance with affordable computation complexity and storage complexity. Although grammar-based codes are developed for universal coding, they can also be applied on image compression and achieve comparable compression performance compared to the algorithms designed specifically for the images sources.

Chapter 3

Context Quantization

In general, images can be assumed to be emitted by some information sources. In this chapter, we first introduce a general context model for information sources. For image compression purpose, we then assume that the Markov type property holds for image sources to some degree, and from that assumption, we will introduce a K-pixel template context model. Based on that model, we will propose a Lloyd-like iterative context quantization algorithm to deal with the context storage problem and dilution problem existed in the arithmetic coding which utilizes a large number of contexts.

3.1 General Context Models

In this section, we introduce a general context model [39] for information sources.

Definition 3.1 (General Context Model)**.** Let $X = \{X_i\}_{i=1}^n$ be an alphabet $\mathcal{A}$ source. Let $\mathcal{C} = \{c_1, c_2, \cdots, c_{|\mathcal{C}|}\}$ be a finite *context set*[1] of size $|\mathcal{C}| = M$ where each element $c \in \mathcal{C}$ is called a *context*. Let $S = \{S_i\}_{i=1}^n = \{S_1, S_2, \cdots, S_n\}$ be a sequence which takes values in $\mathcal{C}$ and corresponds to the source X. S is called the *accompanying context sequence* of X. Each S_i is called an *accompanying context* or simply *context* of symbol X_i. A set of *context generating functions* $f = \{f_i\}_{i=1}^n$ corresponding to the sequence $\{X_i\}_{i=1}^n$ are defined as the mapping

$$f_i : \mathcal{A}^{i-1} \times \mathcal{C} \to \mathcal{C}, \tag{3.1}$$

which is written as

$$S_i = f_i(x_1^{i-1}, S_0) \tag{3.2}$$

[1] In this thesis, we consider finite context sets only.

where $i = 1, 2, \cdots, n$ and $S_0 = s_0 \in \mathcal{C}$ is called an *initial context* and assumed known. The 4-tuple $< \mathcal{A}, \mathcal{C}, f, S >$ is called a *general context model* for the source X.

In a general context model, the accompanying context sequence S of a source X can be built by learning on the past sequences (given a specific scanning order) through Equation (3.2), or determined by utilizing some *a priori* knowledge about the source, or by both. For example, for a text sequence $X = \{X_i\}_{i=1}^n$, one way to build the context model is to use an adaptive arithmetic coder to encode each X_i conditioning on X_1^{i-1}. For image compression, it is widely believed that there are strong correlations among the pixels close to the pixel to be encoded, which can be taken as *a priori* knowledge to improve the compression performance.

3.2 K-pixel Template Context Model

For our image compression purpose, we define a *K-pixel template context model* in this section. Through the rest of this thesis, we may sometimes call the pixel to be encoded as the *current pixel*.

Definition 3.2 (K-pixel Template)**.** A *K-pixel (context) template* or simply a *template* T of an image $\mathbf{I}$ is defined as a 2-D pattern consisting of K geometric locations of pixels. A template for the current pixel x_k is denoted as T_{x_k}. Each member pixel v_j ($j = 1, 2, \cdots, K$) belonging to T_{x_k} (denoted as $v_j \tilde{\in} T_{x_k}$) is called a *context pixel* for x_k. The number of pixels K in the template is called the *order* of the template.

Remark 3.1. The member pixels $v_j \tilde{\in} T_{x_k}$ usually consists of those pixels geometrically close to the current pixel. In that sense, we sometimes call the pixels belonging to the template T_{x_k} as *neighboring pixels* of the pixel x_k.

Remark 3.2. At the edge areas of an image, some locations in a given template may lie outside an image. In such case, we assume that there are "virtual" pixels at those locations, and the values of those "virtual" pixels are fixed.

Figure 3.1 shows an example of a 12-pixel template. The current pixel is labeled by "?". The template consists of 12 locations labeled by number 1 to 12 according to their 2-norm distance to the pixel "?". The corresponding pixels I_i ($i = 1, 2, \cdots, 12$) are context pixels for the pixel "?".

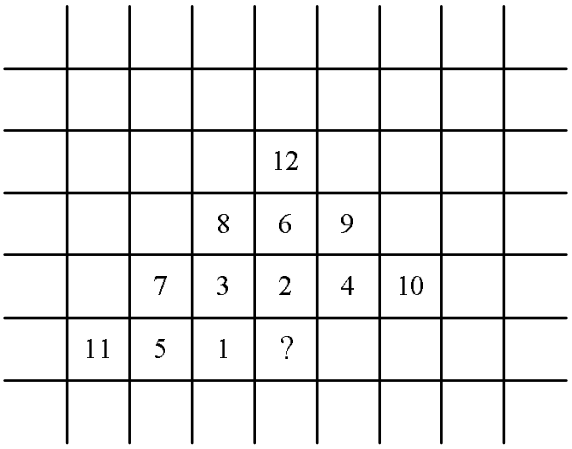

Figure 3.1: A 12-pixel context template

Note that in Figure 3.1, not all pixels surrounding the current pixel "?" are available when we encode the pixel "?" given a specific scanning order. If the raster scanning order is used, we can convert the image from a 2-D matrix to a 1-D sequence, and the mappings between the 2-D pixel indices and 1-D pixel indices are given by Equation (1.3) and Equation (1.4). To encode a pixel X_k with the 1-D pixel index k, all pixels with indices $k' < k$ are available to both the encoder and decoder, and all pixels with indices $k'' > k$ are available to the encoder but not available to the decoder yet. The context template consists of only pixels with indices $k' < k$ is called a *causal context template* for pixel X_k; the context template includes at least one of the pixels with indices $k'' > k$ is called a *non-causal context template*. Traditional image compression schemes which use a raster scanning order and encode pixel by pixel can only use causal templates.

For a K-pixel template, the K-pixel template context is defined as follows.

Definition 3.3 (K-pixel Template Context)**.** Let $\mathcal{A}$ denote the image alphabet. Let T denote a template. Let $v_i \in \mathcal{A}$ $(i = 1, 2, \cdots, K)$ denote the K context pixels belonging to T. A *K-pixel template context* c_k for the current pixel X_k is defined as the vector

$$c_k = \{v_1, v_2, \cdots, v_K\},\ c_k \in \mathcal{A}^K. \tag{3.3}$$

for a given scanning order of the context pixels.

For the example shown in Figure 3.1, We can form the vector $c = \{I_1, I_2, \cdots, I_{12}$ as the context for the pixel "?", where I_i $(i = 1, 2 \cdots, 12)$ denote the pixel values at corresponding locations.

Remark 3.3. In the image compression, the alphabet $\mathcal{A}$ is generally an integer set. For example, for 8-bit gray-scale images, the alphabet is $\mathcal{A} = \{0, 1 \cdots, 255\}$. Thus, a context $c \in \mathcal{A}^K$ is often assigned an index from the *index set* $\mathcal{J} = \{1, 2, \cdots, N\}$, where N is a positive integer greater or equal to 2. A simple way to define the one-to-one mapping

$$g : \mathcal{A}^K \to \mathcal{J} \tag{3.4}$$

is to concatenate the values of each component in $c = \{v_1, v_2, \cdots, v_K\}$ to form a integer and take this integer as the index for the context c.

Now we are ready to define a K-pixel template context model for image compression.

