Lossless audio compression of speech and voice

Yang Liu

University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation
Liu, Yang, "Lossless audio compression of speech and voice" (2002). UNLV Retrospective Theses & Dissertations. 1457.
https://digitalscholarship.unlv.edu/rtds/1457

This Thesis is brought to you for free and open access by Digital Scholarship@UNLV. It has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.
INFORMATION TO USERS

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

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

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

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®
LOSSLESS AUDIO COMPRESSION OF SPEECH AND VOICE

by

Yang Liu

Bachelor of Science
Shanghai University of Science and Technology, Shanghai, China
1987

Doctor of Philosophy
University of Nevada at Las Vegas, U.S.A
2000

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science Degree
Department of Computer Science
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
December 2002
The Thesis prepared by

Yang Liu

Entitled

Lossless Audio Compression of Speech and Voice

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative
ABSTRACT

Lossless Audio Compression of Speech And Voice

by

Yang Liu

Dr. Evangelos A. Yfantis, Examination Committee Chair
Professor of Computer Sciences
University of Nevada, Las Vegas

Data compression plays an important role in modern telecommunication, entertainment, computing and networking areas because the capacities of media for storage and bandwidth for transmission are not growing proportionally with the rapid demand for multimedia data. Speech and voice compression is one of applications in this field. Although there are a lot of techniques used in speech coding, new algorithms need to be developed to achieve better performance. Our research focus is on lossless speech and voice compression using wavelet transform, prediction, and Rice coding. These techniques have some properties, such as, very fast computation and easily exploiting the redundancy in the speech signal. By taking these advantages, we can reduce the number of bits required to represent the audio signal and get better lossless audio compression. In this thesis, basic concepts and properties of these techniques will be displayed and a new loseless algorithm based on these methods is presented as well. The test results of the new algorithm are shown and some analyses are given.
### TABLE OF CONTENTS

**ABSTRACT** .................................................................................................................... iii  
**LIST OF FIGURES** ......................................................................................................... vi  
**LIST OF TABLES** ........................................................................................................... vii  
**ACKNOWLEDGEMENTS** ........................................................................................... viii  
**CHAPTER 1 INTRODUCTION** ................................................................................ 1  
**CHAPTER 2 AUDIO COMPRESSION OPERATION 1: WAVELET TRANSFORM** .................. 5  
  2.1 Background .......................................................... 5  
  2.2 Wavelet Analysis .................................................. 7  
    2.2.1 Concepts and principles ................................. 7  
    2.2.2 Multiresolution Analysis ............................... 10  
    2.2.3 Fast and Efficient Algorithm ....................... 12  
    2.2.4 The Filter Representation ............................ 18  
  2.3 An Example: Harr Wavelet Transform .................. 22  
    2.3.1 definition: Harr Wavelet ............................ 22  
    2.3.2 The ordered fast Harr wavelet Transform ........ 23  
**CHAPTER 3 AUDIO COMPRESSION OPERATION 2: PREDICTION** ....................... 25  
  3.1 Digital Audio ....................................................... 25  
  3.2 Prediction ............................................................ 26  
    3.2.1 Differential Pulse Code Modulation (DPCM) .... 26  
    3.2.2 Adaptive Differential Pulse Code Modulation (ADPCM) ........ 26  
**CHAPTER 4 AUDIO COMPRESSION OPERATION 3: CODING** .............................. 29  
  4.1 Terminology ......................................................... 29  
  4.2 Fundamental Limits ............................................ 30  
  4.3 Some lossless Coding techniques ....................... 31  
**CHAPTER 5 A NEW LOSSLESS COMPRESSION ALGORITHM** ......................... 35  
  5.1 Scheme .............................................................. 35  
    5.1.1 Stage 1: Wavelet Transform ....................... 38
LIST OF FIGURES

Fig. 2.1 Diagram for decomposition and reconstruction algorithms  . . . . 15
Fig. 2.2 Diagram of filter banks ...................................................... 19
Fig. 2.3 Representation diagram of decomposition algorithm ............. 21
Fig. 2.4 Harr wavelet mother function ........................................... 22
Fig. 5.1 The systematic diagram of our compression system .......... 36
Fig. 5.2 Wavelet Decomposition .................................................... 39
LIST OF TABLES

Table 5.1 Some information of ten digital speech test files ................. 37
Table 5.2 The average errors of ten decomposed levels using predictor 1 41
Table 5.3 The average errors of ten decomposed levels using predictor 2 41
Table 5.4 The results of \(a, b, c\) of predictor 1 for ten decomposed level . . 42
Table 5.5 The average entropies for ten test files ............................... 45
Table 5.6 The compression ratio for ten test files ............................... 46
ACKNOWLEDGEMENTS

I would like to express my special thanks to Dr. Yfantis for his valuable advice, continuing encouragement and warmhearted guidance during the course of my graduate program in UNLV. I would also like to thank the professors on my committee, Dr. Bein, Dr. Gewali and Dr. Wang, for their time and critical suggestions on this work. Thanks also go to Ms. Willis and all staff of Department of Computer Science of UNLV for their enormous assistance.

It is a great pleasure to thank the consultants of the UNLV writing center, they gave me a lot of help for correcting writing errors in this thesis.

Finally, Thanks must go to my family and friends for their love and support over years.
CHAPTER 1

INTRODUCTION

Today people have been experiencing with information explosion since the world goes into a multimedia and Internet era. The need to store and transmit large masses of digitized multimedia information, such as data, voice, still image, audio, video, etc, is growing very rapidly in modern telecommunications, entertainment, computing and networking areas. Although computers and network become faster and data storage becomes less expensive and more efficient, it still can not be satisfying with the demand for multimedia data. One of wonderful solutions for this problem is to use data compression techniques. These techniques are so powerful and important that the data compression field has been an important part of computer science. Many researchers have dedicated to this area in both theories and applications and a lot of terrific results are achieved.

Data compression is the process of converting an input data stream (or the original raw data) into another data stream (the output) that has a smaller size. A stream at here is referred to either a file or a buffer in memory. Data Compression is possible only because a stream tends to have some information redundancies. These redundancies mostly come from data representation or from unimportant parts corresponding to human's visual or hearing sense. From this point of view, data compression can be considered as lossy or lossless. Lossless compression methods provide for exact recovery of the original data from its compressed version. Sometimes it is necessary because, for example, a file containing computer program may become worthless if
even one bit gets modified. Lossy compression means that when the compressed stream is decompressed, the result is not identical to the original data stream. But this method achieves better compression by losing some unimportant information. Data compression is combination of many principles. As far as mathematic and techniques are concerned, data compression can be classified as statistical methods (e.g., Huffman Coding, Arithmetic Coding, Lempel-Ziv Compression, Run-Length Encoding and Rice Coding), Dictionary techniques, Linear prediction (e.g. DPCM), Subband coding, Transform coding (e.g. Fourier transform and wavelet transform), as well as various forms of sampling and vector quantization.

For designing or making a compression system, by which is meant a program that is used to minimize the cost of storing messages containing a specified type of data in some specified storage format, three fundamental procedures are carried out [Rissanen and Langdon, 1981]. The first of these procedures is modeling: the process of learning, or making assumptions about, the structure of the data being compressed. The second important procedure is probability estimation, or statistics gathering: the process of assigning a probability to each of the possible “next” symbols in the input stream that is being compressed, given a particular model of the data. The third of the three principal procedures is that of coding. Given a probability distribution for the symbols in a defined source alphabet, and a symbol drawn from that alphabet, the coder communicates to the waiting decoder the identifier corresponding to that symbol. The coder is required to make use of a specified channel alphabet (normally, but not always, the binary values zero and one), and to make as efficient use as possible of the channel capacity subject to whatever other constraint are enforced by the particular application. A point to be noted in connection with the three procedures is that in some circumstances the probability estimation component will
sit more naturally with the modeler, and in others will be naturally combined with the coder. Different combinations of model and coder will result in different placements of the statistics module, with the exact placement usually driven by implementation concerns. Nevertheless, in a logical sense, the three components exist in some form or another in all compression systems. In the next several chapters, the details about these operations will be displayed.

In chapter two, one compression modeling, the wavelet transform which are used in our research, are presented. One of its important properties, multiresolution analysis is shown. Two powerful algorithms, decomposition and reconstruction, are discussed and the corresponding filter representation is explained as well. In addition, a typical wavelet form, Harr wavelet is introduced.

In many situations, physical adjacency of samples also means that their values are similar. For example, in an audio wave, values of adjacent samples do not usually vary by much. In other words, there exists redundancy in these data. How to exploit and remove the redundancy in a speech signal is the topic of chapter three. prediction is a useful tool to do this task. The principles of Differential Pulse Code Modulation (DPCM) and Adaptive DPCM are briefly discussed in this chapter.

In chapter four, first of all, the fundamental concepts and properties of coding theory are shown. Then, due to our interest focuses on the lossless compression, several lossless compression coding methods are introduced. Especially, some details about Rice coding are displayed.

One of important applications of data compression is the speech or audio coding. Speech or audio, as the most natural form of communication between humans, is one part of people's life. However, compared with the huge quantities of speech, the capacities of media for storage and bandwidth for transmission is limited. Such
examples are voice mail systems and cordless telephone channel or a mobile radio channel. Speech coding is concerned with the development of techniques that exploit the redundancy in the speech signal, in order to reduce the number of bits required to represent the signal. There are a lot of techniques used in speech coding. Every one has its advantage and disadvantage. New algorithms need to be developed to achieve better performance of compression. In this work, our research focuses on lossless audio compression. A new lossless compression algorithm is shown in chapter five. It includes three main operations, wavelet transform, prediction and Rice coding. Each of these operations is discussed in detail in the chapter. The test results of this new algorithm are presented and some analyses are given.
CHAPTER 2

AUDIO COMPRESSION OPERATION 1: WAVELET TRANSFORM

2.1 Background

Among many data compression methods, transform coding is one of the most powerful techniques. Sometimes the data have to be transformed from one form to another so that the easier operation makes sense in a particular context. Also in the transformed form, certain dependencies or characteristics in data can be seen better than without transformation. Being proper transformed, the transformed data will be uncorrelated, whereby most or, ideally, all of the signal energy is concentrated in a small portion of the output sequence and only these sequence of elements need be transmitted or be stored, so more quicker and efficient data transmission and store usage are achieved.

