4-2-1998

Adaptive segmenting of non-stationary signals

Christopher Albin Edmonds
Florida International University

DOI: 10.25148/etd.FI15101266

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

ADAPTIVE SEGMENTING OF NON-STATIONARY SIGNALS

A thesis submitted in partial satisfaction of the

requirements for the degree of

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

by

Christopher Albin Edmonds

1998

To: Dean Gordon R. Hopkins
College of Engineering

This thesis, written by Christopher Albin Edmonds, and entitled Adaptive Segmenting of Non-stationary Signals, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this thesis and recommend that it be approved.

Armando B. Barreto

Malek Adjouadi

Jean H. Andrian, Major Professor

Date of Defense: April 2, 1998

The thesis of Christopher Albin Edmonds is approved.

Dean Gordon R. Hopkins
College of Engineering

Dr. Richard L. Campbell
Dean of Graduate Studies

Florida International University, 1998

To my mother, father, sister, and brother, who always believed in me.

ACKNOWLEDGMENTS

I would first like to thank the members of my committee, Dr. Armando Barreto, Dr. Malek Adjouadi, and especially my major professor Dr. Jean Andrian, for their assistance and patience. I would also like to thank the members of the Electrical and Computer Engineering faculty, especially Dr. Kang Yen, Dr. Grover Larkins, Dr. Sylvia Mergui, Dr. Subbarao Wunnuva, and Dr. James Story. Special acknowledgment is due to Ms. Pat Brammer, Ms. Marbeth Cochran, Ms. Anne Wood, Ms. Noemi Fernandez, Mr. Mike Urucinitz, Mr. Hamid Ghassemi, and my colleague Mr. Shafiqul Islam.

This work was supported in part by United States Air Force contract number F33615-96-C-1824. In addition, the support provided by the National Science Foundation grant number CDA-9313624 with the Center for Advanced Technology and Education (NSF-CATE) is appreciated, as is that of the Future Aerospace Science and Technology (FAST) Center. I am also indebted to the staff of the FIU Library, in particular Ms. Elena Cruz in the Circulation Department and Ms. Allison Malone in Interlibrary Loan. Finally, let me extend my gratitude to Ms. Tracie Lutchmansingh, Mr. David Douglas, Ms. Lisa Watson, and Ms. Carole Bell, for lending me their emotional support and encouragement throughout my studies.

ABSTRACT OF THE THESIS

ADAPTIVE SEGMENTING OF NON-STATIONARY SIGNALS

by

Christopher Albin Edmonds

Florida International University, 1998

Miami, Florida

Professor Jean H. Andrian, Major Professor

Many data compression techniques rely on the low entropy and/or the large degree of autocorrelation exhibited by stationary signals. In non-stationary signals, however, these characteristics are not constant, resulting in reduced data compression efficiency. An adaptive scheme is developed that divides non-stationary signals into smaller locally stationary segments, thereby improving overall efficiency. Two principal issues arise in implementing this procedure. The first is practical; an exhaustive search of all possible segmentations is in general computationally prohibitive. The concept of dynamic programming is applied to reduce the expense of such a search. The second involves choosing a cost function that is appropriate for a particular compression method. Two cost functions are employed here, one based on entropy and the other on correlation. It is shown that by using an appropriate cost function, an adaptively segmented signal offers better data compression efficiency than an unsegmented or arbitrarily segmented signal.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

1.1 Data Compression and Transform Coding

Data compression is a form of data processing in which an attempt is made to represent a given set of data by another smaller set. It is currently a very popular topic in digital signal processing (and other fields as well), motivated by the need to store ever-increasing amounts of information in limited space, and similarly, to transmit this information over limited bandwidth. There are many different types of data sets that may be compressed, this thesis, however, will be concerned in particular with sets of discretely-valued ordinal data, hereafter referred to as signals; likewise, although many techniques exist for attempting data compression, this thesis will concentrate on the class of techniques known as transform coding.

Transform coding attempts to compress a signal by mapping it from one vector space into another by means of an orthogonal transform, a simple example being the transformation from the discrete-time domain to the discrete-frequency domain using the discrete Fourier transform. The transformation process does not in itself achieve data compression, however, the use of an appropriate transform may yield coefficients in the new domain that may be more efficiently represented than the original signal. The explanation for the this lies in the fact that if the signal is stationary, and thus highly self-correlated, the appropriate transform will produce highly decorrelated coefficients. It will be shown in Chapter 2 how this decorrelation may be exploited for improved data compression performance.

In practice, however, many signals that need to be compressed are not stationary-two common examples of this being speech and music. As a result, it may not be possible to compress these signals as efficiently as required. To help compensate for this, many data compression algorithms subdivide, or segment, the signal into smaller non-overlapping signals, or segments. Although this segmenting is partially motivated by computational constraints, it is hoped that this process will reduce the negative effects of the non-stationarity of these signals by constraining the amount of variation within a given segment. Unfortunately, this segmenting process is usually uniform and arbitrary with respect to a particular signal, thus it may or may not be beneficial.

1.2 Overview of Proposed Methodology

In response to this, this thesis proposes a scheme for adaptively segmenting non-stationary signals. This scheme will segment these signals into variably-sized segments that are locally stationary, i.e. they are stationary within their local support. It is hypothesized then that improved data compression performance may be achieved by adaptively rather than uniformly segmenting signals. This scheme is proposed as a flexible framework that may be easily customized for and integrated into existing data compression algorithms.

Implementation of the adaptive segmenting scheme may be divided into two areas. A set of segment boundaries on a particular signal will be referred to as a segmentation, and the first area is concerned with identifying different possible segmentations This process will be referred to as the search mechanism. The second area is concerned with how to choose the best of these segmentations, this will be done by applying appropriate cost functions to each segmentation and comparing the results.

The overall procedure can be illustrated using a simple example. Let $J(x)$ be a function describing the cost of a particular signal x and let $K(J_0, J_1, \ldots)$ be a function describing the total effective cost of a series of cost functions $\{J_0, J_1, \ldots\}$. Given a signal $x_{012} = \{x_0, x_1, x_2\}$, it can be seen that x can generate a maximum of four segmentations, shown as follows:

$$\begin{aligned} x_{012} &= \{x_0, x_1, x_2\} \\ x_{01}x_2 &= \{x_{01}\}\{x_2\} \\ x_0x_{12} &= \{x_0\}\{x_{12}\} \\ x_0x_1x_2 &= \{x_0\}\{x_1\}\{x_2\} \end{aligned} \tag{1.1}$$

The adaptive segmenting scheme would then calculate the cost of each segment within each segmentation, and then calculate the total effective cost of each segmentation, as illustrated next:

$$\begin{array}{l} K\left(J\left(x_{012}\right)\right) \\ K\left(J\left(x_{01}\right), J\left(x_2\right)\right) \\ K\left(J\left(x_0\right), J\left(x_{12}\right)\right) \\ K\left(J\left(x_0\right), J\left(x_1\right), J\left(x_2\right)\right) \end{array} \qquad (1.2)$$

The minimum value of K is then identified, indicating the best segmentation in terms of the cost function J.

An additional point that must be considered is the side information that accompanies this procedure. Once the data has been adaptively segmented some record of the particular segmentation must be maintained, since a later part of the process cannot assume a known uniform segmentation. While this side information does reduce the overall efficiency of the method, it becomes evident that this additional cost is quite negligible in practice. This is because in general the only extra information required is the length of the next segment.

1.3 Existing Work

This idea of generating non-uniform segmentations has been explored by a number of authors and also exists, in a limited sense, in at least one commercial data compression algorithm. Lee, Kim, and Lee [1], Sinha and Johnston [2], Watkinson [3], Brooks, et al. [4], and others have proposed different methods for intelligently dividing signals into segments of varying lengths. All of these methods, however, have been very application specific, as well as relatively inflexible in their approach. Of special interest though is the work of Xiong, et al. [5], who propose, in the specific context of wavelet packets, a method of reducing the cost of an exhaustive search of all possible segmentations by employing techniques borrowed from dynamic programming theory. This method will be explored in detail in Chapter 4.

Local trigonometric bases as described by Coifman, et al. [6] and others, along with their close relatives, the lapped orthogonal transforms of Malvar [7], can also be considered as forms of adaptive segmenting. These techniques work by splitting a signal, the parent, into two equally sized segments, the children, and then comparing the cost of the parent's transform, based on the sine/cosine family of bases, with the combined cost of the transforms of the children. If the children's cost is the lower of the two, then the children become parents and the process is repeated again with their children. If the parent is the winner at any stage, then further decomposition along that branch ceases. Figure 1.1 illustrates a possible segmentation of a signal of length 8, and Figure 1.2 the resulting binary tree that this process generates.