Definition 3.4 (K-pixel Template Context Model)**.** Let T_K be a K-pixel template. A context model given by Definition 3.1 which takes the K-pixel template contexts as the accompanying context sequence S is called a *K-pixel template context model*. K is called the *order* of the context model.

Definition 3.5 (K-pixel Template Image Source)**.** Let $X = \{X_k\}_{k=1}^n$ be an alphabet $\mathcal{A}$ image source, where n is the total pixel number of the image. Given a K-pixel template context model with a finite context set

$$\mathcal{C} = \{c_1, c_2, \cdots, c_{|\mathcal{C}|}\},\ c_j \in \mathcal{A}^K \text{ for all } j = 1, 2 \cdots, |\mathcal{C}|, \tag{3.5}$$

and an accompanying context sequence

$$S = \{S_k\}_{k=1}^n,\ S_k \in \mathcal{C}, \tag{3.6}$$

X is called a K-pixel template image source if

$$\Pr\{X_k = x_k | X_1^{k-1} = x_1^{k-1}\} = p(x_k | s_k) \tag{3.7}$$

for a given scanning order and for all $k \geq 1$, where the transition probability function $p : \mathcal{A}^K \times \mathcal{A} \to [0, 1]$ satisfies $\sum_{a_{k+1} \in \mathcal{A}} p(a_{k+1} | s_k) = 1$ for any $S_k = s_k \in \mathcal{C}$, and $S_k = f(x_{k-1}, S_{k-1})$.

Remark 3.4. In fact, Equation (3.7) describes a Markov type property of image sources, *i. e.*, for a given scanning order and a K-pixel template, the conditional probability of a sample X_k given all the past samples $X_1, X_2, \cdots, X_{k-1}$, is equal to the conditional probability of that sample X_k, given only the samples in the template T_{X_k}. Equation (3.7) characterizes local correlations between the current pixel and its neighboring pixels.

Remark 3.5. It is easy to verify that, if the image alphabet $\mathcal{A}$ and the order of the template K are both finite, a K-pixel template image source is a finite-state source.

In this thesis, we assume that images have the Markov type property given by Equation (3.7) and as a result, we assume images are emitted by K-pixel template image sources defined by Definition 3.5.

3.3 Problem Formulation

As we mentioned before, high order context models pose difficulties in practical image compression. For a K-pixel template image source drawn from an alphabet $\mathcal{A}$, the cardinality of the context set $\mathcal{C}$ is $|\mathcal{C}| = |\mathcal{A}|^K$, *i. e.*, the number of contexts increases exponentially in the order of the template. For example, if we choose $K = 12$ for an 8-bit gray-scale image with an alphabet $\mathcal{A} = \{0, 1, \cdots, 255\}$, the total number of possible contexts in the context set is 256^{12}. Practically, we can not afford to store and process such a huge number of context statistics. Besides, we need sufficient samples to obtain a reasonably steady statistical state for each context. Supposing that we have a training set consisting of 100 gray-scale images of size 512×512, we then have a total of $512 \times 512 \times 100 = 26,214,400$ pixel samples. For a 3-pixel template which has $256^3 = 16,777,216$ possible contexts in total, on the average we have only less than 2 samples per context. In this case, the estimation of the conditional probabilities is not accurate and the compression performance will be negatively affected. This is the so-called context dilution problem. In a word, the context storage problem and context dilution problem restrict the use of a large number of contexts directly. To improve the compression performance, we need to quantize a large number of original contexts into a relatively small number of quantized contexts.

Before we formulate our problem, we first define a context quantizer as follows.

Definition 3.6 (Context Quantizer)**.** Let X be an alphabet $\mathcal{A}$ source. A *context quantizer* Q of dimension K and (codebook) size N is defined as a mapping from a finite set $\mathcal{C}$ containing $M = |\mathcal{A}|^K$ *original contexts*, into a finite set $\tilde{\mathcal{C}}$ containing N *quantized contexts*

$$Q : \mathcal{C} \to \tilde{\mathcal{C}}, \tag{3.8}$$

where $\mathcal{C} = \{c_1, c_2 \cdots, c_{|\mathcal{A}|^K}\}$ is called the *original context set*, $\tilde{\mathcal{C}} = \{\tilde{c}_1, \tilde{c}_2 \ldots, \tilde{c}_N\}$ is called the *quantized context set* or *codebook*, and $c_i \in \mathcal{A}^K$ for each $i \in \mathcal{I} =$

$\{1, 2, \cdots, |\mathcal{A}|^K\}$. Associated with each quantized context is a *partition* of the K-dimensional Euclidean space $\mathcal{A}^K$ into N *regions* or *cells*. The mth cell is defined by

$$\mathcal{P}_m = \{c \in \mathcal{A}^K : Q(c) = \tilde{c}_m\} \tag{3.9}$$

where $m \in \mathcal{J} = \{1, 2, \ldots, N\}$ and

$$\bigcup_{i=1}^{N} \mathcal{P}_i = \mathcal{C} \text{ and } \mathcal{P}_i \bigcap \mathcal{P}_j = \emptyset, \text{ for } i \neq j \tag{3.10}$$

To design an optimal context quantizer in the sense of minimizing the image compression rate, we need to optimize over the following two variables:

1. The codebook size N $(1 \leq N \leq |\mathcal{A}|^K)$ of the quantizer.
2. The context mapping Q given N.

Let $r(X)$ denote the total code length in bits resulted from encoding the sequence X. Let $r(x)$ denote the compression rate in bits per symbol resulted from encoding X. The relationship between $r(X)$ and $r(x)$ is simply given by

$$r(x) = \frac{1}{n} r(X), \tag{3.11}$$

where n is the sequence length of X. We have the following problem formulation.

Problem 1 (Context Quantization Problem 1)**.** Let X be an alphabet $\mathcal{A}$ source with an original context set $\mathcal{C}$. Let C denote the alphabet $\mathcal{C}$ context random variable which is joint distributed with X. We wish to find the optimal codebook size N^* and the optimal mapping Q^* given N^*, such that $r(x|Q(C))$ is minimized, *i. e.*,

$$\min_{N, Q(C)} r(x|Q(C)). \tag{3.12}$$

By using context-based arithmetic coding algorithms, the quantity $r(X|Q(C))$ can be written as

$$r(X|Q(C)) = \tilde{r}(X|Q(C)) + \psi. \tag{3.13}$$

In Equation (3.13), the first part $\tilde{r}(X|Q(C))$ is given by

$$\tilde{r}(X|Q(C)) = -\log \prod_{i=1}^{n} p(x_i|Q(C)), \tag{3.14}$$

which denotes the ideal code length assigned by the context-based arithmetic coding algorithms with the transition probability function $p(\cdot|\cdot)$, conditioned on the context $Q(C)$.

The second part ψ of Equation (3.13) denotes the total modeling cost in bits by using the context-based, adaptive arithmetic coding algorithms. One method to derive ψ is as follows. Fix N contexts. The original sequence X is separated into N i.i.d. sub-sequences, each of which is encoded by one of the N arithmetic coders. Let n_i denote the length of the sub-sequence encoded by the ith arithmetic coder, where $\sum_{i=1}^{N} n_i = n$. The extra bits, besides the ideal code length, resulted by using the ith arithmetic coder is given by $\frac{1}{2}\log(n_i+1)$. Thus the total modeling cost over N arithmetic coders can be approximated by

$$\psi \triangleq \sum_{i=1}^{N} \frac{1}{2}\log(n_i+1) \tag{3.15}$$

$$\leq \frac{N}{2}\log\sum_{i=1}^{N}\frac{n_i+1}{N} \tag{3.16}$$

$$= \frac{N}{2}\log\left(\frac{n}{N}+1\right) \tag{3.17}$$

where (3.16) follows from the Jensen's inequality since $\log(t)$ is a concave function of t.