Fourier and Wavelet transform are two important transform methods. Roughly speaking, wavelet transform is a refinement of Fourier Transform. So before describing wavelet transform, let us begin with a brief review on the Fourier transform. The definition of the continuous Fourier transform of a function \( f(t) \in L^2(R) \) (where \( L^2(R) \) is a set of all square integrable real functions, that is, functions for which \( \int f^2(t)dt < \infty \)) is given as

\[
F(\omega) = \int_{-\infty}^{\infty} f(t)e^{i\omega t}dt
\]  \hspace{1cm} (2.1)

It transforms a signal or data set from one domain (e.g. time) to another domain (e.g. frequency) in which many characteristics of the signal or data set are revealed.
The Fourier transform is useful for signals with statistical properties that are constant over time or space. Because the Fourier transform is a function independent of time, it does not reflect frequencies that vary in time so it is less suitable as the representation of nonstationary signals, such as a spike, whose statistical properties change with time. Another shortcoming of standard Fourier methods is that, to extract frequency information at even a single $\omega$, it requires the computation of an integral over an infinite interval of time and makes impractical for many applications. Short-time Fourier transform (STFT), introduced by Gabor, addresses the problem of locality of functions by introducing a window with which the analyzed function is scaled. Gabor proposed a windowed Fourier transform as:

$$F_g(\omega, \tau) = \int_{-\infty}^{\infty} f(t)g(t - \tau)e^{j\omega t}dt \tag{2.2}$$

where $\tau$ is a shift (translation) parameter and $g \in L^2(R)$ represents a well-localized in time window function. When the window function is a Gaussian

$$g_\alpha(t) = \frac{1}{\sqrt{2\alpha}e^{-\tau^2/(4\alpha)}} \tag{2.3}$$

where $\alpha > 0$, STFT transform becomes the Gabor transform. In STFT, $g_\alpha(t - \tau)$ acts as time-localization and $e^{-j\omega t}$ as frequency-localization. However, the Heisenberg uncertainty principle limits the STFT time and frequency resolutions. This principle states that for any suitably chosen window function $g$ and its Fourier transform $G$ (also a suitable window), the product of the two window widths in the time-frequency plane satisfies

$$(2\Delta_g)(2\Delta_G) \geq 2 \tag{2.4}$$

with equality only for the Gabor transform. The inequality means that time and frequency cannot be measured at any precision level at the same time, and in particular, very high frequencies cannot be localized to very small time windows. Due

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
to the size of the window is fixed and cannot be adapted to the changing characteristics of the signal in STFT, the rendering of low frequencies (e.g. wavelengths longer than the window width) is inadequate; on the other hand, the long window gives very poor localization of high frequencies. But the solution is offered by wavelet analysis. Wavelet analysis is a scale-independent means of analyzing signals. It uses short windows at high frequencies and long windows at low frequencies. The notion of scale is introduced as an alternative to frequency, which leads to the time-scale representation.

2.2 Wavelet Analysis

2.2.1 Concepts and principles

Now we focus on wavelet transform and look at some basic concepts of wavelet and its specific properties. Traditionally defined, a wavelet is a wavy function \( \psi \) carefully constructed so as to have certain mathematical properties. The prototype function \( \psi \) called "mother wavelet" (or basic wavelet) is used as a window or a scale. At the same time, an entire set of wavelet functions \( \psi_{j,k} \) derived from \( \psi \) is also used as the same as \( \psi \) does. These basis functions, called wavelets, are constructed by means of two simple operations, translation (shifting) and dilation (scaling, e.g. stretching or shrinking), on the mother wavelet \( \psi \). This set of wavelet functions \( \psi_{j,k} \) provides useful "building block" functions which can be used to represent general functions and data sets. In other words, any time-varying signals (functions) can be represented in terms of these simple basis wavelet functions. Compared with Fourier transform, wavelet transform has several advantages. Simply stated[Ogden.1997], they are 1) good time-frequency localization, 2) simplicity of form, 3) fast and efficient algorithms.

As mentioned above, for a suitably defined mother wavelet \( \psi \), each wavelet derived
from the mother wavelet is generated by two operations, translating and dilating, on the mother wavelet. If allowing these two operations to vary continuously, the wavelet derived from a mother wavelet is mathematically expressed as:

$$\psi_{a,\tau}(t) = \frac{1}{\sqrt{\alpha}} \psi\left(\frac{t - \tau}{\alpha}\right)$$  \hspace{1cm} (2.5)

where $\alpha > 0$ is a scale (dilation) parameter and $\tau \in R$ is a shift (translation) parameter. For low-frequency wavelets, $\alpha > 1$, while for high-frequency wavelets, $\alpha < 1$. The normalization factor $\alpha^{-1/2}$ is included so that $\| \psi_{a,\tau} \| = \| \psi \|$. $\| f \|$ is called the norm of a function and is defined as

$$\| f \| = \int_{-\infty}^{\infty} |f(x)|^2 dx$$  \hspace{1cm} (2.6)

If restricting scaling and shifting of the mother wavelet by the set of integers and taking scaling $\alpha$ to be of the form $\alpha = 2^{-j}$ and dyadic translations $\tau$ to be of the form $k2^{-j}$, the wavelet becomes:

$$\psi_{j,k}(t) = 2^{j/2} \psi(2^j t - k)$$  \hspace{1cm} (2.7)

where $j$ and $k$ are integers and denoted as dilation index and translation index, respectively.

Given a function $f(t) \in L^2(R)$ with respect to some analyzing wavelet $\psi$, the continuous wavelet transform (CWT; integral wavelet transform) of the function $f$ is defined by

$$CWT(\alpha, \tau) = <f, \psi_{a,\tau}> = \int_{-\infty}^{\infty} f(t) \psi_{a,\tau}(t) dt$$  \hspace{1cm} (2.8)

$CWT(\alpha, \tau)$ represents the detail contained in the signal $f$ at the scale $\alpha$.

To reconstruct the original function from its continuous wavelet transform, the inverse continuous wavelet transform is done as follows:

$$f(t) = \frac{1}{C_\psi} \int_{0}^{\infty} d\alpha \int_{-\infty}^{\infty} CWT(\alpha, \tau) \psi_{a,\tau}(t) \frac{d\alpha}{\alpha^2}$$  \hspace{1cm} (2.9)
where $C_\psi$ is a constant and meets the admissibility condition

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega < \infty \quad (2.10)$$

This condition restricts the class of functions which can be wavelet. In particular, if $\psi(t)$ decays to zero at infinity, the admissibility condition implies that all wavelets must have $\hat{\psi}(0) = \int_{-\infty}^{\infty} \psi(t) dt = 0$ in order to make the left-hand side of (2-10) a finite number. This also means that $\psi$ must be a wave that goes up and down along with the $x$ axis.

Under the discrete circumstance, a discrete wavelet transform (DWT, a wavelet series expansion) is given by [Daubechies, 1990]

$$DWT(j, k) = \langle f, \psi_{j,k} \rangle = 2^{j/2} \int_{-\infty}^{\infty} f(t) \psi (2^j t - k) dt \quad (2.11)$$

Now discretizing the function $f(t)$ and setting the sampling rate to be 1, then (2-11) can be written as

$$DWT(j, k) \approx 2^{j/2} \sum_n f(n) \psi_{j,k}(2^j n - k) \quad (2.12)$$

Assumed that $\{\psi_{j,k}(t); j, k, t \in \mathbb{Z}\}$ forms an orthonormal basis of $L^2(\mathbb{R})$. If $\{\psi_{j,k}(t); j, k, t \in \mathbb{Z}\}$ is not an orthonormal basis, we still can obtain $\{DWT(j, k)\}$ using the dual wavelet $\tilde{\psi}_{j,k}(t)$. The detail about dual wavelet can be found in [Daubechies, 1992].

The original function $f(t)$ can be reconstructed with

$$f(t) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} DWT(j, k) \psi_{j,k}(x) \quad (2.13)$$

Definition DWT and other formula presented above are of little practical value. Discrete wavelet transform used in practical applications is the result of an algorithm derived from multiresolution theory made by [Mallat, 1989]. Next, a brief description of multiresolution theory is given and a fast and efficient algorithm is displayed based on the theory.
2.2.2 Multiresolution Analysis

Mallat’s Multiresolution Analysis (MRA) brings out important properties of wavelet and provides an elegant theoretical framework for discrete wavelet transform. Here is the definition:

A multiresolution analysis (MRA) is a sequence of linear function subspace \( \{V_j\} \) such that they possess the following properties:

1) The subspaces are nested: \( \{0\} \subset \cdots \subset V_1 \subset 0 \subset V_{-1} \subset \cdots \subset L^2(R) \)

2) \( \cap_j V_j = \{0\}, \bigcup_j V_j = L^2(R), j \in \mathbb{Z}; \)

3) \( f(t) \in V_j \), iff \( f(2t) \in V_{j-1} \)

4) \( f(t) \in V_0 \) implies \( f(t - k) \in V_0 \) for all \( R \in \mathbb{Z} \)

5) There exists a function \( \phi \in V_0 \) such that the set \( \{\phi_k(x) = \phi(x - k), k \in \mathbb{Z}\} \)
form an orthonormal basis for \( V_0 \).

The main idea of multiresolution analysis is to analyze a signal function at different scales (resolutions). For example, given a discrete function \( f(t) \in L^2(R) \), the decomposition into various scales begins by mapping the function into a sufficiently high-resolution subspace \( V_M \), that is

\[
f(t) \in L^2(R) \mapsto f_M(t) = \sum_k \nu_{M,k} \phi(2^M t - k) \in V_M
\]

(2.14)

where the basic function \( \phi \in V_0 \) is called scaling function since its dilates and translates constitute orthonormal bases for all \( V_j \) subspaces, which are simply scaled versions of \( V_0 \).

Then, an approximation \( f^j \) of the original function \( f(t) \in L^2(R) \) at any resolution (scale) level \( j \) is composed of the sum of the next lower resolution, or averaged (coarser) signal approximation \( f^{j-1} \) and a detail signal function \( g^{j-1} \) made up of fluctuation. These fluctuations provide the added details necessary to produce the full resolution signal \( f(t) \in L^2(R) \). The approximation \( f^{j-1} \) can be expressed as
a linear combination of the elements in the basis of appropriately scaled functions \( \{ \phi_{j,k} : 2^j/2 \phi(2^j t - k); k \in \mathbb{Z} \} \), and also that the detail signal \( g^{j-1} \) is a linear combination of wavelet functions \( \{ \psi_{j,k} : 2^j/2 \psi(2^j t - k); k \in \mathbb{Z} \} \). They are mathematically represented as

\[
\begin{align*}
\begin{cases}
  f^j = f^{j-1} + g^{j-1} \\
  f^j = \sum_k \nu_{j,k} \phi(2^j t - k) \in V^j \\
  g^j = \sum_k \omega_{j,k} \psi(2^j t - k) \in W^j
\end{cases}
\end{align*}
\tag{2.15}
\]

where \( V^j \) is a function subspace and is generated by the bases \( \{ \phi_{j,k} : 2^j/2 \phi(2^j t - k); k \in \mathbb{Z} \} \) and \( W^j \) called wavelet subspace is also a function subspace generated by \( \{ \psi_{j,k} : 2^j/2 \psi(2^j t - k); k \in \mathbb{Z} \} \). \( \psi(t) \in L^2(\mathbb{R}) \) is called the wavelet and constructed by the scaling function \( \phi(t) \) so these \( \psi(t) \in L^2(\mathbb{R}) \) can be expressed by \( \phi \in V_0 \). The basis function \( \phi_{j,k} \) of \( V^j \) together with basis function \( \psi_{j,k} \) of \( W^j \) form a basis for space \( V^{j+1} \). As the scale \( j \) runs from large to small, the corresponding approximations run from fine to coarse, and vice versa.