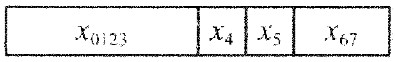

Figure 1.1 A possible segmentation

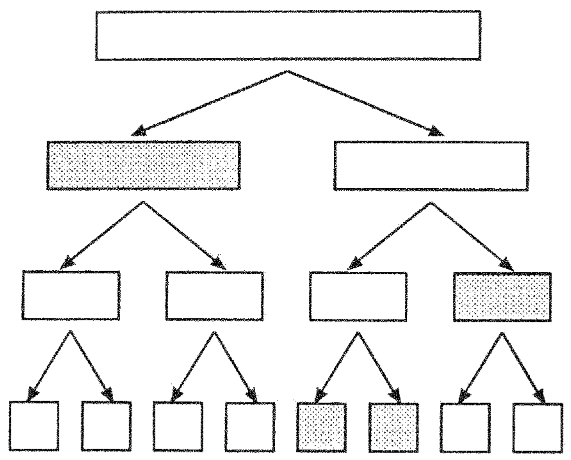

Figure 1.2 A binary tree leading to the segmentation of Figure 1.1

Although this method may generate segmentations that are better than uniform ones, it suffers from several significant drawbacks. Because of its binary nature, it is applicable only to signals of dyadic length, and it is also inflexible in the sense that this binary division only can generate a subset of the total possible segmentations of a given signal. In addition, it is not shift-invariant. Finally, when the decision process halts the decomposition at a particular level, i.e. when the parent is the winner, further segmentations are not considered below the current children, despite the fact that better segmentations may exist in lower levels of that particular branch.

The ATRAC (Adaptive Transform Acoustic Coding) algorithm is a proprietary data compression algorithm used by MiniDisc-based audio systems. It transforms segments of data using a modified form of the discrete cosine transform, and then codes the transform coefficients using a psychoacoustic/perceptual model. Of particular interest, though, is the segmentation technique it uses. The algorithm analyses the incoming signal, and alters the segment length between 1.45 ms, 2.9 ms, and 11.6 ms (corresponding to 64, 128, and 512 samples at the sampling rate of 44.1 kHz), based on the transient information present in the signal. [8]

1.4 Overview of Remaining Chapters

The remainder of this thesis is divided as follows: Chapter 2 explores data compression and transform coding in more detail and details their relationship to the concepts of stationarity, correlation, and entropy. Chapter 3 gives a description of the various cost functions based on these concepts. Chapter 4 then gives an explanation and discussion of the various search mechanisms employed.. Chapter 5 provides examples of the performance of the adaptive segmenting method, and compares them with existing methods. Finally, Chapter 6 offers conclusions and recommendations for further work.

Chapter 2: Data Compression and Transform Coding

2.1 Data Compression

The term data compression refers to the process of mapping a set of discrete data of size N to another set of size M, such that $M \leq N$, and such that the information contained in the new set is sufficient to reconstruct the original set if required. The data sets of interest here will be discretely valued and ordinal, representing some sampled one-dimensional signal such as speech or music. This reconstruction and the associated mapping scheme are usually categorized as either lossy or lossless. Lossy compression implies that the reconstruction of the original set may not be exact, but is instead sufficient for some purpose. Lossless compression implies that the reconstruction is identical to the original set.

More explicitly, let $x[n]$ be a finite-energy discretely valued sequence of length N. The process of data compression of $x[n]$ is then a mapping of $x[n]$ into another sequence $y[m]$ of length M, such that $M \leq N$. It must then be possible to generate a reconstruction $\hat{x}[n]$ of $x[n]$ from $y[m]$ alone. The reconstruction error $r[n]$ is defined as:

$$r[n] = x[n] - \hat{x}[n] \qquad \textbf{(2.1)}$$

The goal of lossy compression is to minimize M and $|r[n]|$ simultaneously, usually within some constraint on the magnitude of $r[n]$. For lossless compression, $r[n]=0$ for all n, so the goal is simply to minimize M. Note that in practice, some error may be introduced even in lossless schemes by finite word length effects and other processing-related artifacts, but this error is inescapable and so will not be considered.

Before continuing, it is also useful to define one of several measures of performance. The compression ratio R is:

$$R=\frac{N}{M}\text{ samples/sample} \qquad \textbf{(2.2)}$$

R is a measure of the efficiency of the compression scheme; a larger compression ratio implies more efficient compression. R has the following bounds:

$$1\leq R\leq N \qquad \textbf{(2.3)}$$

The lower bound implies the worse-case scenario, that is no compression has taken place. A value of less than one implies data expansion; any algorithm should of course recognize this occurrence and react to it appropriately. The upper bound implies that $M=1$, that is optimal compression has been achieved.

Often, the concern is with the lengths of $x[n]$ and $y[m]$ in terms of their base two representations, thus let R_B be the binary compression ratio:

$$R_B = \frac{N_B}{M_B} \text{ bits/bit} \tag{2.4}$$

where N_B and M_B are the lengths of the base two representations of $x[n]$ and $y[m]$, respectively. Note that R will not necessarily be equal to R_B.

2.2 Transform Coding

Data compression schemes can be divided into a number of general categories, or coding schemes, based on the approach that they take to attempt data compression. These include predictive coding, sub-band coding, arithmetic and run-length coding, direct entropy coding, and so on. Many schemes may actually employ more than one of these techniques; to some degree, however, all of these attempt to exploit some form of redundancy or predictability that may (or may not) be present in the given data. Of particular interest is the category of transform coding.

Data compression schemes that utilize transform coding work by transforming a signal from one vector space into another, and then processing the resulting transform coefficients. By choosing an orthogonal transform, these coefficients may then be used to reconstruct the original signal with a minimum of effort, although orthogonality is not necessarily a requirement. As was mentioned in Chapter 1, the process of transformation does not in itself yield any data compression. Compression arises instead from the processing of the transformed coefficients, on the premise that the transform operation has yielded a sequence that may somehow be represented more efficiently. This efficiency relies on two factors, the first is which transform is chosen, and the second is what type of processing is applied to the resulting coefficients.

A wide variety of transforms exist to choose from, and all are application and/or signal specific to varying degrees; in other words, there is no one transform scheme that universally achieves optimal compression of all data. The discrete-time Karhunen-Loeve transform [9] (KLT), also known as the Hotelling transform or principal component transform, can be used to form an exception to this specificity, however, it will be shown that this is only a theoretical advantage. In general, though, a particular transform is usually chosen based on its ability to statistically decorrelate a class of signals.

2.3 Stationarity and Correlation

A stationary signal is defined as a signal that's statistical behavior does not vary over time; in particular, it has a constant mean μ_x, a constant variance σ_x^2, and an autocorrelation function R_{xx} and autocovariance function C_{xx} that vary only with the difference or lag k. In other words, if $E[\cdot]$ is the expected value operator, and if R_{xx} and C_{xx} are defined as follows,

$$R_{xx}[n,m] = E\big[x[n]x[m]\big] \qquad \textbf{(2.5)}$$

$$C_{xx}[n,m] = E[(x[n]-\mu_x)(x[m]-\mu_x)] \quad \textbf{(2.6)}$$

then for a stationary signal,

$$R_{xx}[n,n+k] = E[x[n]x[n+k]] = R_{xx}[k] \quad \textbf{(2.7)}$$

$$C_{xx}[n,n+k] = E[(x[n]-\mu_x)(x[n+k]-\mu_x)] = C_{xx}[k] \quad \textbf{(2.8)}$$

That is to say, R_{xx} and C_{xx} depend only on k, and not on the absolute location n which the autocorrelation or autocovariance is measured from. Note that for a zero-mean signal, $R_{xx} = C_{xx}$, and this condition will be assumed in all further discussion.

Considering some signal $x[n]$ as a column vector $\vec{\mathbf{x}}$, the autocovariance matrix $\mathbf{C}_{xx}$ can be defined as:

$$\mathbf{C}_{xx} = E[\vec{\mathbf{x}}\vec{\mathbf{x}}^T] \quad \textbf{(2.9)}$$

It can be seen that when $\vec{\mathbf{x}}$ represents a stationary signal, $\mathbf{C}_{xx}$ is symmetric and has Toeplitz form, in which all the values along each diagonal are equal, as shown here:

$$\mathbf{C}_{xx} = \begin{bmatrix} C_{xx}[0] & C_{xx}[1] & C_{xx}[2] & \cdots & C_{xx}[N-1] \\ C_{xx}[1] & C_{xx}[0] & C_{xx}[1] & \cdots & C_{xx}[N-2] \\ C_{xx}[2] & C_{xx}[1] & C_{xx}[0] & \cdots & C_{xx}[N-3] \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ C_{xx}[N-1] & C_{xx}[N-2] & C_{xx}[N-3] & \cdots & C_{xx}[0] \end{bmatrix} \quad \textbf{(2.10)}$$

The elements along the main diagonal of $\mathbf{C}_{xx}$ indicate each sample's correlation with itself, while the off-diagonal elements indicate the degree of correlation between different samples.