Plugging Equation (3.14) and (3.17) into Equation (3.13), we have

$$r(X|Q(C)) \approx -\log\prod_{i=1}^{n} p(x_i|Q(C)) + \frac{N}{2}\log\left(\frac{n}{N}+1\right). \tag{3.18}$$

Thus, the original Problem 1 can be approximated by the following problem.

Problem 2 (Context Quantization Problem 2)**.**

$$\min_{N,Q(C)} \frac{1}{n}\left[-\log\prod_{i=1}^{n} p(x_i|Q(C)) + \frac{N}{2}\log\left(\frac{n}{N}+1\right)\right]. \tag{3.19}$$

In this thesis, we are more interested in how the compression performance can be improved by finding the optimal context mapping. For this reason we further simplify the Problem 2 by fixing the quantization codebook size N to optimize over Q only. The jointly optimization of N and Q is still interesting and will be left for future works at this time. To make the problem more trackable, we approximate the quantity $-\frac{1}{n}\log\prod_{i=1}^{n} p(x_i|Q(C))$ by the conditional entropy $H(X|Q(C))$. The simplified problem is as follows.

Problem 3 (Context Quantization Problem 3)**.** Fix the quantization codebook size N. Find an optimal context mapping Q such that the conditional entropy $H(X|Q(C))$ is minimized, *i. e.*,

$$\min_{Q(C)} H(X|Q(C)). \tag{3.20}$$

It should be noted that we still can refine N by repeating (3.20) for a set of empirical chosen codebook sizes N during an off-line training process. Therefore, the Problem 3, which is a subset of Problem 2, still has its theoretical and practical importance. The problem formulation (3.20) is also considered in references [36] and [40], which use dynamic programming to solve the problem in binary cases.

In the following section, we will present a Lloyd-like context quantizer design algorithm to address the Problem 3. The algorithm does not assume any alphabets of sources.

3.4 A Lloyd-like Context Quantizer Design Algorithm

Let X be a K-pixel template image source with an alphabet $\mathcal{A}$. Let $\mathcal{C}$ and $\tilde{\mathcal{C}}$ denote the original context set and the quantized context set, respectively. Let $M = |\mathcal{C}|$ and $N = |\tilde{\mathcal{C}}|$ denote the size of $\mathcal{C}$ and $\tilde{\mathcal{C}}$, respectively. Let C ($C \in \mathcal{C}$) denote the context random variable which is jointly distributed with X according to the jointly distribution function $\Pr\{X = x, C = c\} = p(x, c)$. Let $Q : \mathcal{C} \to \tilde{\mathcal{C}}$ denote the context mapping and we have

$$Q(C) = \tilde{C}, \quad C \in \mathcal{C},\ \tilde{C} \in \tilde{\mathcal{C}}. \tag{3.21}$$

The theoretical limit to compress X given C is the conditional entropy $H(X|C)$. We wish to minimize $H(X|Q(C))$ for a fixed N. Let R_Q denote the difference between the two quantities $H(X|Q(C))$ and $H(X|C)$. We have

$$\begin{aligned} R_Q &= H(X|Q(C)) - H(X|C) && (3.22)\\ &= H(X|Q(C)) - H(X|C, Q(C)) && (3.23)\\ &= I(X; C|Q(C)) && (3.24)\\ &\geq 0 && (3.25) \end{aligned}$$

where (3.23) follows from Theorem 2.1, and we have equality in (3.25) if and only if X and C are conditionally independent given $Q(C)$. The quantity R_Q is the extra bits (on the average) we use to encode X conditioned on the quantized context $Q(C)$ instead of the original context C, *i. e.*, it is a measure of inefficiency of the context quantizer Q. Apparently, to minimize $H(X|Q(C))$, it is equivalent to minimize R_Q.

Let $\mathcal{P}_m$ $(m = 1, 2, \cdots, N)$ denote the mth cell given by Equation (3.9), and $\mathcal{P}_m$ satisfies the properties given by Equation (3.10). We have:

$$\begin{aligned} H(X|C) &= \sum_{c_j \in \mathcal{C}} p(c_j) H(X|C = c_j) \\ &= -\sum_{c_j \in \mathcal{C}} p(c_j) \sum_x p(x|c_j) \log p(x|c_j) \\ &= -\sum_{m=1}^{N} \sum_{c_j \in \mathcal{P}_m} p(c_j) \sum_x p(x|c_j) \log p(x|c_j) \end{aligned} \tag{3.26}$$

and

$$H(X|Q(C)) = H(X|\tilde{C}) \tag{3.27}$$

$$\begin{aligned} &= \sum_{\tilde{c}_m \in \tilde{C}} p(\tilde{c}_m) H(X|\tilde{C} = \tilde{c}_m) \\ &= -\sum_{\tilde{c}_m \in \tilde{C}} p(\tilde{c}_m) \sum_x p(x|\tilde{c}_m) \log p(x|\tilde{c}_m) \\ &= -\sum_{m=1}^{N} p(\tilde{c}_m) \sum_x p(x|\tilde{c}_m) \log p(x|\tilde{c}_m), \end{aligned} \tag{3.28}$$

where

$$p(\tilde{c}_m) = \sum_{c_j \in \tilde{\mathcal{C}}_m} p(c_j). \tag{3.29}$$

Since for all $i \neq j$, $c_i \neq c_j$, we have

$$p(x, \tilde{c}_m) = \sum_{c_j \in \mathcal{P}_m} p(x, c_j) \tag{3.30}$$

Thus, we have

$$\begin{aligned} p(x|\tilde{c}_m) &= \frac{p(x, \tilde{c}_m)}{p(\tilde{c}_m)} \\ &= \frac{\sum_{c_j \in \mathcal{P}_m} p(x, c_j)}{\sum_{c_j \in \mathcal{P}_m} p(c_j)} \\ &= \frac{\sum_{c_j \in \mathcal{P}_m} p(c_j) p(x|c_j)}{\sum_{c_j \in \mathcal{P}_m} p(c_j)}, \end{aligned} \tag{3.31}$$

i. e., the conditional probability $p(x|\tilde{c}_m)$ is the "centroid" of set $\{p(x|c_j):\ c_j \in \mathcal{P}_m\}$.