The deep insight into wavelet analysis is exploited according to the properties of MRA. At each scale \( j \), \( V^j \) is a proper subspace of \( V^{j+1} \) and let the subspace \( W^j \) is an orthogonal complement of \( V^j \) in \( V^{j+1} \), meaning that

\[
\begin{align*}
\begin{cases}
  V^j \oplus W^j = V^{j+1} \\
  V^j \cap W^j = \{0\} \\
  V^l \cap V^m = V^l & m > l \\
  W^l \cap W^m = \{0\} & m \neq l
\end{cases}
\end{align*}
\tag{2.16}
\]

where \( \oplus \) represents the orthogonal sum of two subspaces so the formula \( V^{j+1} = V^j \oplus W^j \) can be recursively extended like

\[
\begin{align*}
V^{j+1} &= W^j \oplus V^{j-1} \\
&= W^j \oplus W^{j-1} \oplus V^{j-2} \\
&= W^j \oplus W^{j-1} \oplus W^{j-2} \oplus \cdots \\
&= \oplus_{l=-\infty}^j W^j
\end{align*}
\tag{2.17}
\]

in our case

\[
\begin{align*}
V^m &= W^{M-1} \oplus V^{M-1} \\
&= W^{M-1} \oplus W^{M-2} \oplus V^{M-2} \\
&= W^{M-1} \oplus W^{M-2} \oplus W^{M-3} \oplus \cdots \\
&= \sum_{n=1}^N \oplus W^{M-n} \oplus V^{M-N}
\end{align*}
\tag{2.18}
\]
That means each function \( f(t) \in L^2(R) \) has a unique orthogonal decomposition

\[
f(t) = \cdots + g^{-1}(x) + g^0(x) + g^1(x) + \cdots
\]

(2.19)

where \( g_j \in W_j \). That is, each component \( f_j(x) \) of function \( f \) has a unique representation in terms of the wavelet series. In our case,

\[
f^M(t) = \sum_{a=1}^{N} g^{M-n}(t) + f^{M-N}(t)
\]

(2.20)

where \( f^{M-N}(t) \) is the coarsest approximation of \( f(t) \).

2.2.3 Fast and Efficient Algorithm

1) Construction of Wavelets
   a) Two-scale relations
      As pointed out by MRA, the scaling functions and wavelets at one scale (coarser) are related to the scaling functions at the next-higher scale by so-called two-scale relations, namely

\[
\begin{cases}
\phi(t) = \sqrt{2} \sum_k h_k \phi(2t - k) \\
\psi(t) = \sqrt{2} \sum_k g_k \psi(2t - k)
\end{cases}
\]

(2.21)

It is obvious that this formula can be recursively operated.

b) Construction of wavelets

For the construction of wavelets, what we do is to find suitable two-scale and decomposition sequences \( \{g_k, h_k\} \). In the following it will be shown that \( \{h_k\} \) can be expressed in terms of \( \{g_k\} \) so that only \( \{g_k\} \) and hence the scaling function need to be constructed.

Since space \( V_j \) and \( W_j \) are orthogonal to each other, that is, \( \Theta \), then so are function \( \phi \) and \( \psi \), meaning that \( \langle \phi, \psi \rangle = 0 \). Therefore,

\[
\langle \phi, \psi \rangle = 2 \sum_m \sum_n h_m g_n \langle \phi_{-1,m}, \psi_{-1,n} \rangle = 2 \sum_k h_k g_k = 0
\]

(2.22)
so that one choice for satisfying the result of above formula is

\[ g_k = (-1)^k h_{1-k} \]  \hspace{1cm} (2.23)

It gives correct solutions. For \( N \) coefficients \( h_0, \ldots, h_{N-1} \), where \( N \) is an even number and the remaining \( h \) coefficient are equal to zero, the choice can be \( g_k = (-1)^k h_{N-1-k} \).

2) Fast and Efficient Algorithm

For an arbitrary finite-energy signal \( f(t) \in L^2(R) \), there is no guarantee that this signal is in any of the approximation subspace \( V_M \). To make use of the two-scale relations for processing, a signal must be in one of these nested approximation subspace. In other words, multiresolution analysis concerns square integrable functions, but the functions have to be adjusted to the spaces \( V_j \) and \( W_j \) for which scaling functions and wavelets form the bases. One way of meeting this requirement is by projecting orthogonally the signal into one of the \( V_j \) for some \( j \). It is particularly important if one only knows the sampled values of the signal at \( f(t = k/2^j, k \in Z) \) for some large value of \( j \).

Assuming that the signal \( f(t) \in L^2(R) \) is not in the approximation subspace \( V_j \), we wish to find \( f(t) \in V_j \) such that

\[ P f(t) \cong f_j(t) = \sum \phi_{j,k} \phi(2^jt - k) \]  \hspace{1cm} (2.24)

where \( \nu_{j,k} \) are the scaling function coefficients to be computed from the signal samples. Using the orthogonal projection of \( f(t) \) onto the \( V_j \) subspace. \( \nu_{j,k} \) can be determined from the sample data \( f(t = k/2^j) \).

Since \( V_j \) is a subspace of \( L^2(R) \) and \( f(t) \in L^2(R) \), we consider \( f^j(t) \) as the orthogonal projection of \( f(t) \) onto the \( V_j \) subspace. Then \( f(t) - f^j(t) \) is orthogonal
to $V_j$ and therefore orthogonal to the basis function $\phi_{j,l}$:

$$< (f(t) - f_j(t)), \phi_{j,l} >= 0 \quad \forall l \in \mathcal{Z} \quad (2.25)$$

Consequently, the coefficients are determined from the equation

$$< f_j(f), \phi_{j,l} >= < f(t), \phi_{j,l} >= < \sum_k C_{j,k} \phi_{j,k}, \phi_{j,l}(t) > \quad (2.26)$$

This yields

$$2^{j/2} \int_{-\infty}^{\infty} \phi(2^j t - l)dt = 2^j \sum_k C_{j,k} [\int_{-\infty}^{\infty} \phi(2^j t - k)\phi(2^j t - l)dt] = \sum_m C_{j,m}[\int_{-\infty}^{\infty} \phi(t)\phi(t - m)dt] \quad (2.27)$$

where we have made a change of index $m = l - k$.

By assuming an orthonormal basis and approximating the integral by a sum, the coefficients are given as

$$C_{j,m} = 2^{j/2} \int_{-\infty}^{\infty} f(t)\phi(2^j t - m)dt \approx 2^{-j/2} \sum_k f(k)\phi(k - m) \quad (2.28)$$

a) Decomposition Algorithm

With the two-scale relations (2-7) and (2-21), we have

$$\psi_{j,k}(t) = 2^{i/2} \psi(2^j t - k)$$
$$= 2^{(j+1)/2} \sum_n g_n \phi(2^{j+1} t - 2k - n)$$
$$= \sum_n g_n \phi_{j+1,2k+n}(t)$$
$$= \sum_n g_n \phi_{j+1,n}(t) \quad (2.29)$$

and thus,

$$< f, \psi_{j,k} > = \int f(t)\psi_{j,k}(t)dt$$
$$= \int f(t) \sum_n g_n\phi_{j+1,n}(t)dt$$
$$= \sum_n g_n < f, \phi_{j+1,n} > \quad (2.30)$$

Similarly,

$$\phi_{j,k}(t) = 2^{i/2} \phi(2^j t - k) = \sum_n h_{n-2k}\phi_{j+1,n}(t) \quad (2.31)$$

Therefore,

$$< f, \phi_{j,k} >= \sum_n h_{n-2k} < f, \phi_{j+1,n} > \quad (2.32)$$
Let $C_{j,k} = \langle f, \phi_{j,k} \rangle$ and $d_{j,k} = \langle f, \phi_{j,n} \rangle$, then

$$C_{j,k} = \langle f, \phi_{j,k} \rangle = \sum_n h_{n-2k} \langle f, \phi_{j+1,n} \rangle = \sum_n h_{n-2k} C_{j+1,k}$$

and

$$d_{j,k} = \langle f, \psi_{j,k} \rangle = \sum_n g_{n-2k} \langle f, \phi_{j+1,n} \rangle = \sum_n g_{n-2k} C_{j+1,k}$$

Substituting $h$ with $g$ by (2-23), the expression of $d_{j,k}$ becomes

$$d_{j,k} = \sum_n (-1)^n h_{n+2k+1} C_{j+1,k}$$

Formulas (2-33) and (2-35) relate the coefficients of the scaling functions and wavelets at any scale to the coefficients at the next higher scale. By repeating this algorithm, all lower-level wavelet coefficients $(j < J)$ can be computed from the scaling function coefficients using (2-33) and (2-35) so we can construct finer resolution of a function $f$ from its coarser resolution and the differences between these resolution.

This decomposition algorithm is represented schematically in figure 2.1(upper).

![Figure 2.1 Diagram for decomposition and reconstruction algorithms](image)

b) Reconstruction Algorithm
It is important for any transform to have an unique inverse such that the original data can be recovered perfectly. For random signals, some transforms have their unique inverse in theory but cannot be implemented in reality. There exists an unique inverse discrete wavelet transform (or synthesis transform) such that the original function can be recovered perfectly from its components at different scales. The reconstruction algorithm is based on the two-scale relations of the scaling function and wavelet. Let us consider a sum of these components at the jth resolution.

Because $V_{j+1} = V_j \oplus W_j$, then for a function $f^{j+1} \in V_{j+1}$, we have

$$\sum_k C_{j+1,k} \phi_{j+1,k}(t) = f^{j+1}(t)$$

$$= f^j(t) + g^j(t) \text{for } f^j \in V_j \text{ and } g^j \in W_j$$

so that

$$C_{j+1,k} = < f^{j+1}, \phi_{j+1,k} > = < f^j + g^j, \phi_{j+1,k} >$$

$$= < f^j, \phi_{j+1,k} > + < g^j, \phi_{j+1,k} >$$

$$= < \sum_n C_{j,n} \phi_{j,n}, \phi_{j+1,k} > + < \sum_n d_{j,n} \psi_{j,n}, \phi_{j+1,k} >$$

$$= \sum_n C_{j,n} < \phi_{j,n}, \phi_{j+1,k} > + \sum_n d_{j,n} < \psi_{j,n}, \phi_{j+1,k} >$$

$$= \sum_n \phi_{j,n} \phi_{j+1,k} + \sum_n \phi_{j+1,k}$$

(2.36)

(2.37)

The last expression above is due to $< \phi_{j+1,m}, \phi_{j+1,n} > = 1$ for $m = k$ and 0 otherwise using (2-23)

$$C_{j+1,k} = \sum_n h_{n-2k} C_{j,n} + \sum_n (-1)^k h_{2n-k+1} d_{j,n}$$

(2.38)

The process of reconstructing coefficients $C$ is summarized in figure 2.1(lower).

It is noted that coefficients $h_k$ are sufficient to find the wavelet transform; in particular, there is no need to know the scaling function or mother wavelet to accomplish the task. As a matter of fact, fast wavelet transform is the way of determining the wavelet transform without referring to the wavelet at all. What we need are the tables of $h_k$ (and $g_k$) values to perform the computations. Scaling function and mother

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
wavelet were used to determine these values, but afterwards, these functions are no longer needed.

c) Linear Time Complexity.

The algorithms discussed in above sections are said fast and efficient because the time complexity of operations is linear. At here, a rough proof is given [Ogden, 1997].

Let $K$ denotes the number of non-zero terms used in the sequence. Computing a single coefficient at level $j$ according to or would take at most $K$ operation. Suppose giving exactly $M$ non-zero coefficients at level $j + 1$, thus, the total number of operations required to complete the one-level-down wavelet and scaling function coefficient is approximately $2K \cdot (M/2)$. There will be approximately $M/4$ non-zero scaling function coefficients at level $j - 1$ and the computation will requir approximately $2K(\frac{M}{4})$. Continuing in this way, we see that the total number of operations required doing all the decomposition is approximately

$$2K(\frac{M}{2} + \frac{M}{4} + \frac{M}{8} + ...) = O(M)$$ (2.39)

Given a set of top-level scaling function coefficients, this decomposition is very efficient, even better than the $O(M \log M)$ operations required for the fast Fourier transform algorithm.