In the context of transform coding, the statistics of the signal's transform must also be considered. If $\mathbf{T}$ is an $N \times N$ matrix representing some linear transform, then let $\vec{\mathbf{X}}$ be a column vector of the transform coefficients of $\vec{\mathbf{x}}$, that is:

$$\vec{\mathbf{X}} = \mathbf{T}\vec{\mathbf{x}} \quad \textbf{(2.11)}$$

The autocovariance matrix $\mathbf{C}_{XX}$ of the transform coefficients is then:

$$\mathbf{C}_{XX} = E\left[\vec{\mathbf{X}}\vec{\mathbf{X}}^{T}\right] = \mathbf{T}\mathbf{C}_{xx}\mathbf{T}^{T} \quad \textbf{(2.12)}$$

The elements of the main diagonal of $\mathbf{C}_{XX}$ represent the variances of the of the individual transform coefficients. The off-diagonal elements represent any cross-correlation between different elements of $\vec{\mathbf{x}}$ that has not been removed by the transform; thus for an optimal transform:

$$\mathbf{C}_{XY} = \mathrm{diag}\left[\mathbf{C}_{XY}\right] \tag{2.13}$$

where $\mathrm{diag}[\cdot]$ is a diagonal matrix constructed from the main diagonal of $[\cdot]$. This phenomenon is referred to as the diagonalization of the autocovariance matrix.

The aforementioned KLT is the only linear orthogonal transform that can achieve the condition in Equation 2.13 for any signal $\vec{\mathbf{x}}$, however, the basis vectors of $\mathbf{T}$ in this case are the eigenvectors of the input signal's autocovariance matrix $\mathbf{C}_{xx}$. The consequence of this is that the $\mathbf{T}$ must be recalculated for every different $\vec{\mathbf{x}}$, and additionally, $\mathbf{T}$, $\mathbf{C}_{xx}$, or $\vec{\mathbf{x}}$ must be known to reconstruct $\hat{\vec{\mathbf{x}}}$. Saving or transmitting $\mathbf{T}$, $\mathbf{C}_{xx}$, or $\vec{\mathbf{x}}$ along with the compressed signal $\vec{\mathbf{y}}$ obviously defeats the purpose of data compression. Rao and Yip [10], as well as others, have shown that the discrete cosine transform (DCT) family of transforms can approach the decorrelation performance of the KLT for many classes of stationary signals, and thus the DCT will be used in many of the algorithms presented later. The DCT also has the advantage of being calculable using a fast method similar to that used by the fast Fourier transform.

2.4 Thresholding and Quantizing

Once a signal has been transformed into a new domain using an appropriate transform, the processing of the transform coefficients to achieve data compression still remains. For lossy compression, two main types of processing exist, both of which may be used exclusively or in tandem. The first of these is quantization, in which the resulting transform coefficients are mapped into some discrete alphabet of values. The simplest form of this is scalar quantization, in which each individual coefficient is uniformly mapped into a discretely valued alphabet. In reality, this process occurs by default, since the transform coefficient will already be represented by a fixed- or floating-point binary word. This form of quantization may still be applied, however, to constrain floating-point coefficients to fixed-point form, or to reduce the resolution of fixed-point words. Uniform scalar quantization does not yield any data compression unless the binary resolution is reduced; this is an example of a case where $R = 1$ but $R_B > R$.

More advanced methods include non-uniform quantization and vector quantization. In non-uniform quantization, the coefficients are mapped to a reduced resolution binary alphabet, however, the ranges of values corresponding to a particular word in the new alphabet are not uniformly sized. Instead, some ranges of values are quantized more finely than others, with this variation defined by the application. Vector quantization works by considering a sequence of adjacent coefficients as coordinates of a higher-dimensional vector. A predefined codebook of vectors is then searched for the nearest match, and an index value representing this match is what is stored or transmitted.

There are a very large number of vector quantization techniques, many of them are discussed in [9] and [11].

The second type of processing is thresholding, in which only a subset of the total set of transform coefficients are retained for storage or transmittal, and the rest are set to zero and discarded. A number of types of thresholding exist. Hard or absolute thresholding discards any coefficient less than some fixed threshold value τ, such that:

$$\hat{X}[n] = \begin{cases} X[n] & |X[n]| \geq \tau \\ 0 & |X[n]| < \tau \end{cases} \tag{2.14}$$

Quantile thresholding [12] retains a fixed percentage of coefficients, such that:

$$\hat{X}[n] = \begin{cases} X[n] & |X[n]| \geq p \\ 0 & |X[n]| < p \end{cases} \tag{2.15}$$

where p is a p-quantile of $X[n]$.

A number of methods also exist to choose τ or p. Relative energy thresholding [6] sets $\tau^2 = \varepsilon\|x\|^2$, where $0 < \varepsilon \leq 1$. p can also be chosen to with the goal of achieving a specific value of M or R. Obviously the class of signal, type of transform, and desired performance will effect the choice of τ or p.

All of the preceding discussion invites the question of how quantizing and thresholding relate to correlation and data compression. In this context, it is important to remember Parseval's Theorem and its relation to linear orthogonal transforms, which says:

$$\sum_{n=0}^{N-1} |x[n]|^2 = \sum_{n=0}^{N-1} |X[n]|^2 \qquad \textbf{(2.16)}$$

In other words, no energy is lost or gained when transforming a signal; it is instead redistributed among the coefficients. An optimal transform for a particular class of signals will redistribute the majority of the energy of the signal into only a few of the transform coefficients; this phenomenon known as energy packing. As a result, quantizing or discarding small-valued coefficients will introduce only a negligible error into the reconstruction $\hat{\mathbf{x}}$. This result can be extended in a more general sense; quantizing or thresholding of coefficients will introduce error into the reconstructed signal $\hat{\mathbf{x}}$, however, the more decorrelated the coefficients are, the more this error will be distributed uniformly throughout $\hat{\mathbf{x}}$. Careful choice of the thresholding or quantizing technique for a particular application, e.g. choosing some psychoacoustic criteria for compression of audio, can ensure this occurs.

2.5 Entropy

For lossless compression, and for post-processing in lossy compression, entropy-based techniques are used. Entropy is a measure of the average information content of a set of data. Given a set of discretely valued data x of length N, assume that each individual element of x is drawn from some alphabet set of I possible values or symbols. For example, in a data set whose elements consisted of B-bit binary words, each element would be drawn from an alphabet set of I words, where $I \leq 2^B$. Let $P_x[i]$ represent the probability of the occurrence of the i th element of the alphabet set in x. The entropy of x is then defined as:

$$H_x = \sum_{i=0}^{I-1} P_x[i] \log_2\left(\frac{1}{P_x[i]}\right) \quad \text{bits/word} \qquad \textbf{(2.17)}$$

Notice that a base two logarithm is used in the definition of H_x since the concern is with an alphabet of binary words. When considering alphabets of different symbol forms, Equation 2.17 must be modified appropriately. When using data compression schemes that exploit the entropy of the data set, the binary compression ratio R_B increases as H_x decreases, although the compression ratio itself remains $R = 1$.

The entropy function can be shown to have the following bounds:

$$0 \leq H_x \leq \log_2(I) \qquad \textbf{(2.18)}$$

Consider the probability function $P_x[i]$ of some data set. By definition, $P_x[i]$ has the following characteristics:

$$0 \le P_x[i] \le 1 \tag{2.19}$$

$$\sum_{i=0}^{I-1} P_x[i] = 1 \tag{2.20}$$

Equations 2.19 and 2.20 imply that $P_x[i]$ can take on two extreme forms, as shown in Figures 2.1 and 2.2.

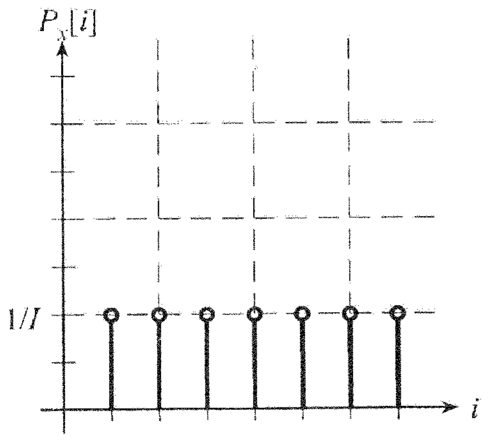

Figure 2.1 Example of a uniform probability distribution function

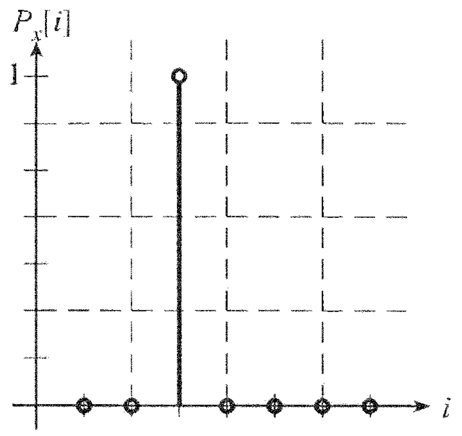

Figure 2.2 Example of a singular probability distribution function

In Figure 2.1, the probability function is uniformly distributed, that is all values of $P_x[i]$ are the same. This implies that every event in the data set has an equal probability of occurrence, thus for a uniform distribution:

$$P_x[i] = \frac{1}{I} \text{ for all } i \tag{2.21}$$

Now, the entropy for a uniform probability function may be calculated:

$$\begin{aligned} H_x &= \sum_{i=0}^{I-1} P_x[i] \log_2\left(\frac{1}{P_x[i]}\right) = \sum_{i=0}^{I-1} \left(\frac{1}{I}\right) \log_2\left(\frac{1}{1/I}\right) \\ &= (I)\left(\frac{1}{I}\right) \log_2\left(\frac{1}{1/I}\right) = \log_2(I) \text{ bits / word} \end{aligned} \tag{2.22}$$

Thus for a uniformly distributed probability function of size I, the entropy is simply $\log_2(I)$ bits/word.