Plugging (3.26) and (3.28) into (3.22), we have

$$
\begin{aligned}
R_Q &= H(X|Q(C)) - H(X|C) && (3.32)\\
&= -\sum_{m=1}^{N} p(\tilde{c}_m) \sum_x p(x|\tilde{c}_m) \log p(x|\tilde{c}_m) \\
&\quad + \sum_{m=1}^{N} \sum_{c_j \in \mathcal{P}_m} p(c_j) \sum_x p(x|c_j) \log p(x|c_j) && (3.33)\\
&= -\sum_{m=1}^{N} p(\tilde{c}_m) \sum_x \frac{\sum_{c_j \in \mathcal{P}_m} p(c_j) p(x|c_j)}{\sum_{c_j \in \mathcal{P}_m} p(c_j)} \log p(x|\tilde{c}_m) \\
&\quad + \sum_{m=1}^{N} \sum_{c_j \in \mathcal{P}_m} p(c_j) \sum_x p(x|c_j) \log p(x|c_j) && (3.34)\\
&= -\sum_{m=1}^{N} \sum_x \sum_{c_j \in \mathcal{P}_m} p(c_j) p(x|c_j) \log p(x|\tilde{c}_m) \\
&\quad + \sum_{m=1}^{N} \sum_{c_j \in \mathcal{P}_m} p(c_j) \sum_x p(x|c_j) \log p(x|c_j) && (3.35)\\
&= -\sum_{m=1}^{N} \sum_{c_j \in \mathcal{P}_m} p(c_j) \sum_x p(x|c_j) \log p(x|\tilde{c}_m) \\
&\quad + \sum_{m=1}^{N} \sum_{c_j \in \mathcal{P}_m} p(c_j) \sum_x p(x|c_j) \log p(x|c_j) && (3.36)\\
&= \sum_{m=1}^{N} \sum_{c_j \in \mathcal{P}_m} p(c_j) \sum_x p(x|c_j) \log \frac{p(x|c_j)}{p(x|\tilde{c}_m)} && (3.37)
\end{aligned}
$$

Define the quantity $\delta(c_j, \tilde{c}_m)$ as

$$\delta(c_j, \tilde{c}_m) \triangleq \sum_x p(x|c_j) \log \frac{p(x|c_j)}{p(x|\tilde{c}_m)} \tag{3.38}$$

We see that $\delta(c_j, \tilde{c}_m)$ is the *relative entropy* or *Kullback Leibler distance* [28] between the two probability mass function $p(x|c_j)$ and $p(x|\tilde{c}_m)$, *i. e.*,

$$
\begin{aligned}
\delta(c_j, \tilde{c}_m) &= D(p(x|c_j) \parallel p(x|\tilde{c}_m)) && (3.39)\\
&= D(p(x|c_j) \parallel Q(p(x|c_j)) && (3.40)
\end{aligned}
$$

In view of (3.40), we regard $\delta(c_j, \tilde{c}_m)$ as the "distortion" between $p(x|c_j)$ and $p(x|\tilde{c}_m)$, which is similar as the distortion measure between the original variable

and its reproduction codeword in the traditional vector quantization problem [38]. Therefore, we can use Lloyd-like iterative algorithm to design a context quantizer such that R_Q is minimized. The Lloyd-like iterative algorithm is described as follows.

Algorithm 1 Lloyd-like context quantization algorithm

1. Set $i = 0$, $R_Q^{(0)} = R_{\max}$. Set the maximal iteration number $N_{\max}$ and a small threshold ϵ.

2. Form an initial partition $\mathcal{G}^{(i)}$ of the original context set $\mathcal{C}$ into N cells $\mathcal{P}_m^{(i)}$ $(m = 1, 2, \cdots, N)$.

3. Calculate the centroid $p(x|\tilde{c}_m)$ for each $\mathcal{P}_m^{(i)}$ to get an initial codebook $\tilde{\mathcal{C}}^{(0)}$, where $p(x|\tilde{c}_m)$ is given by (3.31).

4. Set $i + 1 \to i$, and update the partition. For each context $c_j \in \mathcal{C}$, calculate

$$m' = \arg\min_m \sum_x p(x|c_j) \log \frac{p(x|c_j)}{p(x|\tilde{c}_m)} \tag{3.41}$$

   and assign c_j to the cell $\mathcal{P}_{m'}$ to form the new partition $\mathcal{G}^{(i)}$

5. Calculate the new $R_Q^{(i)}$ given by (3.37). If $\frac{R_Q^{(i-1)} - R_Q^{(i)}}{R_Q^{(i-1)}} > \epsilon$ and $i < N_{\max}$, go to Step 3.

6. Stop. The final partition $\mathcal{G}^{(i)}$ contains an optimal context mapping Q^*.

It is easy to verify that $R_Q^{(i)}$ $(i = 1, 2...)$ are non-increasing in the above algorithm. Since $R_Q^{(i)} \geq 0$ for all i, the algorithm will converge to a stationary point and then stop after a finite number of iterations. Same as the generalized Lloyd algorithm for traditional vector quantization, there is no guarantee that Algorithm 1 finds the global optimal mapping Q in the sense that $H(X|Q(C))$ is minimized.

3.5 Implementation Issues

In this section, we will discuss some important algorithm implementation issues.

3.5.1 Initial Codebook Design

To perform the Lloyd-like algorithm, we have to begin with an initial codebook or partition. There are many ways to generate an initial codebook [38]. In this section, we will discuss two of them.

Random mapping: The simplest method to generate an initial context quantization codebook is to randomly map the original contexts into N cells numbered from 1 to N. This technique has often been used in pattern recognition. Although Random mapping is quite simple, it may form a bad initial partition in the sense that we can not get a good final mapping after we use the Lloyd-like algorithm. Nevertheless, we can always generate a set of initial codebooks by random mapping, apply the Lloyd-like algorithm, and select the final mapping which gives us the best performance.

Splitting: Linde *et al.* introduced a technique called the splitting method for the traditional vector quantization codebook design [38]. A similar method can be applied to design an initial codebook for context quantization. First, we select the centroid of the conditional probabilities of the whole training set as the resolution 0 codeword, say $p(x|\tilde{c}_0)$. Then we split this codeword into two codewords, say $p(x|\tilde{c}_0)$, and $p(x|\tilde{c}'_0)$ which is a small disturbance on $p(x|\tilde{c}_0)$. Now we have a two-codeword codebook and then apply the Lloyd-like algorithm to improve it to generate the final resolution 1 codebook. The reason we include the codeword generated at the last step into the new codebook design process is to guarantee that the new codebook will not be worse the old one. In this manner, we can always generate a resolution $r+1$ codebook from the resolution r codebook. Finally, we can design a codebook for context quantization from a training set. Another way to do the splitting is to employ a tree structure. The method is to make a disturbance on the initial codebook $p(x|\tilde{c}_0)$ to make two codewords, $p(x|\tilde{c}'_0)$ and $p(x|\tilde{c}'_0)$. Each time we increase the codebook size, we make a disturbance on every codeword in the old codebook to generate two new codewords from each of the old codewords. Although the above two methods are not optimal, they generally produce a good context quantization codebook when the training set is large enough.

3.5.2 Empty cell problem

The Lloyd-like iterative algorithm assumes that the partition is *nodegenerate* in the sense that each cell of the quantizer has nonzero probability of containing a context, i.e., $p(\tilde{c}_m) \neq 0$ for all $m = \{1, 2, \ldots, N\}$. For a cell which contains no

context, $p(\tilde{c}_m) = 0$, the centroid of the cell is undefined. This is called the *empty cell problem* for context quantization. Apparently, a quantizer with one or more empty cells will not be optimal because we can remove those empty cells and split other cells with nonzero probabilities into two cells, and keep the codebook size unchanged while reducing the conditional entropy given the quantized contexts. In practice, the empty cell problem must be considered in the context quantizer design.

In practice, the number of original contexts could be much larger than the number of quantized contexts. We declare a cell as an "empty" cell if it contains 3 or less original contexts. All the empty cells have to be removed and the same number of new substituted cells have to be added in. We can select some of the non-empty cells to further splitting. We propose two heuristic criterion to select the candidate cells for further splitting. One criterion is to select the cell with the largest number of contexts. Another criterion would be selecting the cell with the maximal contribution to the conditional entropy. The two criterion can both produce good initial codebooks.