Of course, everything we have discussed so far has assumed that the top-level scaling function coefficients are provided as input. The remarkable efficiency of the decomposition algorithm is moot without an efficient algorithm for computing the top-level coefficients. Regarding the original data $Y_1, \ldots, Y_n$ as a low-pass filter $H$ (the concepts of low-pass and high-pass filters will be explained in the next section) will result in a smoother version of a signal. This is precisely what is done in the standard discrete wavelet transform of data - the filters $H$ and $G$ are applied to $Y_1, Y_2, \ldots, Y_n$. The standard wavelet decomposition thus regards the input data $Y_1, \ldots, Y_n$ as the
highest-level scaling function coefficients, or, equivalently, the least-smoothed signal. Thus, there is no computational expense at all in computing the $C_{j,k}$ sequence, and the entire wavelet decomposition requires only $O(M)$ operation. In practice, then, it is possible to do a full wavelet decomposition and reconstruction without performing any numerical integration, indeed, without ever computing a single value of any wavelet function.

It should be noted here that the wavelet reconstruction algorithm is also fast, requiring $O(M)$ operation.

### 2.2.4 The Filter Representation

The wavelet decomposition and reconstruction algorithms previously discussed present the mechanism how the wavelet transform analyzes and processes a signal function. Additional insight into some of the practical issues in wavelet analysis can be gained by regarding these algorithms as example of signal processing filters. The subband transform associated with filter banks displays the other aspects of signal transform.

Subband transform decomposes the input signal into several frequency components, or subband, and then encode these components separately. The subbands are generated by using a bank of filters, with each filter passing signals of a specified frequency range (Figure2.2). In mathematical point of view, a filter is a linear operator defined in terms of its filter coefficients, $h(0), h(1), h(2), \ldots$. The subbands are gotten by taking the convolution of the input signal with a set of bandpass filters and decimating the results. Each decimated set of transform coefficients is a subband signal that encodes a specific range of the frequencies of the input. In other words, an output vector (subband) $\tilde{x}$ is produced by applying a filter to an input vector $x$,
according to
\[ \hat{x}(x) = \sum_{n} h(k)x(n - k) = h \ast x \]  
(2.40)
where the symbol * indicates a convolution. Because statistical characteristics of the subbands are different, each subband can be controlled separately. That is each subband can be encoded differently and thus more efficiently than the original signal. Reconstruction is done by unsampling, follow by computing the inverse transforms, and adding the resulting sets of outputs from the inverse filters.

Figure 2.2 Diagram of filter banks

The wavelet transform can be interpreted as a bank of two filters, a lowpass and a highpass. For a given wavelet basis, represent the filter \( H \) by the sequence \( \{ h_k \} k \in \mathbb{Z} \) given in \( h_k = < \phi, \phi_{1,k} > \). The concepts of wavelet decomposition and reconstruction algorithms may be expressed in terms of these filtering operations by regarding the set of scaling function coefficients at a particular level as a signal: \( \{ C_{j,k} \} k \in \mathbb{Z} \). Scaling function coefficients at the next lower level are obtained by applying the filter \( H \) to the signal
\[ C_{j,.} = \{ C_{j,k} \} \quad k \in \mathbb{Z} ; \]
\[ C_{j-1,.} = HC_{j,.} \]  
(2.41)
which corresponding to \( C_{j,k} = \sum_{l} h_{l-2k}C_{j+1,l} \). In fact, scaling function coefficients at any level can be obtained by repeatedly applying the filter \( H \):
\[ C_{j-m,.} = H^mC_{j,.} \]  
(2.42)
Now define a new filter $G$ by

$$g_k = (-1)^k h_{1-k}, \quad k \in \mathbb{Z}$$

(2.43)

where the $h_k$'s are again defined as in $h_k = \langle \phi, \phi_{1,k} \rangle$. Wavelet coefficients at level $j - 1$ can be obtained from scaling function coefficients at level $j$ via this filter.

$$d_{j-1,c} = GC_{j,c}$$

(2.44)

which corresponds to $d_{j,k} = \sum_{l \in \mathbb{Z}} (-1)^l h_{-l+2k+1} C_{j+1,l}$. By combining these two filters, wavelet coefficients at any lower level can be computed from scaling function coefficients at level $j$:

$$d_{j-m,c} = GH^{m-1} C_{j,c}$$

(2.45)

Now that the decomposition algorithms presented in previous section have been written in terms of these filtering concepts, it is natural to inquire into the nature of the filters $H$ and $G$. The filter $H$ is known as a lowpass filter which correspond to averaging operations, while $G$ is an example of a highpass filter corresponding to differencing.

Usually, applying the $H$ filter results in a signal composed of localized weighted averages performed on the original signal. And also, applying the $G$ filter results in a signal whose elements are contrasts of localized elements of original signal. This idea is reinforced by summing the coefficients of the filter representation, That is, for a lowpass (averaging or smoothing) filter $H$,

$$\sum_{k \in \mathbb{Z}} h_k = \sqrt{2}$$

(2.46)

For a highpass (contrasting or detail) filter $G$,

$$\sum_{k \in \mathbb{Z}} g_k = 0$$

(2.47)
It is noted that, in decomposition algorithms, if the dilation index $k$ is increased by one, the indices of the \{\{h_t\}\} sequence are all offset by two. Thus, in computing either decomposition, there is an inherent down-sampling of coefficients that correspond to the decimation in subband transform. Looking at the algorithm for reconstruction, after going up every scale $j$, the total function coefficients will be counted by both $C_j$ and $d_j$ which are called up-sampling as same as interpolation happened in subband transform (shown in figure 2.3)

![Figure 2.3 Representation diagram of decomposition algorithm](image)

The main advantage of subband transform is that they isolate the different frequencies of the input signal, thereby making it possible for the user to precisely control the loss of data in each frequency range. In practice, such a transform decomposes a signal into several subbands, corresponding to different signal frequencies. and each subband can be handled differently. On contrast, the major drawback of this type of transform is the introduction of artifacts, such as aliasing and ring, into the reconstructed signal, because of the down-sampling. This is why a lot of different types of wavelets have been designed and studied for finding better sets of filters.
2.3 An Example: Harr Wavelet Transform

2.3.1 Definition: Harr Wavelet

The Harr wavelet is nothing new, having been developed by Harr (1910). The Harr basic wavelet is a step function and defined by

$$\psi_H(t) = \begin{cases} 1 & 0 \leq t < 1/2 \\ -1 & 1/2 \leq t < 1 \\ 0 & \text{otherwise} \end{cases}$$

and corresponding wavelets are defined as

$$\psi_{j,k}(t) = 2^{j/2}\psi_H(2^j t - k) \begin{cases} 1 & k^j \leq t < k^j + 2^{j-1} \\ -1 & k^j + 2^{j-1} \leq t < (k+1)2^j \\ 0 & \text{otherwise} \end{cases}$$

The Harr wavelet transform uses a scale function $\phi(t)$ and a wavelet $\psi(t)$, both shown in Figure 2.4, to represent a large number of functions. The representation is

![Figure 2.4 Harr wavelet mother function](image)

the infinite sum

$$f(t) = \sum_{k=-\infty}^{\infty} C_k \phi(t-k) + \sum_{k=-\infty}^{\infty} \sum_{j=0}^{\infty} d_{j,k} \psi(2^j t - k)$$

where $C_k$ and $d_{j,k}$ are coefficients to be calculated.
The basic scale function $\phi(t)$ is the unit pulse

\[
\phi(t) = \begin{cases} 
1 & 0 \leq t < 1 \\
0 & \text{otherwise}
\end{cases}
\]  

(2.51)

and

\[
\phi_{j,k} = 2^{j/2} \phi_H(2^j t - k) \begin{cases} 
1 & k2^j \leq t < (k+1)2^j \\
0 & \text{otherwise}
\end{cases}
\]  

(2.52)

In practice, the shifted copies are used to approximate $f(t)$ at different time $t$. The scale copies are used to approximate at different resolutions (scales). It is clear that from the subband transform point of view,

\[
\phi_H(t) = \phi_H(2t) + \phi_H(2t - 1)
\]  

(2.53)

and compared with $\phi(t) = \sqrt{2} \sum_{k=-\infty}^{\infty} h_k \phi(2t - k)$ (two-scale relation), we obtain $h_0 = h_1 = 1/\sqrt{2}$. Hence, the lowpass filter $H$ for the Harr system consists of only two non-zero elements: $H_0 = H_1 = 1/\sqrt{2}$. In general, for a signal $f = \{f_k\}$ $k \in \mathbb{Z}$, the elements of the filtered signal $f' = Hf$ using the Harr filter are $f'_k = 1/\sqrt{2}(f_{2k} + f_{2k-1})$ which is proportional to the average of adjacent elements. Moreover, $\psi_H(t) = \phi_H(2t) - \phi_H(2t - 1)$ and thus $\psi(t) = \sqrt{2} \sum_{k=-\infty}^{\infty} g_k \phi(2t - k)$ is satisfied for $g_0 = h_1 = 1/\sqrt{2}$, $g_1 = -h_0 = -1/\sqrt{2}$.

As the same situation as that for detail functions, applying the $G$ filter corresponding to the Harr system to the same signal $f$ would give a signal $f^*$ with elements

\[
f^*_k = \frac{1}{\sqrt{2}}(f_{2k} - f_{2k-1})
\]  

(2.54)

which is proportional to the difference between adjacent elements.

### 2.3.2 The ordered fast Harr wavelet Transform

Because our research focuses on digital speech signal, only one dimension array of the signal is considered. At here, one of the Ordered Fast Harr Wavelet Transform
algorithms is introduced [Yves Nievergelt, 1999]. Before describing the algorithm, the definition of Harr scale function is modified to construct a step function at a different height $C$, starting at the location $u$ and ending $w$. A scalar multiple by $C$ of the function so that

$$C, \phi_{[u,w]}(t) = \begin{cases} C & \text{if } u \leq t < w \\ 0 & \text{otherwise} \end{cases}$$

(2.55)

Step 1: Supposing there is a one-dimensional array of the form

$$\bar{a}^{(n)} = (a_0^{(n)}, a_1^{(n)}, \ldots, a_j^{(n)}, \ldots, a_{2^n-2}^{(n)}, a_{2^n-1}^{(n)})$$

$$= \bar{s} = (s_0, s_1, \ldots, s_j, \ldots, s_{2^n-2}, s_{2^n-1})$$

(2.56)

with a total number of sample values equal to an integral power of two, $2^n$, as indicated by the superscript $n$. The array correspond to the sampled step function

$$\tilde{f}^{(n)} = \sum_{j=0}^{2^n-1} a_j^{(n)} \phi_j^{(n)}$$

(2.57)

Step 2: The first sweep. The shifted and dilated basic transform applies to all the consecutive pairs of values, separated here by semicolons for convenience, in a sample with $2^n$ value $s_0, s_1; s_2, s_3; \cdots; s_{2k}, s_{2k-1}; \cdots; s_{2(n-1)}, s_{2n-1}$

$$\bar{s}^{(n-1)} = (\bar{a}^{(n-1)}, \bar{d}^{(n-1)}); a_k^{(n-1)} := \frac{a_{2k}^{(n-1)} + a_{2k+1}^{(n-1)}}{2}; d_k^{(n-1)} := \frac{a_{2k}^{(n-1)} + a_{2k+1}^{(n-1)}}{2}$$

(2.58)

and put these elements into $a^{(n-1)} = \{a_0^{(n-1)}, a_1^{(n-1)}, \ldots, a_k^{(n-1)}\}$ and $d^{(n-1)} = \{d_0^{(n-1)}, d_1^{(n-1)}, \ldots, d_k^{(n-1)}\}$

Step 3: Repeat the step 2 until the $l$th sweep.
CHAPTER 3

AUDIO COMPRESSION OPERATION 2: PREDICTION

3.1 Digital Audio

Usually, audio is recorded through a microphone and is converted into a voltage that varies continuously with time. Such voltage is the analog representation of the sound. Now most speech signal processing operations are carried out using digital technology because Digital technology is more advanced and more powerful than analogue technology. For example, digital signals are less sensitive to transmission noise than analogue signals. It is easy to error-protect and encrypt digital signals so that digital transmission can be made very secure. In addition, digital signals of different types can be treated in a unified way and, provided adequate decoding arrangements exist, can be mixed on the same channel. Of course, digital technology has some disadvantages. One of them is that it requires greater channel bandwidth.