For the case of Figure 2.2, consider a probability function $P_x[i]$ such that:

$$P_x[i] = \begin{cases} 1 & \textit{for one unique value of } i \\ 0 & \textit{elsewhere} \end{cases} \tag{2.23}$$

Such a distribution will be called a singular distribution and the location of the non-zero element of $P_x[i]$ within the span $i = 0,\ldots,(I-1)$ is irrelevant. The entropy of this probability function may also be explicitly calculated:

$$H_x = \sum_{i=0}^{I-1} P_x[i]\log_2\left(\frac{1}{P_x[i]}\right) = (1)\log_2\left(\frac{1}{1}\right) = 0 \text{ bits/word} \tag{2.24}$$

Therefore, from Equations 2.22 and 2.24, the bounds in Equation 2.19 may be assumed.

It may be postulated then that as $P_x[i]$ moves from a uniform distribution as given in Equation 2.21 to a singular distribution as given in Equation 2.23, the value of the entropy approaches its lower bound, and vice versa. That is:

$$P_x[i] \to P_{\text{SINGULAR}} \Rightarrow H_x \to 0 \tag{2.25}$$

$$P_x[i] \to P_{\text{UNIFORM}} \Rightarrow H_x \to \log_2(I) \tag{2.26}$$

The concept of entropy can now be related to data compression. Shannon's noiseless coding theorem [13] shows that the binary compression ratio R_B of an entropy-based data compression scheme is bounded in the maximum by the reciprocal of the entropy H_x:

$$(R_B)\log_2(I) \leq \frac{1}{H_x} \tag{2.27}$$

Therefore, reducing the entropy of a given data set, and consequently altering its probability density function, can result in a higher compression ratio. In general, data compression schemes exploit entropy by taking advantage of the fact that most real data sets do not have a uniformly distributed probability function, and thus an average binary compression ratio R_B can be achieved such that $R_B >> 1$.

In practice, achieving a value of R_B that approaches the entropy bound involves choosing an appropriate encoding technique. For creating optimal binary codes for any given input, Huffman coding can be used [14]. To measure data compression performance in this regard, it is only necessary to calculate the binary length of a Huffman encoded data set; this can be done without actually performing the Huffman encoding.

It is also very important to note that when the entropy of transform coefficients is being considered, the singular distribution of Equation 2.23 is equivalent to achieving optimal energy packing efficiency as discussed in the end of Section 2.4. Thus, the goal of improving decorrelation and increasing variance benefits both thresholding and quantizing, and entropy methods.

Chapter 3: Cost Functions

3.1 Introduction

As outlined in Chapter 1, it is necessary to define some cost functions $J(x)$ in order to determine which segmentation of a particular signal is optimal. These cost functions should be chosen with concern for the type of processing that is going to be used. It would be appropriate, for example, to choose a cost function that measures entropy if entropy-based coding is to be used, or a cost function that measures decorrelation or energy packing if quantizing or threshold coding is to be used, and so on.

It is also necessary to determine how to appropriately evaluate the combined effective costs of these cost functions, by choosing an appropriate combining function $K(J_0, J_1, \ldots)$. In general, if the cost function $J(x)$ is also a function of the segment length, $K(J_0, J_1, \ldots)$ will be calculated as the sum of the individual cost functions. If the cost function $J(x)$ is independent of the segment length, $K(J_0, J_1, \ldots)$ will be calculated as the mean of the individual cost functions.

3.2 Measures of Decorrelation

As discussed in Chapter 2, successful decorrelation of the input signal using transform coding can lead to improved data compression performance. This implies that a cost function that measures the amount of decorrelation achieved would be appropriate to use when thresholding or quantizing techniques are employed. Hamidi and Pearl [15] proposed a metric of decorrelation they termed "fractional correlation," which quantifies the amount of residual correlation remaining in the autocovariance matrix of the transform coefficients. Formally:

$$J_{FC}(x) = \frac{\left|\mathbf{C}_{xx} - \mathbf{C}'_{xx}\right|^2}{\left|\mathbf{C}_{xx} - \mathbf{I}_N\right|^2} \tag{3.1}$$

$$\mathbf{C}'_{xx} = \left[\mathbf{T}^T\right]\left[\operatorname{diag}\left[\mathbf{C}_{XX}\right]\right]\left[\mathbf{T}\right] \tag{3.2}$$

where $\mathbf{I}_N$ is the $N \times N$ identity matrix, $\mathbf{T}$ is an $N \times N$ linear orthogonal transform matrix, and $|\cdot|^2$ is the Hilbert-Schmidt weak norm of a matrix defined as follows:

$$\left|\mathbf{A}_N\right|^2 \equiv \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(a_{ij}\right)^2 \tag{3.3}$$

This operation takes the difference between the autocovariance of the original signal, and the autocovariance of a reconstruction of the original signal from only the decorrelated elements, and then normalizes it by the denominator term. As less and less off-diagonal terms of $\mathbf{C}_{XX}$ have significant value, thus indicating that more decorrelation has occurred, the difference in the numerator of Equation 3.1 will become smaller and smaller. If the transform perfectly decorrelates the input signal, $\mathbf{C}_{XX}$ will be diagonal, and thus $\mathbf{C}_{xx} = \mathbf{C}'_{xx}$ and $J_{FC}(x) = 0$.

Since $J_{FC}(x)$ is independent of the length of the segment x, the combining function will be the mean of the cost functions, that is:

$$K(J_0, J_1, \ldots, J_{L-1}) = \frac{1}{L}\sum_{l=0}^{L-1} J_l \tag{3.4}$$

$J_{FC}(x)$ is also independent of the absolute magnitude of the segment. Although this cost function is a very accurate measure of decorrelation, it is also very computationally expensive, as the calculation of the various autocovariance matrices is quite costly.

3.3 Measures of Energy Packing

Another metric that may be employed is the ordered energy packing efficiency (OEPE), which is a modified form of the energy packing efficiency measure proposed by Katajima [16]. The OEPE is defined as follows:

$$J_{OEPE}(x, \theta) = -\frac{\sum_{j=0}^{\theta-1} |X'[j]|^2}{\sum_{i=0}^{N} |X[i]|^2} \tag{3.5}$$

where X is the transform of x, X' is a rearrangement of X in decreasing order of magnitude, and θ is some arbitrary constant. The OEPE measures the amount of energy contained in the first θ coefficients relative to the total energy of the segment. The result is made negative, since all the other cost functions return lower values for improved costs.

As in the previous case, $J_{OEPE}(x)$ is a ratio, and its value is independent of the absolute magnitude the segment x, as well as the length of the segment x, and so the combining function will again be the mean as defined in Equation 3.4. Although the OEPE cost function still requires the calculation of the transform coefficients, its computational cost is significantly less than the cost of the fractional correlation cost function.

3.4 Measures of Entropy

When entropy coding will be employed, it may be appropriate to choose cost functions based on entropy. One obvious measure is the direct entropy of the transform coefficients:

$$J_{ENTROPY}(x) = -\sum_{i=0}^{I-1} p[i]\log_2(p[i]) \qquad \textbf{(3.6)}$$

where, as defined in Chapter 2, $p[i]$ is the probability of occurrence of the i th symbol in X, as drawn from an alphabet of I possible symbols. As in the previous two cases, Equation 3.4, the mean of the cost functions, is used as the combining function, since the entropy of a segment x is independent of the segment length or magnitude.