Chapter 4

Image Decomposition

Our objective is to design an efficient and effective progressive lossless image compression scheme. In the last chapter, we use context quantization to address the context dilution problem in order to improve image compression performance. In this chapter, we will use image decomposition to obtain the progressive image transmission feature. Furthermore, our compression scheme elegantly combines image decomposition with context quantization.

4.1 Introduction

In general, image decomposition means separating images into components. In progressive image compression, we decompose an image into a series of progressively refined sequences and compress them dependently. Kieffer and Yang generalized progressive lossless compression schemes in the framework of "refinement source coding" [8]. To compress an image X, a progressive compression scheme generates a finite sequence Y from X

$$Y = \{y^1, y^2, \cdots, y^L\}, \tag{4.1}$$

where $L > 1$ denotes the total number of decomposition levels and $y^L = X$. The sequence Y is called the progressive data sequence of X. Each y^l $(1 \leq l < L)$ is obtained through processing of y^{l+1}, and regarded as a coarsened version of y^{l+1}. Equivalently, y^{i+1} is regarded as a refined version of y^l. In refinement source coding, the encoder compresses X by encoding y^1 dependently, and then encoding each y^l conditioned on y^{l-1} for $l = 2, 3 \cdots, L$, *i. e.*, the encoder will generate a series of

encoded sequence

$$\{v_l\}_{l=1}^{L} = \{v_1(y^1), v_2(y^2|y^1), v_3(y^3|y^2), \cdots, , v_L(y^L|y^{L-1})\}, \tag{4.2}$$

and send $\{v_l\}_{l=1}^{L}$ to the decoder sequentially. If each v_l $(l = 1, 3 \cdots, L)$ is constructed losslessly, X will be losslessly compressed. The decoder can fully recover the sequence X by decoding v_1 to get y^1, and then decoding v_l conditioned on y_{l-1} for $l = 2, 3 \cdots, L$ to get y^l, where $y^L = X$.

There are many ways to generate the progressive data sequence Y. In this thesis, we will use a generalized version of the color splitting method used in [21], which is essentially an entropy-constrained vector quantization [41] based decomposition method with a binary tree structure.

4.2 Image Decomposition using Color Splitting

The color splitting method splits a representative color into several colors based on some criterion such as minimizing the distortion. To reduce the complexity, a tree structure, especially a binary-tree structure, is usually preferred [42].

The binary tree structure yields binary tree color splitting methods. Initially, we represent the whole image with one representative color node, and map all image pixels into it. In the next step, we split this root color node into two child color nodes, each of which is assigned by a representative color, and map the pixels belonging to its father color node into the two child color nodes. When we have n leaf color nodes in the tree by splitting, we add one leaf color node by selecting one of the existing n leaf color nodes and split that color node into two. Thus the tree grows from n leaf nodes to $n + 1$ leaf nodes and the total number of colors represented by the tree is $n + 1$. The process continues until each color in the original image is represented by one leaf node.

4.2.1 Binary Tree Color Splitting

Assume that a gray-scale image has a color (gray-level) set $S = \{s_0, s_1, \ldots, s_{L-1}\}$, where L is the total number of distinct colors (gray levels). Let $f(s_i)$ $(i = 1, 2, \cdots, L)$ denote the occurrence number of the color s_i in the image. Let $f(S)$ denote the occurrence number of all the colors of the set S in the image, *i. e.*, $f(S) = \sum_{s_i \in S} f(s_i)$. Let T_0 denote the initial tree with only one root node. Let T_l $(l = 1, 2 \cdots, L - 1)$ denote the binary tree at the lth level splitting.

For each leaf node S_t of the binary tree, we define the representative color q_{S_t} as the centroid of all color occurrences in node S_t,

$$q_{S_t} = \frac{\sum_{s_j \in S_t} f(s_j) s_j}{f(S_t)}. \tag{4.3}$$

At level l, we use the representative colors associated with each leaf node of T_l to reconstruct the image I_{T_l}, the distortion associated with each leaf node S_t is thus,

$$d_{S_t} = \sum_{s_j \in S_t} f(s_j) d(s_j, q_{S_t}), \tag{4.4}$$

where $d(s_j, q_{S_t}) = (s_j - q_{S_t})^2$ for gray-scale images.

The total distortion associated with T_l is,

$$D_{T_l} = \sum_{t=0}^{L_{T_l}-1} d(S_t), \tag{4.5}$$

where L_{T_l} denotes the number of leaf node of T_l.

When we split a leaf node S into two child node S' and S'' at level l to build the level $l+1$ tree T_{l+1}, the distortion reduction is calculated as,

$$\begin{aligned} \Delta D_l &= D_l - D_{l+1} && (4.6) \\ &= \sum_{s_j \in S} f(s_j) d(s_j, q_S) - \sum_{s_j \in S'} f(s_j) d(s_j, q_{S'}) - \sum_{s_j \in S''} f(s_j) d(s_j, q_{S''}). && (4.7) \end{aligned}$$

On the other hand, we need more bits to indicate which child node, S' or S'' a pixel in S belongs to. The increased rate for the splitting is calculated as,

$$\begin{aligned} \Delta R_l &= R_{l+1} - R_l && (4.8) \\ &= -f(S') \log \frac{f(S')}{f(S)} - f(S'') \log \frac{f(S'')}{f(S)}, && (4.9) \end{aligned}$$

where R_l and R_{l+1} are the rate corresponding to the T_l and T_{l+1}.

Each time we split a leaf node, we will get a distortion D and a rate R. We define a Lagrangian cost function

$$J = D + \lambda R \tag{4.10}$$

as the criterion to determine the node to be split at each level: we choose the node which produces the minimum cost J to split at each level. Here, the parameter λ is empirically chosen. By fixing λ, we can use an iterative algorithm to minimize J to refine the pixel mapping when we split a leaf node into two child nodes.

4.2.2 Algorithm Description

The binary tree color splitting algorithm is described as follows.

Algorithm 2 Binary tree color splitting algorithm

1. Set $l = 0$. Initially, all the colors belong to the root node $S_0 = S$ of the binary tree T_0. Set $R_0 = 0$ and $D_0 = \sigma^2$, where σ^2 is the variance of S_0. Set the maximum number of iteration loops $N_{\max}$ and a small threshold ϵ.

2. Find the minimum achievable cost for splitting each leaf node of T_l. For each leaf node S_t of T_l, we tentatively split each leaf node as follows.

   Step 2.1: Set $J_{\text{old}} = D_l + \lambda R_l$.

   Step 2.2: Make an initial splitting. For each color $s_j \in S_t$, if $s_j \leq q_{S_t}$, assign s_j to S'_t. Otherwise, assign s_j to S''_t.

   Step 2.3: Calculate $q_{S'_t}$ and $q_{S''_t}$. For each color $s_j \in S_t$, if

   $$d(s_j, q_{S'_t}) - \lambda f(S'_t) \log \frac{f(S'_t)}{f(S_t)} \leq d(s_j, q_{S''_t}) - \lambda f(S''_t) \log \frac{f(S''_t)}{f(S_t)}, \tag{4.11}$$

   we assign s_j to S' and generate a binary value 0. Otherwise, we assign s_j to S'' and generate a binary value 1. We concatenate these binary values to generate a binary sequence γ_l.

   Step 2.4: Calculate ΔD_l and ΔR_l. We have

   $$J^{(j)}_{\text{new}} = (D_l - \Delta D_l) + \lambda(R_l + \Delta R_l) \tag{4.12}$$

   If $\frac{|J_{\text{old}} - J_{\text{new}}|}{J_{\text{old}}} \leq \epsilon$, or the number of iteration loops is greater than $N_{\max}$, we stop. Otherwise, we continue Step 2.3 and 2.4.