To convert the analogue speech signal into a digital format in which it is represented by a sequence of numbers, it involves two processes, sampling and quantization. This two-stage process is sometimes referred to as Pulse Code Modulation (PCM). Sampling is the process of obtaining values of the analogue signal at discrete instants of time, and quantization refers to the conversion of the amplitude at each sampling instant into a discrete binary number with a specified bit-length. The less the numbers used in quantization, the better the compression, but also the greater the loss of information. this aspect of quantization is used by several speech compression meth-
ods. In our research, we use some digitized speech file, so this issue is not concerned in this thesis.

3.2 Prediction

3.2.1 Differential Pulse Code Modulation (DPCM)

An analysis of speech signals shows that adjacent speech samples tend to be similar (correlation), particularly in regions of voiced speech, and thus audio samples tend to have redundancies. The simplest way to exploit this redundancy is to subtract adjacent samples and code the differences, which tend to be small numbers. Following this idea, the value of each sample is predicted based on the values of a number of neighboring samples. This predicted value represents redundant information that can be omitted; therefore, instead of encoding a sample itself, the difference (prediction error) between its value and the predicted value is encoded. The decoder uses the quantized difference to reconstruct the original sample by computing its predicted value from already decoded sample values and adding them to the encoded prediction error. Any method based on this principle is called DPCM (differential pulse code modulation).

3.2.2 Adaptive Differential Pulse Code Modulation (ADPCM)

In the foregoing discussion, DPCM was using a fixed set of predictor coefficients. So in most time, such methods are inefficient, since they do not adapt themselves to the varying magnitudes of the audio stream. That means different coefficients should be used for different segments of samples, which leads to the concept of adaptive DPCM. There are at least two components of adaptive DPCM that can be modified: predictor coefficients and quantization levels. ADPCM can be based on adjusting
only one of these two components. But it is possible to modify both.

Adaptive quantization attempts to make the quantizer design adapt to the varying input statistics in order to achieve better performance. By statistics, we mean the statistical mean, variance (or the dynamic range), and type of input pdf (probability distribution function). When the mean of the input changes, differential coding is suitable method to handle the variation. For other types of cases, adaptive quantization is found to be effective. The price paid for adaptive quantization is processing delays and an extra storage requirement.

Similar to the discussion on adaptive quantization, adaptive prediction can be done in two different ways: forward adaptive and backward adaptive prediction. In the former, adaptation is based on the input of a DPCM system, in other words, it periodically modifies the predictor coefficients so that they reflect the change in the stream of input samples. In the later, adaptation is based on the output of the DPCM, that means predictor coefficients are estimated based on quantized and transmitted data. Therefore, forward adaptive prediction is more sensitive to changes in local statistics. Prediction parameters (the coefficients of the predictor), however, need to be transmitted as side information to the decoder. On the other hand, quantization error is involved in backward adaptive prediction. Hence, the adaptation is less sensitive to local changing statistics. But it does not need to transmit side information.

Something in detail about the later ADPCM method is discussed here. This kind of ADPCM uses the previous sample (or several previous samples) to predict the current sample. It then computes the difference between the current sample and its prediction, and quantizes the difference. For each input sample \( X[n] \), the output \( C[n] \) of the encoder is simply a certain number of quantization levels. The decoder
multiplies this number by the quantization step (and may add half the quantization step, for better precision) to obtain the reconstructed audio sample. The method is efficient because the quantization step is modified all the time, by both encoder and decoder, in response to the varying magnitudes of the input samples. It is also possible to modify adaptively the prediction algorithm.

Various ADPCM methods differ in the way they predict the current sound sample and in the way they adapt to the input (by changing the quantization step size and/or the prediction method).
CHAPTER 4

AUDIO COMPRESSION OPERATION 3: CODING

Coding is the last important operation in the process of lossless compression. Coding itself is also a topic of computer science. Giving a whole description of coding theory is out of scope of this thesis so only some concepts and methods to be used in our research are explained at here.

4.1 Terminology

Given a source with alphabet \( S = \{x_1, x_2, \ldots, x_n\} \) which is a finite set, and probabilities \( P = \{p_1, p_2, \ldots, p_m\} \) for which letters \( x_i \) are encoded as strings or codewords, \( C = \{c_1, c_2, \ldots, c_m\} \), a code is a mapping from \( S \) onto \( C \), that is \( c = x_{i_1}x_{i_2}\ldots x_{i_k} \). The length \( l_i \) of a string \( c \) is the number of alphabets appearing in the string. A code whose alphabet is \( \{0, 1\} \) is called a binary code. The unary code of a nonnegative integer \( n \) is defined as \( n - 1 \) ones followed by a single 0, or alternatively, as \( n - 1 \) zeros followed by a single one. The length of the unary code for the integer \( n \) is thus \( n \) bits. A code has a prefix (or irreducibility) property if no codeword can be obtained from another codeword by adding more 0s or 1s, or no codeword is a prefix of another codeword. In other words, prefix means once a certain bit pattern has been assigned as the code of a symbol, no other codes could start with that pattern (the pattern can not be the prefix of any other code). Variable-size codes are such codes that they should satisfy the prefix property and should be assigned to symbols based on their probabilities. On the other hands, fixed-size codes indicate that each symbol in this
code has the same length. A variable-size code has less redundancy than a fixed-size code but it is more difficult to represent and implement than fixed-size code.

4.2 Fundamental Limits

In the section above, we got some concepts how to build a code. Now we wonder a question: given a code, what is its lower bound on the expected length of the code? In data compression, we are interested in minimizing the expect (average) cost $L_{\text{avg}} = \sum_{i=1}^{n} p_i l_i$. The same question is: what this minimum cost could be? The answer is: it is Shannon's entropy.

In 1940s Claude E. Shannon discovered that the extent to which a message can be compressed and then accurately restored is limited by its entropy [Weiss, 1993]. Where entropy is a measure of information. The higher the entropy of a message, the more information it contains. It can be expressed in

$$H(P_1, \ldots, P_n) = H(S) = -\sum_{i=1}^{n} P_i \lg(P_i)$$

(4.1)

where $S = \{x_1, \ldots, x_n\}$ is a set of $n$ independent events with the set of probabilities of their occurrence $P = \{p_1, \ldots, p_n\}$. It is typical convention that the logarithm is taken base 2, in which case entropy is measured in bits. So the function shown above represents the average number of bits of all individual symbols(or integer numbers).

Our goal is to design an optimum code with prefix property. So another question rises: in what situation is this possible to avoid looking for such a code when it cannot be found? Some information concerning this possibility can be garnered from a theorem proven by Leon G Kraft in his 1949 master's thesis.

Kraft's theorem: There exists a prefix binary code with codewords $\{c_1, \ldots, c_m\}$ with corresponding lengths $\{l_1, \ldots, l_m\}$ iff

$$\sum_{i=1}^{n} 2^{-l_i} \leq 1$$

(4.2)
At here just is the statement. The proof will be found in [Adam Drozdek, 2002]. The theorem refers to prefix codes only. This theorem guarantees finding a prefix code for a given set of lengths but do not change the lengths whereby the constructed code may not be optimal. The Kraft inequality allows us to decide whether a prefix code exists if length of prospective codewords is given.

4.3 Some lossless Coding techniques

One of the simplest techniques for lossless compression is Run Length Encoding. The idea behind this approach to data compression is this: If a data item $d$ occurs $n$ consecutive times in the input stream, replace the $n$ occurrences with the single pair $nd$. The $n$ consecutive occurrences of a data item are called a run length encoding or RLE. Run Length Encoding is suited for compressing any type of data regardless of its information content but the content of the data will affect the compression ratio achieved by Run Length Coding.

Based on the theorems described above and other principles derived from them, people invented a lot of coding techniques. The most popular lossless coding techniques are Huffman and arithmetic methods. Roughly speaking, both Huffman and arithmetic methods are example of statistical coding, where character frequency statistics are used to allocate appropriate codewords for output. Huffman coding is a universal entropy compression method which produces variable length code. Symbols, which occur with a high probability, are assigned short codewords while symbols, which are rare, are assigned longer codewords. Huffman codes have the unique prefix attribute, which means they can be correctly decoded despite their variable length. Arithmetic coding enables characters to be represented as fractional bit lengths and replaces a stream of input symbols with a single floating point output number. This
is done by representing the source as a real number, greater than or equal to zero, but less than one, as denoted as the range \([0, 1)\). In this scheme, sequences of characters are represented by individual codes, according to their probability of occurrence. The advantage of arithmetic coding is that it can code arbitrarily close to the entropy.

In addition to Huffman and arithmetic coding, the simplest coding methods are those that ignore or make only minimal use of the supplies probabilities. In doing so, their compression effectiveness may be relatively poor, but the simple and regular codewords that they assign can usually be encoded and decoded extremely quickly. Moreover, some compression applications are such that the source probabilities \(p_i\) have a distribution to which the regular nature of these non-parameterized codes is well suited. Golomb and Rice codings are among this kind of coding methods.

**Golomb Code** is the method to generate a variable-size code for integers \(n\). It depends on the choice of a parameter \(b\) and it is created in two steps

1. Compute the two quantities
   \[
   q = \lfloor (n - 1)/b \rfloor, \quad r = n - qb - 1.
   \]
2. Construct the golomb code of \(n\) in two parts; the first is the value of \(q + 1\), coded in unary, and the second, the binary value of \(r\) coded in either \(\lfloor \log_2 b \rfloor\) bits (for the small remainders) or in \(\lceil \log_2 b \rceil\) bits (for the large ones).

**Rice coding** is a specialized form of Golomb coding. Rice coding makes most sense when most of the number to be coded are relative small. This suits the "exponential" or "laplacian" distribution of numbers which typically make up the difference of a lossless audio encoding algorithm. In other words, if most of the numbers are small, fairly good compression can be achieved. Rice coding is generally used to encode entropy in an audio/video codec.
Here is a little more technical description of the process of Rice coding:

1. Make your best guess as to how many bits a number will take, and call that $k$
2. Take the rightmost $k$ bits of the number and remember what they are
3. Imagine the binary number without those rightmost $k$ bits and look at its new value (this is the overflow that doesn't fit in $k$ bits)
4. Use these values to encode the number. This encoded value is represented as a number of zeroes corresponding to step 3, then a terminating 1 to tell that your done sending the "overflow", then the $k$ bits from step 2.

Decoding works the same way, just backwards. In general, a lower value of $k$ will make smaller numbers cheaper and bigger numbers more expensive to store, while a bigger value of $k$ will make big numbers relatively cheap to store, while increasing the storage overhead on all smaller values and making them more expensive to store.