In some cases, it may be beneficial to employ a cost function whose value is a function of the segment length. Such a cost function is termed an *additive* measure. Coifman, et al. [6] have proposed the so-called "norm" cost function as:

$$J_{NORM}(x) = -\sum_{j=0}^{N-1} |X[j]|^2 \log_2 |X[j]|^2 \qquad \textbf{(3.7)}$$

Coifman has also shown how minimizing Equation 3.7 also minimizes the entropy as given in Equation 3.6, while at the same time incorporating some consideration for variations in segment lengths. Since Equation 3.7 is a function of segment length, the combining function is defined as the sum of the cost functions, instead of the mean:

$$K(J_0, J_1, \ldots, J_{L-1}) = \sum_{l=0}^{L-1} J_l \qquad \textbf{(3.8)}$$

Chapter 4: Search Mechanisms

4.1 General Issues

A large number of methods exist to search through possible segmentations, in general, however, the computational cost increases as the thoroughness and overall flexibility of the method increases. In addition, as the size of the individual segments becomes smaller, there is often a diminishing return in terms of performance; for example, it is generally pointless to try to compress segments of one or two samples in length. Considering segmentations with very short segments, which can be termed fine resolution, can also increase the computational cost.

To reduce both the computational cost, as well as account for the diminishing returns, a compromise is made by choosing a maximum resolution, or conversely a minimum segment size. This minimum segment length N_U will be termed the unit segment length. An additional compromise will be made by requiring that all segments within a particular segmentation have lengths which are integer multiples of the unit segment length. The latter restriction greatly reduces the complexity of any search algorithm, without sacrificing too much flexibility.

4.2 Exhaustive Search

Given the aforementioned constraints on resolution, an exhaustive search will be defined as a search that explicitly considers every possible segmentation of a particular signal. For example, given a signal x of length $N = 4$, and choosing the finest possible resolution, that is a unit segment length of $N_U = 1$, then an exhaustive search of x would consider the following segmentations:

$$\begin{array}{ll} x_{0123} & x_0 x_1 x_{23} \\ x_0 x_{123} & x_0 x_{12} x_3 \\ x_{01} x_{23} & x_{01} x_2 x_3 \\ x_{012} x_3 & x_0 x_1 x_2 x_3 \end{array} \qquad \textbf{(4.1)}$$

For another signal y of length $N = 6$, choosing $N_U = 2$ for an exhaustive search would yield the following segmentations:

$$\begin{array}{l} x_{012345} \\ x_{01} x_{2345} \\ x_{0123} x_{45} \\ x_{01} x_{23} x_{45} \end{array} \qquad \textbf{(4.2)}$$

Thus the number of possible segmentations considered by an exhaustive search is:

$$2^{\left(\frac{N}{N_U}-1\right)} \quad (4.3)$$

The cost of this type of search can be considered by calculating the total number of unit segments that need to be evaluated. The search cost of an exhaustive search is simply the product of the total number of segmentations considered, as given in Equation 4.3, and the length of the signal normalized by the unit segment length, as follows:

$$\left(\frac{N}{N_U}\right)\left(2^{\left(\frac{N}{N_U}-1\right)}\right) \quad (4.4)$$

4.3 Reducing the Cost of an Exhaustive Search

While an exhaustive search has the advantage of considering every possible segmentation, within the restrictions listed in Section 4.1, it has the decided disadvantage of being extremely computational expensive. Figure 4.1 plots the search cost of an exhaustive search, as given in Equation 4.4, versus the normalized signal length $\frac{N}{N_U}$.

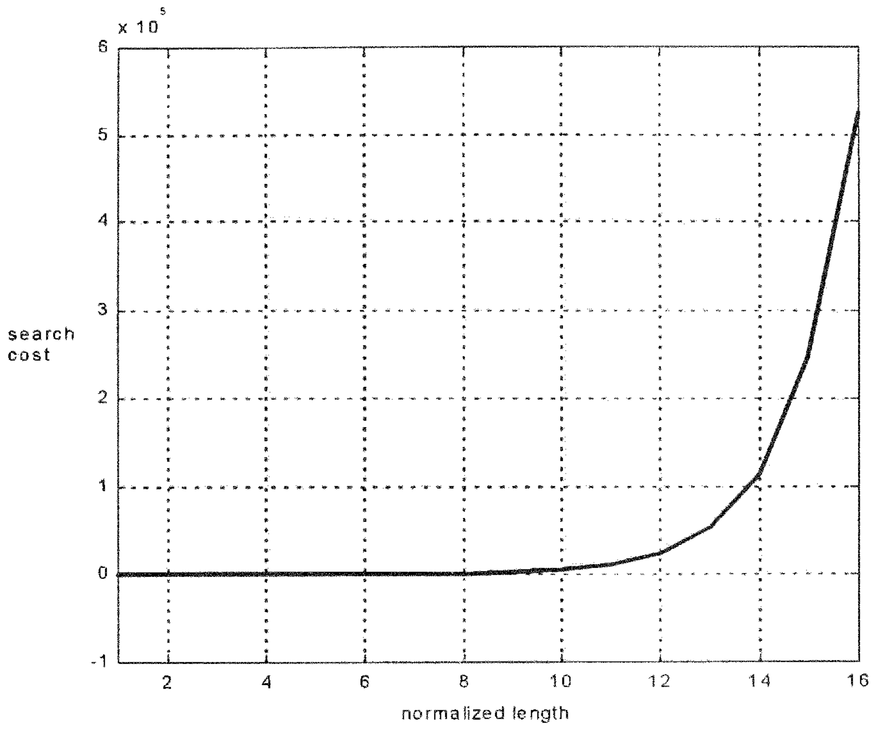

Figure 4.1 Exhaustive search cost

It is therefore prudent to develop a search mechanism that retains the flexibility of the exhaustive search but reduces the computational cost. One such approach was proposed by Xiong, et al. [5] in the context of wavelet packets. This approached, termed a dynamic programming (DP) search, borrows from the algorithmic theory of the same name. This approach works by dividing a set of large problems into smaller problems that can be solved independently, and then storing the solutions to the smaller problems for reuse, where applicable, in the large problems. Essential to this method is the characteristic of independence of the smaller problems, which in this case applies to the cost functions of Chapter 3. All of the cost functions there can be considered independent in this sense, that is, they are functions only of the particular segment being evaluated, and are unaffected by any other segments.

Figure 4.2 provides a graphic example of the DP search approach.

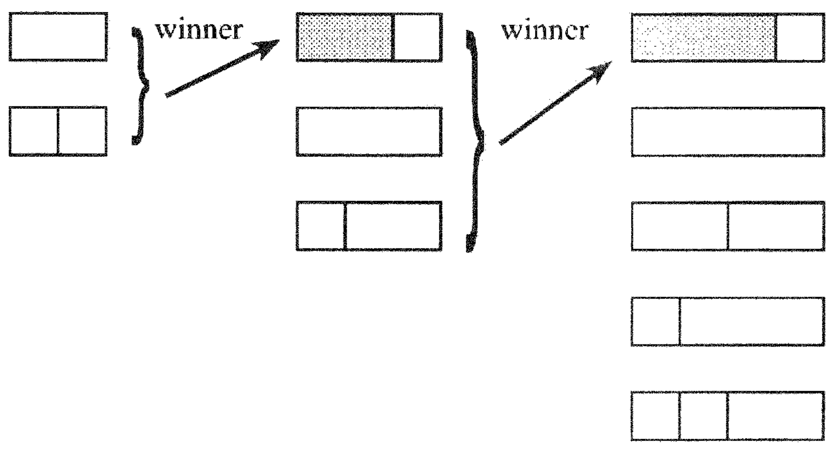

Figure 4.2 Dynamic programming search technique

As may be observed, instead of explicitly calculating each possible segmentation, a graduated approach is taken. First, an exhaustive search is performed on the first two unit segments of the signal; obviously there are only two possible segmentations in the case. The winner of this search is then introduced into a search of the first three unit segments. In this case, the search is no longer explicitly exhaustive, as the previous search accounted for some of the current possibilities already. This process is then repeated for the first four unit segments, and so on.

This method has the advantage of being just as flexible as the exhaustive search, that is it also considers every possible segmentation of x, but at a reduced cost. The cost of a DP search may be calculated as:

$$N - \frac{1}{2} + \sum_{n=1}^{N} \left(n \left(2^{(n-2)} \right) \right) \tag{4.5}$$

Figure 4.3 illustrates the cost of the DP search, and Figure 4.4 compares the cost of the exhaustive search with the cost of the DP search.