3. Find

   $$j = \arg\min_{s_j \in S_t} J^{(j)}_{\text{new}}. \tag{4.13}$$

   Split node j by the same process as Step 2.1 to 2.4.

Algorithm 2 is essentially a generalized Lloyd algorithm for entropy-constrained vector quantization [41]. The algorithm will converge to a stationary point after a finite number of iterations. By using Algorithm 2, we grow the binary tree with l leaf nodes to the binary tree with $l+1$ nodes. The process continues until we have L leaf nodes, where L denotes the total number of distinct colors of the image. Each

time we split a leaf node into two child nodes, we use a binary sequence indicating which child node a pixel belongs to.

4.3 Combining Image Decomposition with Context Quantization

Through the above image decomposition algorithm, we decompose the image into a series of objects . At level l, each object consists of three components with respect to the leaf node S to be split in the tree T_{l-1}: the representative color of the node S, the representative colors of its two child nodes S' and S'', and the binary sequence γ_l. For the three representative colors, we do not encode and send them directly to the decoder. For the binary sequences $\{\gamma_l\}_{l=1}^{L-1}$, we use context-based, adaptive arithmetic coding algorithm to encode them.

In order to efficiently encode the sequences $\{\gamma_l\}_{l=1}^{L-1}$, we need to perform context quantization. Note that, after decomposition the original gray-scale image is transformed into a series of binary sequence. Therefore, the complexity of context quantization and arithmetic coding is reduced. Moreover, the image decomposition not only provides the feature of progressive image transmission, but also offers an opportunity to use non-causal templates which help to further improve the compression performance.

Figure 4.1 shows a non-causal template. Usually, we can only use the past pixels labeled by "c" to form a causal template. However, with image decomposition, at level $l > 1$, we already have some partial information of the non-causal pixel values. Thus, we can use both the past pixels labeled by "c" and the "future" pixels labeled by "x" to form a non-causal template for level $l > 1$. This is an advantage that image decomposition brings out.

Suppose we decompose an image into a series of binary sequence $\{B_l\}_{l=1}^{L} = \{B_1, B_2, \cdots, B_L\}$ with L levels. At the level $l = 1$, we have only two colors, *i. e.*, B_1 represents a binary image. For B_1, we can only use a causal template because we do not have any information about the future pixels right now. At the levels $l > 1$, we can use a non-causal template because we not only know all the past pixels at level l, but also know all the future pixels up to level $l + 1$. Therefore, to encode sequences at level $l > 1$, we shall use a non-causal template instead of a causal template.

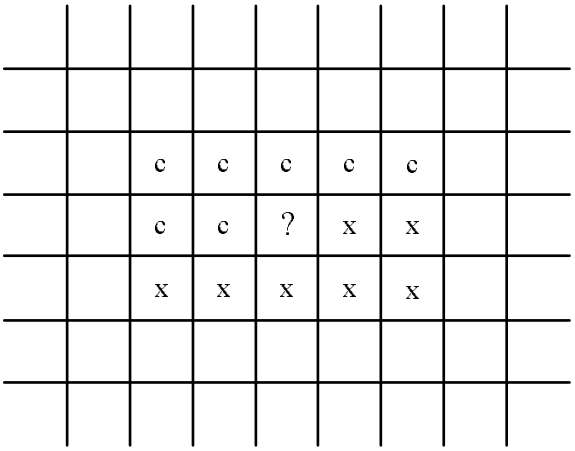

Figure 4.1: A noncausal Template

To get a good compression performance, we shall combine the image decomposition with the context quantization. We do the context quantization level by level in an off-line training process. For each level, we first choose the K_l-pixel template T_l and the codebook size N_l for level l. Note that T_1 is a causal template and all T_l $(l = 2, 3, \cdots, L-1)$ are non-causal templates. For each level l, we refine the context mapping

$$Q_l : \mathcal{A}_l^{K_l} \to \tilde{\mathcal{C}}_l \tag{4.14}$$

by using the context quantization algorithm (Algorithm 1) in Chapter 3, where A_l denotes the alphabet at level l and $|A_l| = l + 1$, and $\tilde{\mathcal{C}}_l$ denotes the quantized context set at level l.

Chapter 5

Experimental Results

To evaluate the compression performance of our proposed algorithm, we compare the proposed algorithm in this thesis with other representative lossless image compression schemes in the literature. The comparison consists of two parts: comparison of bi-level image compression and comparison of gray-scale image compression. For bi-level images, we compare the proposed algorithm with JBIG. For gray-scale images, we compare the proposed algorithm with CALIC. The main performance metric is compression rate in bits per pixel (bpp).

5.1 Image Sets

The images used for compression are classified into two major types: bi-level images and gray-scale images. Bi-level images are from two sources. The first source contains eight ccitt facsimile images and the second source contains ten binary images converted from scientific papers in PDF format downloaded from the Internet. Gray-scale images are also from two sources. The first source consists of 15 images from the miscellaneous categories of USC-SIPI Image Database [43], which include several "standard" images for image compression. Since the images provided in the database are 24-bit color images, we use Matlab to convert them into YCbCr color space and use Y-component images as the image set. The image sizes are 256×256 and 512×512. The second source consists of 20 images extracted from 4 HDTV sequences in YUV format. Again we take only Y-component images as our image set. The sizes of HDTV frames are 1920×1024.

We divide all the 60 images into two sets, the training set and the testing set. We select 20 images from our whole image set to form the training set. The rest

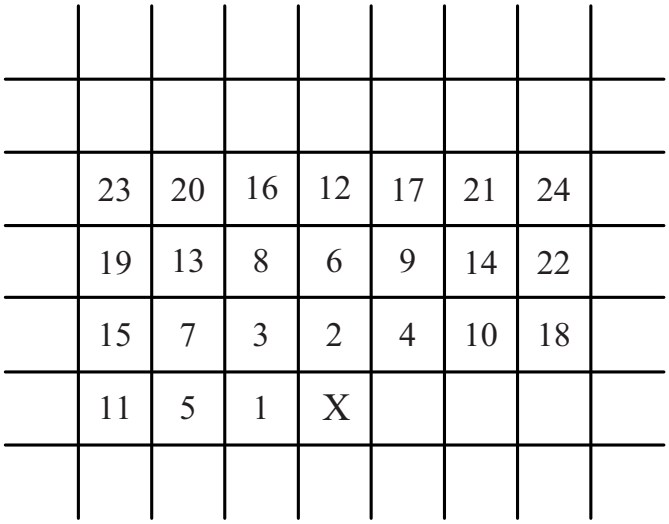

Figure 5.1: The causal template used in experiments (use only 1-15)

images are put into the testing set. The training images include both binary images and gray-scale images with different types. Then we use the image decomposition method in Chapter 4 and the context quantization algorithm in Chapter 3 to decompose those training images and design context quantizers for each level from them. Finally, we apply the obtained quantizers on the images in the testing set. The results we get in the following section are from the same quantizers.

In order to make a fair comparison with JBIG, we fix the codebook size for the binary images as $N_1 = 2^{10}$, *i. e.*, we use the same number of contexts in the arithmetic coders as JBIG. For gray-scale images, we perform the context quantization up to level $l = 180$. As samples available in the following levels are less and less, we do not get much gain from context quantization. The codebook sizes at each level are empirically chosen based on the training set. The maximal codebook size is 2^{12}.