To losslessly compress sound file, conventional compression methods, such as RLE, statistical, can be used, but the results depend heavily on the specific sound. RLE may work well when the sound contains long runs of identical samples. With 8-bit samples this may be common. With 16-bit samples, long runs may be rare and RLE, consequently, ineffective. Statistical methods assign variable-size codes to the samples according to their frequency of occurrence. It works well for the sound file which features skewed probabilities. With 16-bit samples there are more than 65,000 possible samples, so it may match the situation. However, applying Huffman or arithmetic coding directly to the audio is not very efficient due to the long-time correlation in audio signal. A preprocessing stage, so-called prediction which eliminates the redundancies within audio signal, leads to an almost uncorrelated source which is easier to code. These treated data usually are relatively small and satisfying with the "laplacian" distribution. Therefore, Rice coding is a better choice for use to compress
a digitized audio file.
CHAPTER 5

A NEW LOSSLESS COMPRESSION ALGORITHM

The goal of our research is to find a fast, efficient method to compress a audio file losslessly because lossless audio coding is a topic of high interest for both professional and customer applications. Modern lossy coding standards (e.g. ISO MPEG 1 and 2) can achieve large compression ratio with subjective quality; however, multiple coding can reveal originally masked distortions. In addition, reproduction of critical music items shows that even the best systems cannot be considered as truly transparent. Lossless coding is able to overcome these drawbacks although it will sacrifice some compression ratio, comparing with lossy coding.

5.1 Scheme

The main idea of our compression system (encoder) is that after a stream is input, a wavelet transform is operated on the stream, then use a predictor to approximate data of the audio signal, which are filtered by wavelet transform, after that, compute a set of difference values (called error) between the prediction and the data. These difference values are relative small integers (in general) and finally these values are compressed using Rice coding. In the decoder, the predictor is replicated and it must work from prediction parameters and the previously decoded data. The predicted value is added to the “error” value to create the final exactly correct value for that data. Then the prediction algorithm is run again, based on the newly decoded data and some previous ones to predict the next data. After recovering all the data, use
There are two important issues, associated with the compression system we design, have to be considered before we test and analyze the system. The first one is what kind of criteria can be used to measure quantitatively the performance of a compression method. List quantities below are such kind of measurement that is commonly used.

\[
\text{Compression ratio} = \frac{\text{Length of the output stream}}{\text{Length of the input stream}} \times 100\% \tag{5.1}
\]

Inverse of the compression ratio is called the compression factor which is expressed as

\[
\text{Compression factor} = \frac{\text{Length of the input stream}}{\text{Length of the output stream}} \times 100\% \tag{5.2}
\]

\[
\text{Compression rate} = \frac{\text{Length (input)} - \text{Length (output)}}{\text{Length (input)}} \times 100\% \tag{5.3}
\]

Of course, entropy is also often used as low bound of compression because audio files cannot be compressed losslessly to any size smaller than its entropy. The per-

---

**Figure 5.1** The systematic diagram of our compression system

wavelet reconstruction algorithm to rebuild the up-level scale parts and repeat that again and again until to the level in which the data are exactly the same as audio signal samples. The figure (5-1) briefly displays these ideas.
formance can only approach more closely whatever the basic entropy of the audio file is. The more close to that entropy, the better performance the compression has.

The second one is about audio files which are used here for evaluating our lossless compression method. The format of the digitized speech file used in our research is so-called WAV file. This format is very popular upon IBM PC (clone) platform and it supports a variety of bit resolutions, sample rates, and channels of audio. Unlike other compressed formats, such as MPEG, WAVs store samples "in the row" where no pre-processing is required other than formatting of the data. This is why the format is chosen in our research. The WAV file itself consists of basic three important "chunks" of information (there may also be other chunks in a WAV file that contain some information which are not needed to play the recorded sound). These three chunks are: the RIFF chunk which identifies the file as a WAV file, the FORMAT chunk which identifies parameters such as sample rate, etc, and the DATA chunk which contains the actual data (samples). The data is encoded in the WAV file by Pulse Code Modulation (PCM).

Table 5.1 Some information of ten digital speech test files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Channel</th>
<th>Bits/Sample</th>
<th>Sample Rate</th>
<th>Num. of Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>baita1</td>
<td>Mono</td>
<td>16</td>
<td>16000 Hz</td>
<td>3848</td>
</tr>
<tr>
<td>beeta1</td>
<td>Mono</td>
<td>16</td>
<td>16000 Hz</td>
<td>3612</td>
</tr>
<tr>
<td>bitea1</td>
<td>Mono</td>
<td>16</td>
<td>16000 Hz</td>
<td>5588</td>
</tr>
<tr>
<td>boga1</td>
<td>Mono</td>
<td>16</td>
<td>16000 Hz</td>
<td>7302</td>
</tr>
<tr>
<td>fighta1</td>
<td>Mono</td>
<td>16</td>
<td>16000 Hz</td>
<td>7117</td>
</tr>
<tr>
<td>floata1</td>
<td>Mono</td>
<td>16</td>
<td>16000 Hz</td>
<td>7697</td>
</tr>
<tr>
<td>gnat1</td>
<td>Mono</td>
<td>16</td>
<td>16000 Hz</td>
<td>6142</td>
</tr>
<tr>
<td>sata2</td>
<td>Mono</td>
<td>16</td>
<td>16000 Hz</td>
<td>7803</td>
</tr>
<tr>
<td>slighta1</td>
<td>Mono</td>
<td>16</td>
<td>16000 Hz</td>
<td>8303</td>
</tr>
<tr>
<td>slopea1</td>
<td>Mono</td>
<td>16</td>
<td>16000 Hz</td>
<td>8594</td>
</tr>
</tbody>
</table>
The true nature of sound varies with its source, physical environment and recording method, and all these factors will affect on audio files. So a lossless compression program cannot adapt itself entirely to the "true" nature of the sound. Therefore it is not surprising that different algorithm work best on different kind of audio. Some information of the files we use is listed in table 5.1.

Next we will discuss each main function in our compression algorithm and give the test results and analyses.

5.1.1 Stage 1: Wavelet Transform

The first step in our lossless compression is to more efficiently model the data of input audio signal. There are often some forms of relationship between those data, and this can be exploited several ways, with one popular way being through the use of wavelet transform and prediction.

Given a sequence of $N$ integers $s_i$, where $i = 0, 1, \ldots, N - 1$ and $s$ represents the original data of audio signal at the beginning of decomposition $j = 0$, after that it represents the scale (coarse) component at level $j = -n$. Meanwhile, let $w_i^j$ represents the wavelet component at level $j = -n$. These $N/2$ numbers becomes the low-frequency transform coefficients, and are normally transformed again, into $N/4$ low-frequency and $N/4$ high-frequency coefficients, so forth (Figure 5.2).

Then the decomposition algorithm of Harr wavelet transform used in this case can be expressed as

\[
\begin{align*}
    w_i^{j-1} &= s_{2i}^j - s_{2i+1}^j & \text{for } i = 1, 2, \ldots, N/2 \\
    s_i^{j-1} &= \frac{s_{2i-1}^j + s_{2i}^j}{2} = s_{2i}^j + \frac{w_i^{j-1}}{2} & \text{for } i = 1, 2, \ldots, N/2 \\
    w_i^{j-1} &= s_N^j - s_{N-1}^j & \text{for } i = N/2 + 1 \text{ and } N = \text{odd} 
\end{align*}
\]

(5.4)

If we want to decompose the data down to $k$ level, the formulas above can be iteratively operated upon scale component of up-level $j - 1$ until reaching level $k$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The reconstruction of the original data is easy to figure out based on the inverse transform displayed above. The expression of this process is

\[
\begin{align*}
    s^j_{2i} &= s^{j-1}_i - \frac{w^j_{i-1}}{2} & \text{for } i = 1, 2, \ldots, N/2 \\
    s^j_{2i+1} &= w^j_{i-1} + s^j_{2i} & \text{for } i = 1, 2, \ldots, N/2 \\
    s^j_N &= w^j_{N-1} + s^j_{N/2} & \text{for the last odd number } N = \text{odd}
\end{align*}
\]  

(5.5)

It needs to mention that the transforms above are lossless although there is a division by 2 in these formulas. The division causes so-called truncation which is denoted by the "floor" symbols \([\_\_\_]\) and \([\_\_\_]\) and is used to produce integer transform coefficients \(s^j_{i-1}, w^j_{i-1}\) and also integer reconstructed data items \(s^j_i\). Because of truncation, some information is lost when the \(s^j_{i-1}\) is calculated. However, truncation is also used in the calculation of the \(s^j_{2i+1}\), which restores the lost information. Thus, Equations above is a true forward and inverse that reconstructs the original data items exactly.

5.1.2 Stage 2: Predictor

The goal behind audio compression is to make all of the sample values as small as possible by removing any correlation that may exist between them. In our compression algorithm, What we did for this purpose is to let the data of all wavelet components (high-frequency) be passed through a predictor to remove any redundancy.
while still remaining compressible. This stage is what separates one compression scheme from another. There are virtually countless ways to do so. People can make a predictor based upon some reasonable assumption or their experience. Then, these predicted values are compared with the actual value and the difference (error) is what gets sent to the next stage for encoding. From this point, a good predictor is crucial for a good compression system.

In this work, we design two predictors which are expressed by formula 1 and formula 2. The first predictor is made based on the assumption that there is correlation between scale and wavelet components at the same decomposed level. For the second predictor, the assumption is that there exists some relationship among neighboring samples in wavelet component. These predictors will be tested and the results are shown in table 5.2 and table 5.3.

Formula 1:

\[
\hat{W}_i^j = a(S_i^j - S_{i-1}^j) + b(S_{i+1}^j - S_i^j) + cW_{i-1}^j
\]  

(5.6)

Formula 2:

\[
\hat{W}_i^j = aW_{i-1}^j + bW_{i-2}^j + cW_{i-3}^j
\]  

(5.7)

where \(a, b, c\) are parameters to be tested and tuned such that the average error (difference between predicted values and their corresponding actual values) approaches to zero as close as possible. We use brute-force method to scan the values from -5 to 5 for all \(a, b, c\) and try to find the best set of these parameters at each decomposed level. In table 5.2 and 5.3, we show our test results of the minimum average error at every decomposed level and corresponding sets of parameters \(a, b, c\) for both formulas.

It is noted that at here we choose average error as a measurement because the smaller the average error, the more the number of small error values, and this situation is desirable for Rice coding at next stage. The quantities of average error also help
us to decide an important parameter $k$ in Rice coding, which are the number of least significant bits. These bits are regarded as uncompressible and will be sent directly to the output file.