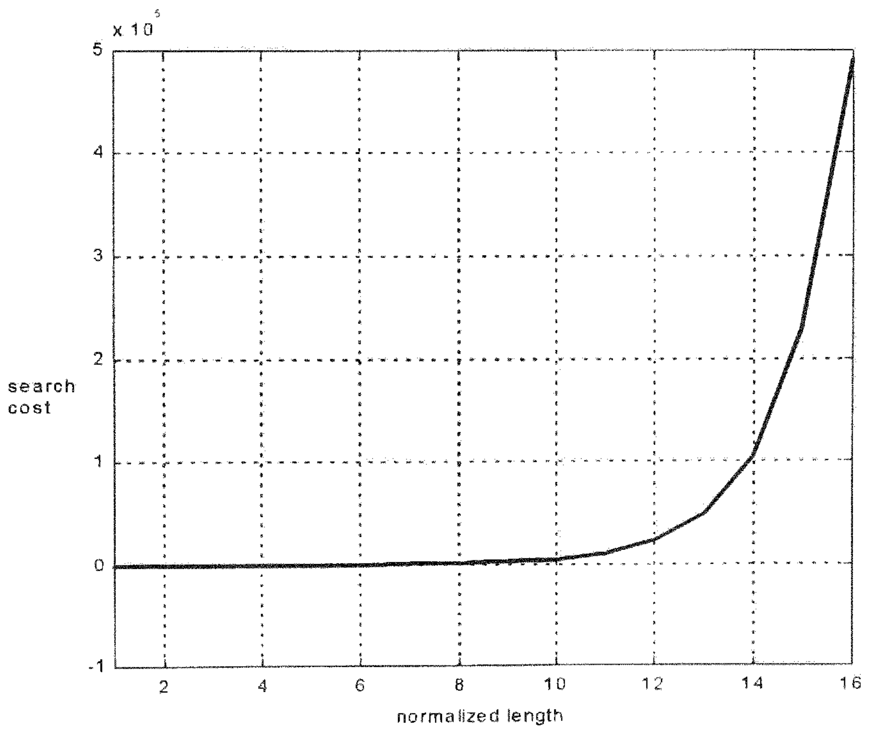

Figure 4.3 Dynamic programming search cost

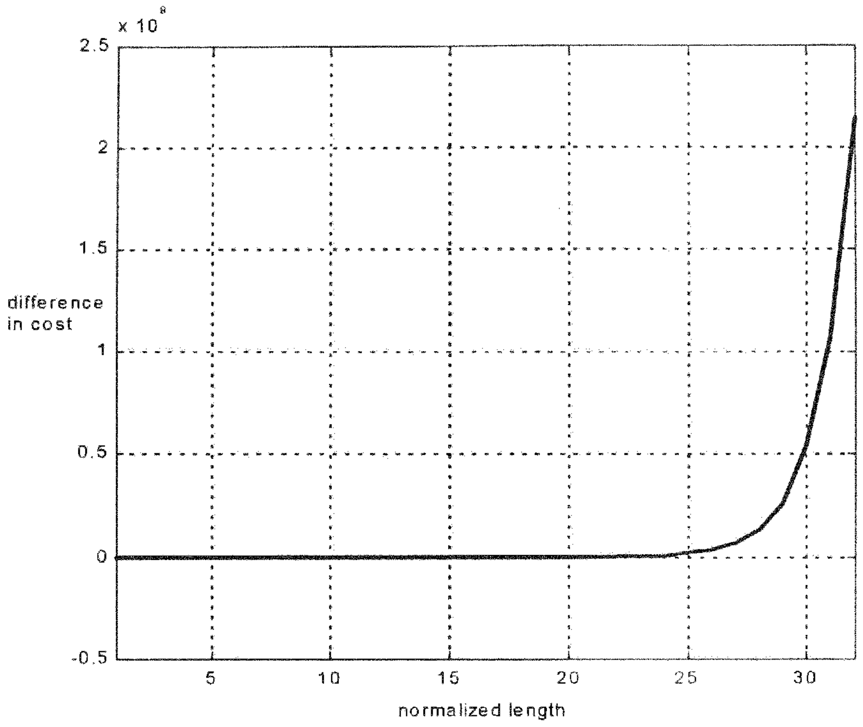

Figure 4.4 Difference between exhaustive search cost and DP search cost

Chapter 5: Examples

5.1 Evaluation of Performance

In order to evaluate the performance of the adaptive segmenting methodology, it is necessary to implement some actual data compression algorithms and use them to process actual data. Some metrics then need to be defined to quantify their performance. As discussed in previous chapters, data compression can be divided into two main categories, lossy and lossless; and so methods will be developed to consider adaptive segmenting in both contexts. In each case, the results using adaptive segmenting will then be compared to the results of applying the same processing using instead either uniform segmenting or no segmenting of any kind, in order to illustrate the performance gain.

For lossy compression, the algorithm will work as follows. The adaptive segmenting algorithm will be applied to the input signal using the appropriate cost functions, fractional correlation or OEPE in this case, and then each segment will be transformed using the DCT. Threshold coding will then be applied to each of these transformed segments, and a fixed number of coefficients will be retained for each unit segment length of each segment. For example, if 10 coefficients were being retained per unit segment, a segment of three unit segments in length would retain 30 coefficients, a segment of one unit segment in length would retain only 10 coefficients, and so on. This requirement will have the effect of retaining the same number of coefficients for each signal x, regardless of the particular segmentation chosen, thus allowing for appropriate comparison between different segmentations of the same signal. The retained coefficients will then be used to generate a reconstruction, and the quality of this reconstruction will be evaluated using parameters to be defined next.

The signal-to-noise ratio (SNR) will be used to specifically quantify the performance of the lossy algorithm by considering Equation 2.1, the reconstruction error, as the noise signal. This is a common metric in signal processing, and evaluates the ratio of signal power or magnitude to noise power or magnitude. The SNR will be calculated in two ways, the first using the mean-squared error (MSE) and the second using the mean-absolute error (MAE). These are as follows:

$$SNR_{MSE} = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_r^2}\right) \text{ dB} \qquad \textbf{(5.1)}$$

$$SNR_{MAE} = 20\log_{10}\left(\frac{\mu_{|x|}}{\mu_{|r|}}\right) \text{ dB} \qquad \textbf{(5.2)}$$

where $\mu_{|x|}$ is the mean of the absolute value of x and σ_x^2 is the variance of x, defined as:

$$\sigma_x^2 = \frac{1}{N}\sum_{n=0}^{N-1}(x[n])^2 \qquad \textbf{(5.3)}$$

While SNR_{MSE} is the more common of the two SNR measurements, the SNR_{MAE} is also important for this application, as it penalizes less for local larger transients in the error signal, and instead gives a good idea of the overall performance of the system.

For lossless compression, the process will be similar. The input signal will be adaptively segmented, using in this case the entropy-based set of cost functions, and each of the segments will then be transformed using the DCT. At this stage, though, all that remains is to calculate the Huffman lengths, as discussed in Chapter 2, of each segment and then sum the results. A lower total length will indicate a higher compression ratio and a superior segmentation. Note that for evaluating lossless methods, it is not necessary to reconstruct the signal. Letting H(x) be the total Huffman length of the transform coefficients in bits, then the metric for lossless compression will be defined as:

$$R_H = \frac{N_B}{H(x)} \text{ bits/bit} \qquad \textbf{(5.4)}$$

where N_B is the length of the original signal x in bits.

5.2 Test Signals

Four different signals will be used to demonstrate and evaluate adaptive segmenting. These signals are plotted in Figure 5.1 through Figure 5.4.