5.2 Compression Results of Binary Images

To test our context quantization algorithm, we use eight CCITT fax image and five PDF pages as the training set. We choose a 15-pixel context template which has a total of 2^{15} original contexts which is shown in Figure 5.1. We set the quantization codebook size as $N_1 = 2^{10}$ which is the same as the JBIG standard.

Table 5.1 shows the compression rates in bits per pixel (bpp) for the eight

CCITT fax images.

Table 5.1: CCITT Images (bpp)

Image	Proposed	JBIG
ccitt1	0.029	0.030
ccitt2	0.016	0.018
ccitt3	0.042	0.044
ccitt4	0.107	0.107
ccitt5	0.050	0.051
ccitt6	0.024	0.025
ccitt7	0.103	0.111
ccitt8	0.027	0.029

From Table 5.1, we see that, by using a large context template and context quantization, we slightly improve the compression performance over the JBIG.

Table 5.2: Binary Images Converted from PDF files (bpp)

Image	Proposed	JBIG	Improvement
PDFpage6	0.0588	0.0610	3.6%
PDFpage7	0.094	0.1776	47.0%
PDFpage8	0.432	0.588	26.5%
PDFpage9	0.0967	0.107	9.6%
PDFpage10	0.1163	0.1051	−9.6%
average	0.1596	0.2077	23%

Table 5.2 shows the compression rates for the 5 binary images converted from PDF pages of scientific papers. From Table 5.2, we see that 4 images in the testing set have smaller compression rates than JBIG, while only one of them is worse than JBIG.

Table 5.3 shows the compression rates for the binary images converted from "standard" images. We see that for those images, our proposed method is still better than JBIG by an average of 6.5%.

Table 5.3: Binary Images Converted from Natural Images (bpp)

Image	Proposed	JBIG	Improvement
Lena	0.148	0.157	5.7%
Elaine	0.187	0.205	8.7%
airport	0.288	0.302	4.6%
peppers	0.108	0.118	8.4%
average	0.1828	0.1955	6.5%

	45	38	30	26	31	39	46	
	37	21	14	10	15	22	40	
	29	13	5	2	6	16	32	
	25	9	1	X	3	11	27	
	36	20	8	4	7	17	33	
	44	24	19	12	18	23	41	
	48	43	35	28	34	42	47	

Figure 5.2: The non-causal template used in experiments (use only 1-15)

5.3 Compression Results of Gray-Scale Images

In this section, we present the gray-scale image compression results obtained by our proposed scheme and CALIC. Table 5.4 shows the compression rates for the HDTV frames extracted from four HDTV sequences. For gray-scale images, we use a 15-pixel template shown in 5.1 to encode the first level binary image. For the rest of levels, since we already have all the surrounding pixels available, we use non-causal context templates shown in Figure 5.2. The maximal pixel number in the non-causal templates are 15.

Table 5.5 shows the compression performance for part of the images taken from USC image database [43]. The image titles are the same as the original USC images. We observe that the proposed method has competitive compression performance as CALIC. CALIC is very efficient for those images with human faces and images with more continuous changes. The reason is that the CALIC's GAP prediction

Table 5.4: HDTV frames (bpp)

Image	Proposed	CALIC
bankrun Frame1	5.26	5.31
bankrun Frame125	5.13	5.18
boat Frame100	4.48	4.46
boat Frame175	4.17	4.24
boat Frame225	4.14	4.15
city Frame1	4.29	4.25
city Frame125	4.32	4.30
manfraseco Frame200	4.18	4.10
manfraseco Frame250	3.91	3.89
average	4.44	4.43

and context modeling are specially designed for those images. On the average, the proposed method has a better performance for a more various types of images than CALIC. It should be noted that the proposed method has the progressive feature which is not included in CALIC.

Table 5.5: Gray-scale images from USC database (bpp)

Image	Proposed	CALIC
m_7.1.03	3.98	4.72
m_7.1.08	3.60	4.37
m_elaine	4.98	4.81
m_4.1.01	1.14	1.00
m_4.2.08	4.36	4.20
lena	4.39	4.12
m_5.2.10	3.68	5.37
average	3.73	4.08

For the purpose of completeness, we will show that the partial "future" information resulted from image decomposition is helpful to improve the compression performance. Recall that to compress gray-scale images, our scheme uses a causal template at the first level and uses non-causal templates at the following levels. We call this scheme as the "non-causal template" version. For our purpose, we modify the "non-causal template" version to use only causal templates, yielding a

"causal template" version. For the causal template version, we redesign the context quantizers with the causal templates by our proposed Lloyd-like algorithm. The results are shown in Table 5.6. We see that using non-causal templates is superior

Table 5.6: Gray-scale images from USC database (bpp)

Image	Non-causal template	Causal template
m_7.1.03	3.98	4.12
m_7.1.08	3.60	3.77
m_elaine	4.98	5.11
m_4.1.01	1.14	1.16
m_4.2.08	4.36	4.51
lena	4.39	4.50
m_5.2.10	3.68	3.86
average	3.73	3.86

to using causal templates in our scheme.

In short, our proposed scheme achieves competitive lossless compression performance for both bi-level images and gray-scale images compared to JBIG and CALIC, respectively. Moreover, we offer progressive image transmission feature for gray-scale image compression which is not provided by CALIC.

Chapter 6

Conclusions and Future Research

6.1 Conclusions

In this thesis, an efficient lossless image compression algorithm for both the binary images and gray-scale images is developed. Lossless image compression has extensive application in medical imaging, space photographing and film industry to archive and transmit images. To efficiently compress images, we first decompose images into a set of binary images to reduce encoding symbols. The benefits lie in four aspects. First, the progressive image transmission is achieved by image decomposition. Second, the encoding alphabet is reduced to the binary alphabet which is suitable for context quantization and adaptive arithmetic coding. Third, decomposition provides an opportunity to use those partial "future" information of non-causal pixels to help encoding. Finally, the decomposition provides a straightforward way to encode bi-level images, considering that current gray-scale image compression algorithms usually have bad performance on bi-level images. To deal with the well-know context dilution problem, we propose a Lloyd-like context quantization algorithm which refines the context mapping to minimize the compression rate. By combining image decomposition and context quantization, we design an efficient lossless image compression scheme for both bi-level images and gray-scale images. Experimental results show that our scheme has competitive compression performance on both bi-level images and gray-scale images compared to JBIG and CALIC, respectively. Our scheme provides an interesting progressive transmission feature for gray-scale image compression as well.

6.2 Future Research

Currently, our method uses an off-line training process to obtain the source statistics. Although the compression performance on images outside the training set is attractive, we have to maintain a context mapping table along with the coding program. In the future work we want to design some heuristic context quantization methods based our current learning results. We expect that the method can adaptively do the context quantization without maintaining a preset context mapping table. The second work is to reduce the computing complexity. The process to decompose images into multi-levels and then encode the image level by level makes the complexity of the proposed scheme higher than JBIG and CALIC. There is still much space to improve the code efficiency and reduce the computing and storage complexity. Finally, extending the work to the video compression is also very interesting.

List of References

[1] C. Poynton. (1997) Frequently asked questions about color. [Online]. Available: http://www.poynton.com/PDFs/ColorFAQ.pdf

[2] A. Ford and A. Roberts. (1998, Aug.) Colour space conversions. [Online]. Available: http://www.poynton.com/PDFs/coloureq.pdf

[3] K. R. Sloan, Jr. and S. L. Tanimoto, "Progressive refinement of raster images," *IEEE Trans. Comput.*, vol. C-28, no. 11, pp. 871–874, Nov. 1979.