Table 5.2 The average errors of ten decomposed levels using predictor 1

<table>
<thead>
<tr>
<th>File Name</th>
<th>-1</th>
<th>-2</th>
<th>-3</th>
<th>-4</th>
<th>-5</th>
<th>-6</th>
<th>-7</th>
<th>-8</th>
<th>-9</th>
<th>-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>baita1</td>
<td>707.1</td>
<td>3442</td>
<td>2234</td>
<td>2241</td>
<td>4946</td>
<td>3367</td>
<td>2062</td>
<td>1126</td>
<td>609</td>
<td>239</td>
</tr>
<tr>
<td>beeta1</td>
<td>736.9</td>
<td>2675</td>
<td>1043</td>
<td>981.8</td>
<td>3610</td>
<td>4973</td>
<td>1327</td>
<td>2225</td>
<td>459.5</td>
<td>82</td>
</tr>
<tr>
<td>bitea1</td>
<td>471</td>
<td>2410</td>
<td>3289</td>
<td>4801</td>
<td>3545</td>
<td>3594</td>
<td>2270</td>
<td>568.9</td>
<td>1766</td>
<td>157.5</td>
</tr>
<tr>
<td>boga1</td>
<td>383.8</td>
<td>1428</td>
<td>2108</td>
<td>7187</td>
<td>5121</td>
<td>5201</td>
<td>3578</td>
<td>1718</td>
<td>1405</td>
<td>272.5</td>
</tr>
<tr>
<td>fighta1</td>
<td>464.5</td>
<td>1417</td>
<td>1744</td>
<td>2650</td>
<td>1967</td>
<td>1342</td>
<td>856</td>
<td>337</td>
<td>2586</td>
<td>72.3</td>
</tr>
<tr>
<td>floata1</td>
<td>578.9</td>
<td>800.7</td>
<td>962.1</td>
<td>3934</td>
<td>5380</td>
<td>4069</td>
<td>3557</td>
<td>1397</td>
<td>821.7</td>
<td>383.8</td>
</tr>
<tr>
<td>gnat1</td>
<td>367.5</td>
<td>1365</td>
<td>1494</td>
<td>1879</td>
<td>1328</td>
<td>1920</td>
<td>891.7</td>
<td>222.4</td>
<td>84</td>
<td>105.2</td>
</tr>
<tr>
<td>sat2</td>
<td>1941</td>
<td>1834</td>
<td>1545</td>
<td>2775</td>
<td>1558</td>
<td>1234</td>
<td>717.6</td>
<td>421</td>
<td>291</td>
<td>131.9</td>
</tr>
<tr>
<td>slighta1</td>
<td>1390</td>
<td>1442</td>
<td>1417</td>
<td>2706</td>
<td>1768</td>
<td>1126</td>
<td>626</td>
<td>179.9</td>
<td>346.8</td>
<td>128.4</td>
</tr>
<tr>
<td>slopea1</td>
<td>969.6</td>
<td>1077</td>
<td>712.8</td>
<td>3071</td>
<td>3477</td>
<td>2254</td>
<td>1671</td>
<td>1114</td>
<td>898</td>
<td>529</td>
</tr>
</tbody>
</table>

Table 5.3 The average errors of ten decomposed levels using predictor 2

<table>
<thead>
<tr>
<th>File Name</th>
<th>-1</th>
<th>-2</th>
<th>-3</th>
<th>-4</th>
<th>-5</th>
<th>-6</th>
<th>-7</th>
<th>-8</th>
<th>-9</th>
<th>-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>baita1</td>
<td>1904</td>
<td>3470</td>
<td>2617</td>
<td>3349</td>
<td>4788</td>
<td>3802</td>
<td>2741</td>
<td>1299</td>
<td>582.5</td>
<td>199.3</td>
</tr>
<tr>
<td>beeta1</td>
<td>1413</td>
<td>2659</td>
<td>1589</td>
<td>2465</td>
<td>3093</td>
<td>3415</td>
<td>2308</td>
<td>2613</td>
<td>1054</td>
<td>757.8</td>
</tr>
<tr>
<td>bitea1</td>
<td>1874</td>
<td>2940</td>
<td>3889</td>
<td>4758</td>
<td>3601</td>
<td>3575</td>
<td>2276</td>
<td>1292</td>
<td>1701</td>
<td>341.8</td>
</tr>
<tr>
<td>boga1</td>
<td>1522</td>
<td>2204</td>
<td>3041</td>
<td>6911</td>
<td>5117</td>
<td>5202</td>
<td>3374</td>
<td>2111</td>
<td>2190</td>
<td>706.1</td>
</tr>
<tr>
<td>fighta1</td>
<td>1184</td>
<td>1826</td>
<td>1977</td>
<td>2497</td>
<td>2020</td>
<td>1259</td>
<td>815.5</td>
<td>363.3</td>
<td>232.2</td>
<td>83.71</td>
</tr>
<tr>
<td>floata1</td>
<td>1044</td>
<td>1306</td>
<td>2301</td>
<td>4243</td>
<td>5364</td>
<td>4135</td>
<td>3520</td>
<td>1473</td>
<td>825.3</td>
<td>614.9</td>
</tr>
<tr>
<td>gnat1</td>
<td>947.5</td>
<td>1483</td>
<td>1655</td>
<td>1567</td>
<td>1420</td>
<td>1687</td>
<td>634</td>
<td>213.3</td>
<td>113.1</td>
<td>104.5</td>
</tr>
<tr>
<td>sat2</td>
<td>2266</td>
<td>1969</td>
<td>2019</td>
<td>2282</td>
<td>1592</td>
<td>1254</td>
<td>684.7</td>
<td>440.9</td>
<td>286.7</td>
<td>151.8</td>
</tr>
<tr>
<td>slighta1</td>
<td>1783</td>
<td>1695</td>
<td>1833</td>
<td>2667</td>
<td>1767</td>
<td>879.7</td>
<td>630.9</td>
<td>206.4</td>
<td>347.3</td>
<td>162.3</td>
</tr>
<tr>
<td>slopea1</td>
<td>1409</td>
<td>1256</td>
<td>1907</td>
<td>3237</td>
<td>3389</td>
<td>1928</td>
<td>1792</td>
<td>1104</td>
<td>955</td>
<td>562.4</td>
</tr>
</tbody>
</table>

Comparing the results of table 5.2 with those of table 5.3, we found that in most cases the minimum average errors corresponding to predictor 1 are smaller than those from predictor 2. That means the predictor 1 explores the deep insight of correlation of the data in our case. It is reasonable because, derived from equation (5-1), we can
get $W_i^{j-1} = 2(S_i^{j-1} - S_i^j)$, and to predict $W_i^{j-1}$ is equivalent to predict $S_i^j$, then we can use $S_i^{j-1}$ and $S_i^{j-1}_{2(i-1)}$ to predict $S_i^j$, finally, one easy way of using $S_i^{j-1}_{2i-1}$ and $S_i^{j-1}_{2(i-1)}$ is to take an average of them, $S_i^{j-1} = (S_i^{j-1}_{2i-1} + S_i^{j-1}_{2(i-1)})/2$. That is what predictor 1 did. So the formula 1 is chosen as the predictor in our compression system.

Table 5.4 The results of $a, b, c$ of predictor 1 for ten decomposed level

<table>
<thead>
<tr>
<th>File Name</th>
<th>-1</th>
<th>-2</th>
<th>-3</th>
<th>-4</th>
<th>-5</th>
<th>-6</th>
<th>-7</th>
<th>-8</th>
<th>-9</th>
<th>-10</th>
</tr>
</thead>
<tbody>
<tr>
<td>bait_1</td>
<td>-0.63</td>
<td>-0.25</td>
<td>-0.19</td>
<td>-0.56</td>
<td>-0.25</td>
<td>-0.63</td>
<td>1.00</td>
<td>0.94</td>
<td>-0.13</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>-0.25</td>
<td>-0.13</td>
<td>-0.38</td>
<td>-0.25</td>
<td>-0.25</td>
<td>-0.81</td>
<td>-0.19</td>
<td>0.50</td>
<td>-1.00</td>
<td>0.31</td>
</tr>
<tr>
<td></td>
<td>-0.75</td>
<td>-0.13</td>
<td>0.31</td>
<td>0.31</td>
<td>0.25</td>
<td>0.13</td>
<td>0.13</td>
<td>-0.06</td>
<td>0.88</td>
<td></td>
</tr>
<tr>
<td>beet_1</td>
<td>-0.63</td>
<td>-0.31</td>
<td>-0.31</td>
<td>-0.56</td>
<td>-0.44</td>
<td>0.63</td>
<td>0.81</td>
<td>0.81</td>
<td>-0.94</td>
<td>0.38</td>
</tr>
<tr>
<td></td>
<td>-0.31</td>
<td>0.25</td>
<td>-0.25</td>
<td>-0.19</td>
<td>-0.38</td>
<td>0.63</td>
<td>0.00</td>
<td>0.38</td>
<td>0.94</td>
<td>-1.00</td>
</tr>
<tr>
<td></td>
<td>-0.75</td>
<td>0.38</td>
<td>-0.13</td>
<td>-0.50</td>
<td>-0.63</td>
<td>0.56</td>
<td>1.00</td>
<td>0.13</td>
<td>0.63</td>
<td>0.50</td>
</tr>
<tr>
<td>bite_1</td>
<td>-0.63</td>
<td>-0.38</td>
<td>-0.44</td>
<td>-0.25</td>
<td>-0.25</td>
<td>-0.25</td>
<td>-0.50</td>
<td>0.94</td>
<td>0.56</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>-0.19</td>
<td>-0.25</td>
<td>-0.31</td>
<td>-0.06</td>
<td>-0.06</td>
<td>0.0</td>
<td>-0.63</td>
<td>1.00</td>
<td>1.00</td>
<td>-1.00</td>
</tr>
<tr>
<td></td>
<td>-0.63</td>
<td>-0.25</td>
<td>-0.25</td>
<td>-0.06</td>
<td>0.13</td>
<td>0.19</td>
<td>0.56</td>
<td>0.81</td>
<td>0.0</td>
<td>0.69</td>
</tr>
<tr>
<td>bog_1</td>
<td>-0.63</td>
<td>-0.50</td>
<td>-0.50</td>
<td>-0.06</td>
<td>0.19</td>
<td>0.06</td>
<td>0.25</td>
<td>0.44</td>
<td>1.00</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>-0.19</td>
<td>-0.25</td>
<td>90.25</td>
<td>-0.06</td>
<td>0.0</td>
<td>0.06</td>
<td>-0.06</td>
<td>0.69</td>
<td>-1.00</td>
<td>-1.00</td>
</tr>
<tr>
<td></td>
<td>-0.63</td>
<td>-0.31</td>
<td>-0.63</td>
<td>-0.13</td>
<td>0.13</td>
<td>0.13</td>
<td>0.38</td>
<td>-0.69</td>
<td>-0.25</td>
<td>-0.19</td>
</tr>
<tr>
<td>fight_1</td>
<td>-0.56</td>
<td>-0.38</td>
<td>-0.50</td>
<td>0.44</td>
<td>-0.31</td>
<td>-0.06</td>
<td>-1.00</td>
<td>-0.50</td>
<td>1.00</td>
<td>0.69</td>
</tr>
<tr>
<td></td>
<td>-0.25</td>
<td>-0.25</td>
<td>-0.31</td>
<td>0.25</td>
<td>-0.25</td>
<td>-0.44</td>
<td>-0.13</td>
<td>0.25</td>
<td>-0.38</td>
<td>0.94</td>
</tr>
<tr>
<td></td>
<td>-0.56</td>
<td>-0.13</td>
<td>-0.38</td>
<td>-0.31</td>
<td>0.0</td>
<td>0.13</td>
<td>0.88</td>
<td>0.81</td>
<td>0.19</td>
<td>0.56</td>
</tr>
<tr>
<td>float_1</td>
<td>-0.5</td>
<td>-0.25</td>
<td>-0.63</td>
<td>-0.38</td>
<td>-0.06</td>
<td>-0.19</td>
<td>0.38</td>
<td>0.31</td>
<td>-0.13</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>-0.25</td>
<td>-0.31</td>
<td>-0.19</td>
<td>-0.38</td>
<td>-0.25</td>
<td>0.06</td>
<td>0.69</td>
<td>0.50</td>
<td>0.31</td>
<td>-0.31</td>
</tr>
<tr>
<td></td>
<td>-0.50</td>
<td>0.0</td>
<td>-0.63</td>
<td>-0.44</td>
<td>0.25</td>
<td>0.06</td>
<td>0.25</td>
<td>-0.06</td>
<td>-0.06</td>
<td>0.44</td>
</tr>
<tr>
<td>gnat_1</td>
<td>-0.63</td>
<td>-0.38</td>
<td>-0.31</td>
<td>-0.25</td>
<td>-0.38</td>
<td>-0.94</td>
<td>1.00</td>
<td>-0.69</td>
<td>1.00</td>
<td>0.63</td>
</tr>
<tr>
<td></td>
<td>-0.19</td>
<td>-0.25</td>
<td>-0.31</td>
<td>-0.19</td>
<td>-0.25</td>
<td>-1.00</td>
<td>1.00</td>
<td>-0.75</td>
<td>1.00</td>
<td>0.10</td>
</tr>
<tr>
<td></td>
<td>-0.63</td>
<td>-0.38</td>
<td>0.0</td>
<td>-0.13</td>
<td>-0.44</td>
<td>-0.06</td>
<td>0.18</td>
<td>-0.19</td>
<td>-0.38</td>
<td>0.56</td>
</tr>
<tr>
<td>sat_2</td>
<td>-0.44</td>
<td>-0.38</td>
<td>-0.56</td>
<td>0.56</td>
<td>-0.13</td>
<td>-0.38</td>
<td>-0.44</td>
<td>-0.13</td>
<td>0.19</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>-0.25</td>
<td>-0.25</td>
<td>-0.44</td>
<td>0.31</td>
<td>-0.19</td>
<td>-0.50</td>
<td>-0.13</td>
<td>-0.69</td>
<td>0.19</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>-0.38</td>
<td>-0.19</td>
<td>-0.19</td>
<td>-0.13</td>
<td>0.0</td>
<td>0.75</td>
<td>-0.13</td>
<td>0.06</td>
<td>0.25</td>
<td>1.00</td>
</tr>
<tr>
<td>slight_1</td>
<td>-0.44</td>
<td>-0.38</td>
<td>-0.56</td>
<td>0.56</td>
<td>-0.19</td>
<td>-0.13</td>
<td>0.56</td>
<td>-0.25</td>
<td>-0.13</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>-0.31</td>
<td>-0.31</td>
<td>-0.25</td>
<td>0.38</td>
<td>0.0</td>
<td>-0.19</td>
<td>0.38</td>
<td>0.0</td>
<td>-0.13</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>-0.44</td>
<td>-0.13</td>
<td>-0.36</td>
<td>-0.25</td>
<td>0.13</td>
<td>0.0</td>
<td>0.69</td>
<td>-0.13</td>
<td>-0.13</td>
<td>0.25</td>
</tr>
<tr>
<td>slope_1</td>
<td>-0.50</td>
<td>-0.06</td>
<td>-0.63</td>
<td>-0.38</td>
<td>-0.19</td>
<td>-0.19</td>
<td>-0.75</td>
<td>-0.31</td>
<td>0.38</td>
<td>0.69</td>
</tr>
<tr>
<td></td>
<td>-0.31</td>
<td>-0.38</td>
<td>-0.19</td>
<td>-0.19</td>
<td>-0.06</td>
<td>0.0</td>
<td>0.0</td>
<td>0.94</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-0.63</td>
<td>0.38</td>
<td>-0.63</td>
<td>-0.50</td>
<td>0.38</td>
<td>0.19</td>
<td>0.81</td>
<td>0.19</td>
<td>0.25</td>
<td>0.31</td>
</tr>
<tr>
<td>average</td>
<td>-0.56</td>
<td>-0.33</td>
<td>-0.46</td>
<td>-0.09</td>
<td>-0.20</td>
<td>-0.21</td>
<td>0.13</td>
<td>0.16</td>
<td>0.28</td>
<td>0.83</td>
</tr>
<tr>
<td></td>
<td>-0.25</td>
<td>-0.21</td>
<td>-0.29</td>
<td>-0.04</td>
<td>-0.18</td>
<td>-0.23</td>
<td>0.09</td>
<td>0.19</td>
<td>0.19</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>-0.59</td>
<td>-0.08</td>
<td>-0.31</td>
<td>-0.31</td>
<td>0.01</td>
<td>0.14</td>
<td>0.63</td>
<td>0.09</td>
<td>0.09</td>
<td>0.43</td>
</tr>
</tbody>
</table>