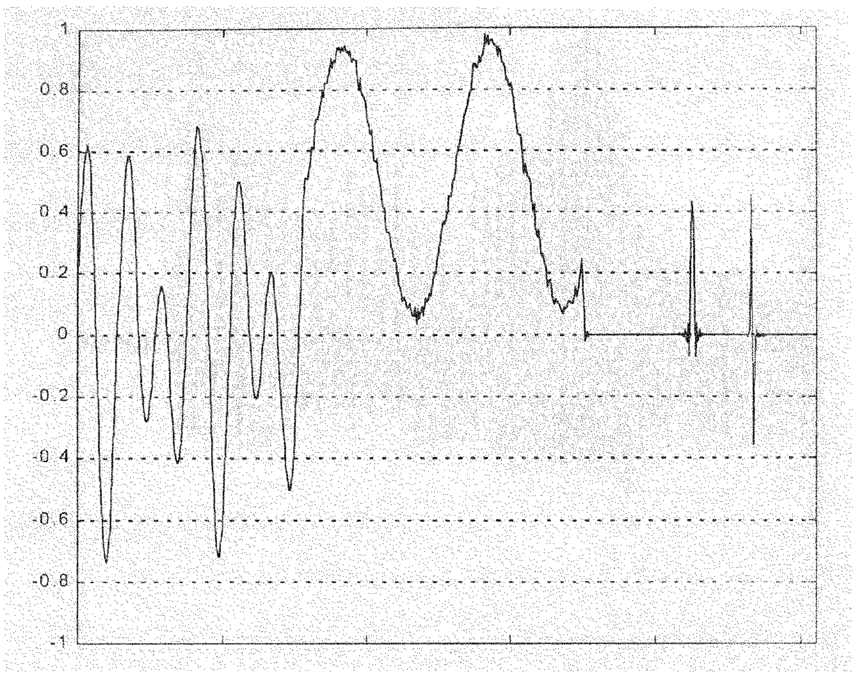

Figure 5.1 Test signal x_1

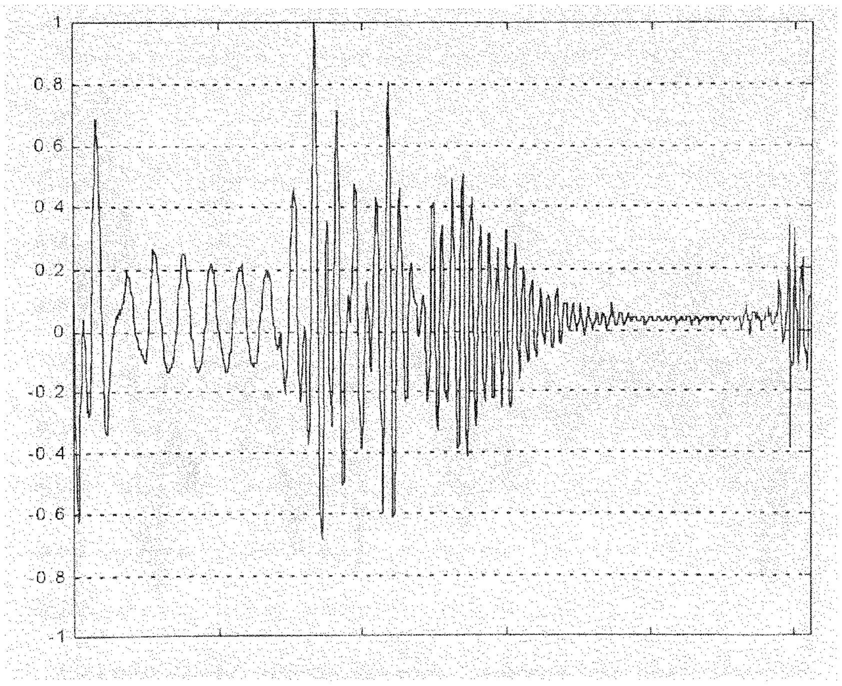

Figure 5.2 Test signal x_2

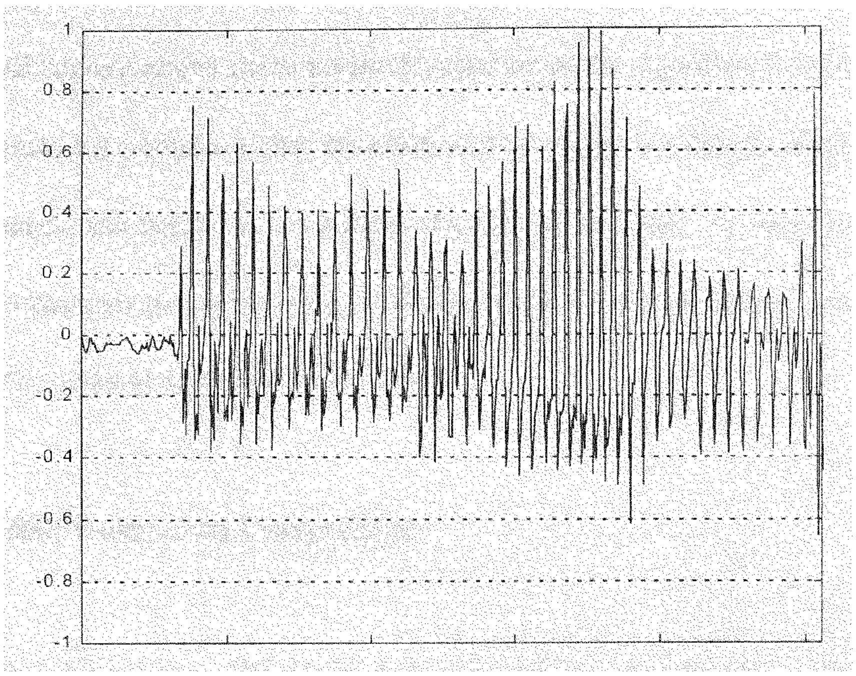

Figure 5.3 Test signal x_3

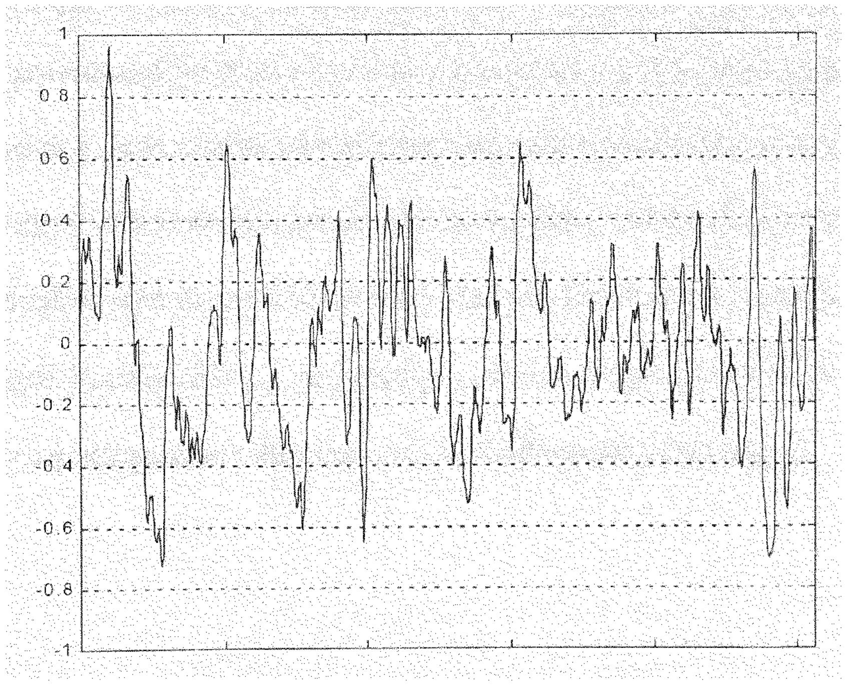

Figure 5.4 Test signal x_4

x_1 is a signal constructed to demonstrate the reaction of the algorithm to various general types of signals, i.e. periodic, noise, transient, and so on. x_2 is a sample of speech, x_3 is a sample of music, and finally x_4 is a sample of a biomedical signal. x_4 was chosen in particular to illustrate the performance of the algorithm with a signal that contains almost no stationarity, even at an extremely local level.

5.3 Evaluation Using Lossy Compression

Figures 5.5 through 5.29 demonstrate the performance gain of the adaptive segmenting methods over uniform or non-existent segmentations. For each signal, three different unit segment sizes are employed for further comparison, and only 25% of the coefficients are retained. Each figure contains three plots: the first is the original signal with its respective segmentation superimposed on it; the second is the reconstruction of the original signal after transform coding and thresholding; and the third is the error signal $r[n]$. Listed below each figure is information regarding performance, specifically, the unit segment length N_U, the SNR_{MSE}, and SNR_{MAE}, and the compression ratio R. Following each signal's set of figures is a discussion on the performance of that signal.