[4] K. Knowlton, "Progressive transmission of grey-scale and binary pictures by simple, efficient, and lossless encoding schemes," *Proc. IEEE*, vol. 68, no. 7, pp. 885–896, Jul. 1980.

[5] K.-H. Tzou, "Progressive image transmission: a review and comparison of techniques," *Optical Engineering*, vol. 26, no. 7, pp. 581–589, Jul. 1987.

[6] P. H. Westerink, J. Biemond, and D. E. Boekee, "Progressive transmission of images using subband coding," in *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP'89)*, vol. 3, Glasgow, UK, May 1989, pp. 1811–1814.

[7] A. Said and W. A. Pearlman, "An image multiresolution representation for lossless and lossy compression," *IEEE Trans. Image Process.*, vol. 5, no. 9, pp. 1303–1310, Sep. 1996.

[8] J. C. Kieffer and E.-H. Yang, "Grammar-based lossless universal refinement source coding," *IEEE Trans. Inf. Theory*, vol. 50, no. 7, pp. 1415–1424, Jul. 2004.

[9] *Coded Representation of Picture and Audio Information—Progressive Bi-level Image Compression*, ITU-T Recommendation T.82 and ISO/IEC International Standard 11 544, 1993.

[10] *Information Technology—Coded Representation of Picture and Audio Information—Lossy/Lossless Coding of Bi-Level Images*, ITU-T Recommendation T.88 and ISO/IEC International Standard 14 492, 2000/2001.

[11] P. G. Howard, F. Kossentini, B. Martins, S. Forchhammer, and W. J. Rucklidge, "The emerging JBIG2 standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 7, pp. 838–848, Nov. 1998.

[12] *Information Technology—Digital Compression and Coding of Continuous-Tone Still Images—Requirements and Guidelines*, ITU-T Recommendation T.81 and ISO/IEC International Standard 10 918-1, 1992/1993.

[13] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. xviii–xxxiv, Feb. 1992.

[14] *Information technology—JPEG 2000 image coding system: Core coding system*, ITU-T Recommendation T.800 and ISO/IEC International Standard 15 444-1, 2000.

[15] *Information technology—JPEG 2000 image coding system: Extensions*, ITU-T Recommendation T.801 and ISO/IEC International Standard 15 444-2, 2000.

[16] D. S. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*, 1st ed. Norwell, MA, USA: Kluwer Academic Publishers, 2002.

[17] X. Wu and N. Memon, "Context-based, adaptive, lossless image coding," *IEEE Trans. Commun.*, vol. 45, no. 4, pp. 437–444, Apr. 1997.

[18] P. G. Howard and J. S. Vitter, "Fast and efficient lossless image compression," in *Proc. IEEE Data Compression Conference (DCC'93)*, Snowbird, UT, Mar./Apr. 1993, pp. 351–360.

[19] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The loco-i lossless image compression algorithm: Principles and standardization into jpeg-ls," Hewlett-Packard Laboratories and Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, HP Laboratories Tech. Report HPL-98-193R1, Oct. 1999. [Online]. Available: http://www.hpl.hp.com/loco/HPL-98-193R1.pdf

[20] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, Jun. 1996.

[21] X. Chen, S. Kwong, and J.-F. Feng, "A new compression scheme for color-quantized images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 10, pp. 755–777, Oct. 2002.

[22] İ. Avcibaş, N. Memon, B. Sankur, and K. Sayood, "A progressive lossless/near-lossless image compression algorithm," *IEEE Signal Process. Lett.*, vol. 9, no. 10, pp. 312–314, Oct. 2002.

[23] A. J. Pinho and A. J. R. Neves, "A context adaptation model for the compression of images with a reduced number of colors," in *Proc. IEEE International Conference on Image Processing (ICIP'05)*, vol. 2, Genova, Italy, Sep. 2005, pp. 738–741.

[24] T. Acharya and P.-S. Tsai, *JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures*, 1st ed. Hoboken, NJ, USA: John Wiley & Sons, Inc., Oct. 2004.

[25] K. Sayood, *Introduction to Data Compression*, 3rd ed., ser. The Morgan Kaufmann Series in Multimedia Information and Systems. San Francisco, CA: Morgan Kaufmann, Dec. 2005.

[26] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[27] G. R. Grimmett and D. R. Stirzaker, *Probability and Random Processes*, 3rd ed. New York: Oxford University Press, Jul. 2001.

[28] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. John Wiley & Sons, Inc., Jun. 2006.

[29] P. E. Tischer, R. T. Worley, A. J. Maeder, and M. Goodwin, "Context-based lossless image compression," *The Computer Journal*, vol. 36, no. 1, pp. 68–77, Jan. 1993.

[30] J. Rissanen, "Universal coding, information, prediction, and estimation," *IEEE Trans. Inf. Theory*, vol. 30, no. 4, pp. 629–636, Jul. 1984.

[31] *Information Technology—Lossless and Near-Lossless Compression of Continuous-Tone Still Images: Baseline*, ITU-T Recommendation T.87 and ISO/IEC International Standard 14 495-1, 1999.

[32] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, Jun. 1987.

[33] A. Moffat, R. M. Neal, and I. H. Witten, "Arithmetic coding revisited," *ACM Transactions on Information Systems (TOIS)*, vol. 16, no. 3, pp. 256–294, Jul. 1998.

[34] D. A. Huffman, "A method for the construction of minimum-redundancy codes," in *Proc. of the IRE*, vol. 40, no. 9, Sep. 1952, pp. 1098–1101.

[35] M. J. Weinberger, J. J. Rissanen, and R. B. Arps, "Applications of universal context modeling to lossless compression of gray-scale images," *IEEE Trans. Image Process.*, vol. 5, no. 4, pp. 575–586, Apr. 1996.

[36] X. Wu, P. A. Chou, and X. Xue, "Minimum conditional entropy context quantization," in *Proc. IEEE International Symposium on Information Theory (ISIT'00)*, Sorrento, Italy, Jun. 2000, p. 43.

[37] R. M. Gray, "Vector quantization," *IEEE ASSP Mag., [see also IEEE Signal Process. Mag.]*, vol. 1, no. 2, pp. 4–29, Apr. 1984.

[38] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, 1st ed. Norwell, MA, USA: Kluwer Academic Publishers, Nov. 1991.

[39] D.-K. He, "Grammar-based codes: from context-free to context-dependent," Ph.D. dissertation, University of Waterloo, Department of Electrical and Computer Engineering, Waterloo, Ontario, Canada, Jun. 2003.

[40] S. Forchhammer, X. Wu, and J. D. Andersen, "Optimal context quantization in lossless compression of image data sequences," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 509–517, Apr. 2004.

[41] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Entropy-constrained vector quantization," *IEEE Trans. Acoust., Speech, Signal Process. [see also IEEE Trans. Signal Process.]*, vol. 37, no. 1, pp. 31–42, Jan. 1989.

[42] P. C. Cosman, K. L. Oehler, E. A. Riskin, and R. M. Gray, "Using vector quantization for image processing," *Proc. IEEE*, vol. 81, no. 9, pp. 1326–1341, Sep. 1993.

[43] The USC-SIPI Image Database. University of Southern California. [Online]. Available: http://sipi.usc.edu/database/

Index