In table 5.4, the parameters $a, b, c$ associated with the minimum average errors in
every decomposed level for ten speech file are listed. For every file, the values of the first row, second row and third row are those of parameter $a, b, c$, respectively. Because of each test file just pronouncing only one word, we take average of parameters $a, b, c$ over the ten files to fit more common situations.

5.1.3 Stage 3: Rice coding

Once removing all redundancy in samples is achieved, next step is to encode. One of the most efficient ways to do this is with Rice coding. Rice coding is a specialized form of Golomb coding. It makes most sense when most of the numbers to be coded are relative small. This suits the "exponential" or "laplacian" distribution of numbers which typically make up the difference of a lossless audio encoding algorithm. In other words, it is used to encode string of numbers with a variable bit length for each number, in which the distribution is stronger for low values than for high. That means, if most of the number are small, fairly good compression can be achieved.

A little more technical and mathematical description of the process of Rice coding are shown in Rice Coding Algorithm below and this is what we use in our compression system.

---

Rice Coding Algorithm

/* Use a Rice code to represent symbol $x$, and $k$ is the parameter of the Rice code */

VARIABLE int i, n, q, everySize, totalSize;

Rice_Encode($x, k$)

1. /* Change the representation of signed integers */
MOVE integer to right One bit;

IF integer > 0 THEN put "0" bit to the most right position;

IF integer < 0 THEN put "1" bit to the most right position;

2. /* Find a optimal parameter $k$ (the most right bits of the codeword) */

FOR $k$ ← 0 to $n$ DO

FOR every codeword $W$ DO

q ← $W/2^k$;

eyeverySize ← q+1+k;

totalSize ← totalSize+everySize;

RETURN $k$ corresponding the smallest totalSize;

3. /* Sent the parameter $k$ */

SendBit( every bits of parameter $k$ );

4. /* For every codeword, send $W/2^k$ 1’s, then terminate 0 to tell that

you done sending the "overflow", then send the most right $k$ bits

of the codeword */

FOR every codeword $W$ DO

FOR i ← 0 to q-1 DO SendBit(1);

SendBit(0);

SendBit( the most right $k$ bits of the codeword );

$\text{RiceDecode}(x, k)$

1. /* get the parameter $k$ */

GetBit( every bits of parameter $k$ );

2. /* Reconstruct every codeword */

WHILE( all bits in a compressed file )

\text{WHILE} ( \text{GetBit(1)} ) q ← q+1;
\[ x \leftarrow 2^k \ast q; \]

\textbf{FOR} every \( k \) bits \textbf{DO}

\[ x \leftarrow x \mid \text{GetBit()} \ll a \text{ bit}; \]

Let's turn back to the test and analyses of our compression system. After computing the entropy of the error data, we found that there at least needs average 8 bits to represent an error value (shown in table 5.5). That means a file generally cannot be compressed beyond half of original file by using our compression algorithm.

<table>
<thead>
<tr>
<th>File Name</th>
<th>Average Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>baita1</td>
<td>7.12</td>
</tr>
<tr>
<td>beeta1</td>
<td>7.08</td>
</tr>
<tr>
<td>bitea1</td>
<td>6.78</td>
</tr>
<tr>
<td>boga1</td>
<td>6.77</td>
</tr>
<tr>
<td>fighta1</td>
<td>7.07</td>
</tr>
<tr>
<td>floata1</td>
<td>7.08</td>
</tr>
<tr>
<td>gnat1</td>
<td>6.86</td>
</tr>
<tr>
<td>sat2</td>
<td>7.38</td>
</tr>
<tr>
<td>slighta1</td>
<td>7.31</td>
</tr>
<tr>
<td>slopea1</td>
<td>7.26</td>
</tr>
</tbody>
</table>

Furthermore, after operating our compression algorithm on the ten test files, we got the compression ratio being around 0.82 (result given in table 5.6).

5.2 Conclusion

Using a new compression algorithm described in this research, a lossless compression is done through the processes of Harr wavelet transform, prediction and

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Table 5.6 The compression ratio for ten test files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Original Bytes</th>
<th>Compressed Bytes</th>
<th>Compression Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>baita1</td>
<td>7696</td>
<td>6544</td>
<td>0.85</td>
</tr>
<tr>
<td>beeta1</td>
<td>7224</td>
<td>6028</td>
<td>0.83</td>
</tr>
<tr>
<td>bitea1</td>
<td>11176</td>
<td>9392</td>
<td>0.84</td>
</tr>
<tr>
<td>boga1</td>
<td>14604</td>
<td>12156</td>
<td>0.83</td>
</tr>
<tr>
<td>fighta1</td>
<td>14234</td>
<td>11356</td>
<td>0.80</td>
</tr>
<tr>
<td>floata1</td>
<td>15394</td>
<td>12332</td>
<td>0.80</td>
</tr>
<tr>
<td>gnata1</td>
<td>12284</td>
<td>9612</td>
<td>0.78</td>
</tr>
<tr>
<td>sat2</td>
<td>15606</td>
<td>13208</td>
<td>0.85</td>
</tr>
<tr>
<td>slighta1</td>
<td>16606</td>
<td>13676</td>
<td>0.82</td>
</tr>
<tr>
<td>slopea1</td>
<td>17188</td>
<td>13908</td>
<td>0.81</td>
</tr>
</tbody>
</table>

Rice coding. The main difference between our algorithm and other common lossless compression methods is the combination of Harr wavelet transform and prediction together as a whole modeling. It takes the advantage of wavelet transform, such as functionality of subband filter so the data in the same band have more correlation between each other, and fast, efficient computing speed. Meanwhile, it keeps the benefit of prediction as well, which can eliminate the redundancy in data. Theoretically, this algorithm could reach a pretty good compression ratio. However, comparing with the results of several other lossless compression softwares (a report given by [Robin Whittle, 2002] states that 60 to 70% of original file-size with pop, rock, techno and other loud, noisy music and 35 to 60% for quieter choral and orchestral pieces can be achieved with lossless compression ), the performance of our algorithm do not reach what we expect. It may be caused by using different kinds of test files, but there are still more rooms for improving our algorithm. One direction is to modify the predictor we used. It is well known that most good predictors are adaptive because they are able to match the varying magnitudes of the input samples. So one of the
steps we need to do next is to design a adaptive predictor and test its performance. Another way to get better lossless compression in our case is to use and test more efficient alternatives to Rice coding, such as Elias Gamma and other codes which are highly efficient at coding large integers and the “cost” of having longer “overflow” part is much less. Standard Rice coding of “overflow” parts of values more than about 3 bits long can clearly be improved upon with Biased Elias Gamma. Clearly, the results shown in table 5-2 indicate that it is not the case in which most of the numbers to be encoded are small, which is the situation handled best by Rice coding [Robin Whittle, 2002; Peter Fenwick, 1996]. Through taking these improvements, we think the compression performance of our algorithm will be better.
REFERENCES


VITA

Graduate College
University of Nevada, Las Vegas

Yang Liu

Local Address:
3820 Spring Valley Rd., #2004
Addison, TX 75001

Degrees:
Bachelor of Science, Physics, 1987
Shanghai University of Science and Technology, Shanghai, China
Ph.D., Physics, 2000
University of Nevada, Las Vegas, U.S.A

Publication:

Thesis Title:
Lossless Audio Compression of Speech And Voice

Thesis Examination Committee:
Chairperson. Dr. Evangelos A. Yfantis. Ph.D.
Committee Member. Dr. Gewali. Ph.D.
Committee Member. Dr. Bein. Ph.D.
Graduate Faculty Representative. Dr. Wang. Ph.D.