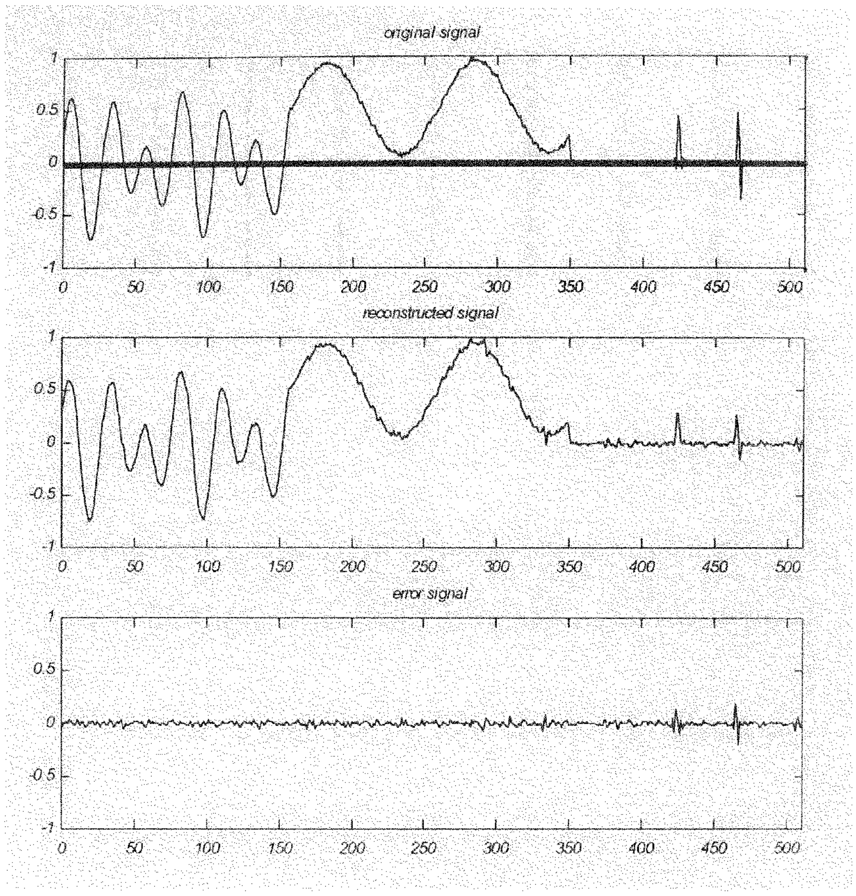

Figure 5.5 Example 1 of lossy compression of x_1

$N_U = 64$

$SNR_{MSE} = 23.3875$ dB

$SNR_{MAE} = 24.861$ dB

$R = 4$

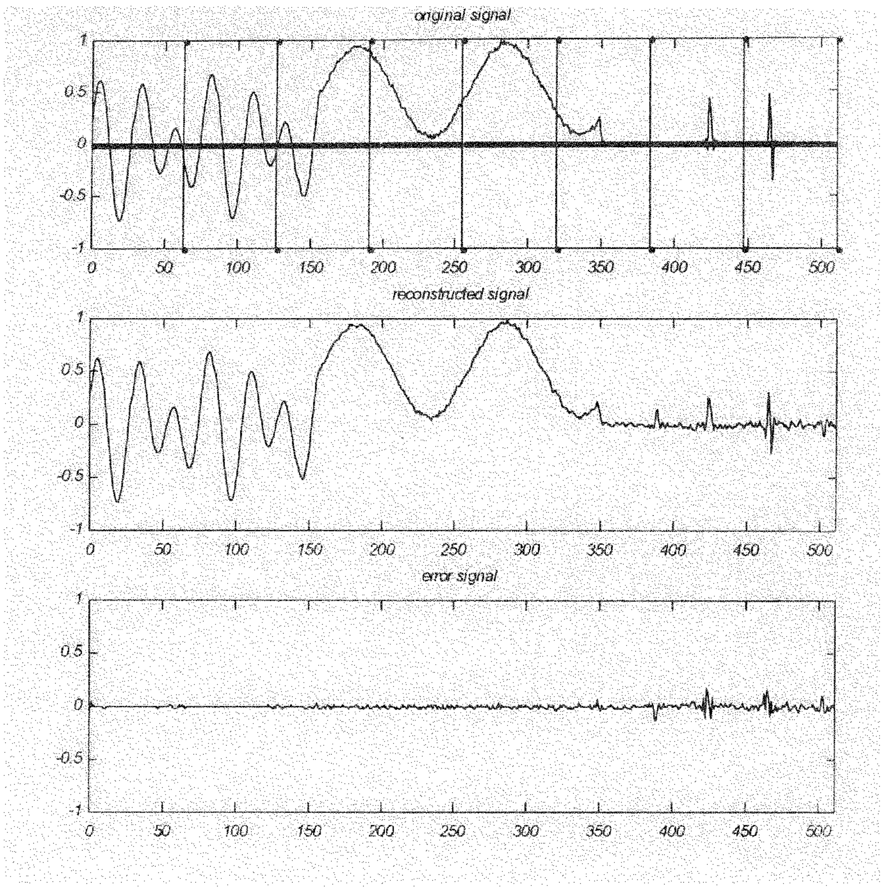

Figure 5.6 Example 2 of lossy compression of x_1

$N_U = 64$

$SNR_{MSE} = 23.5566$ dB

$SNR_{MAE} = 26.4746$ dB

$R = 4$

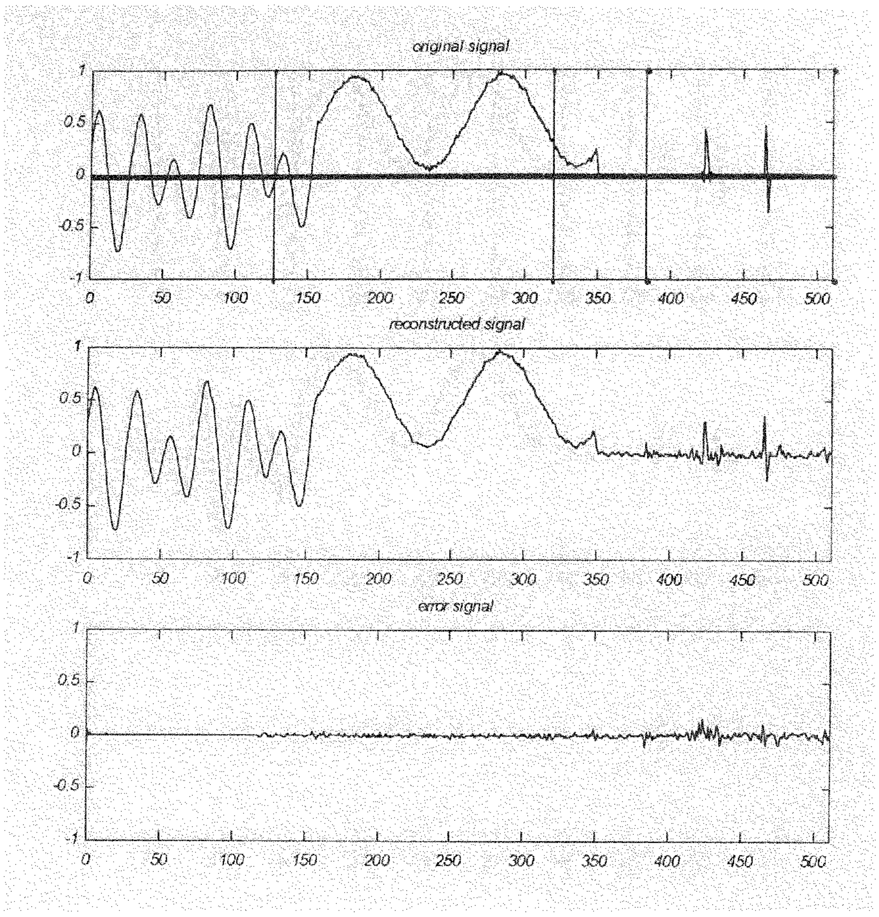

Figure 5.7 Example 3 of lossy compression of x_1

$N_U = 64$

$SNR_{MSE} = 24.6089$ dB

$SNR_{MAE} = 26.9645$ dB

$R = 4$

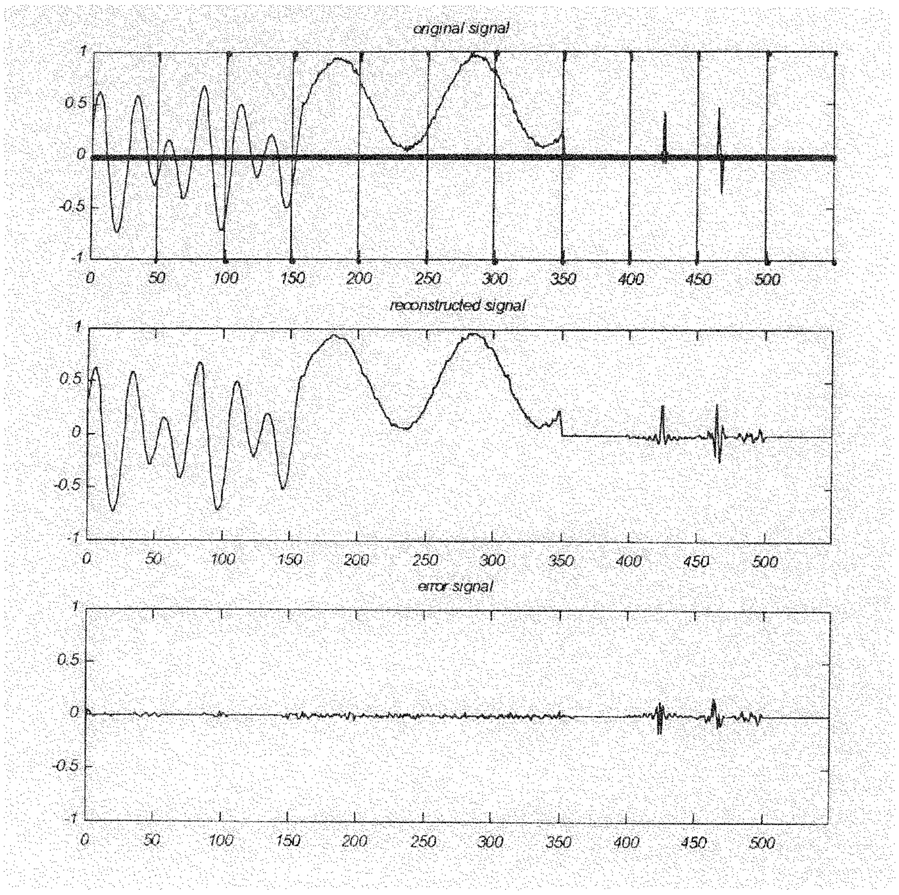

Figure 5.8 Example 4 of lossy compression of x_1

$N_U = 50$

$SNR_{MSE} = 23.8969$ dB

$SNR_{MAE} = 27.3993$ dB

$R = 4.1667$

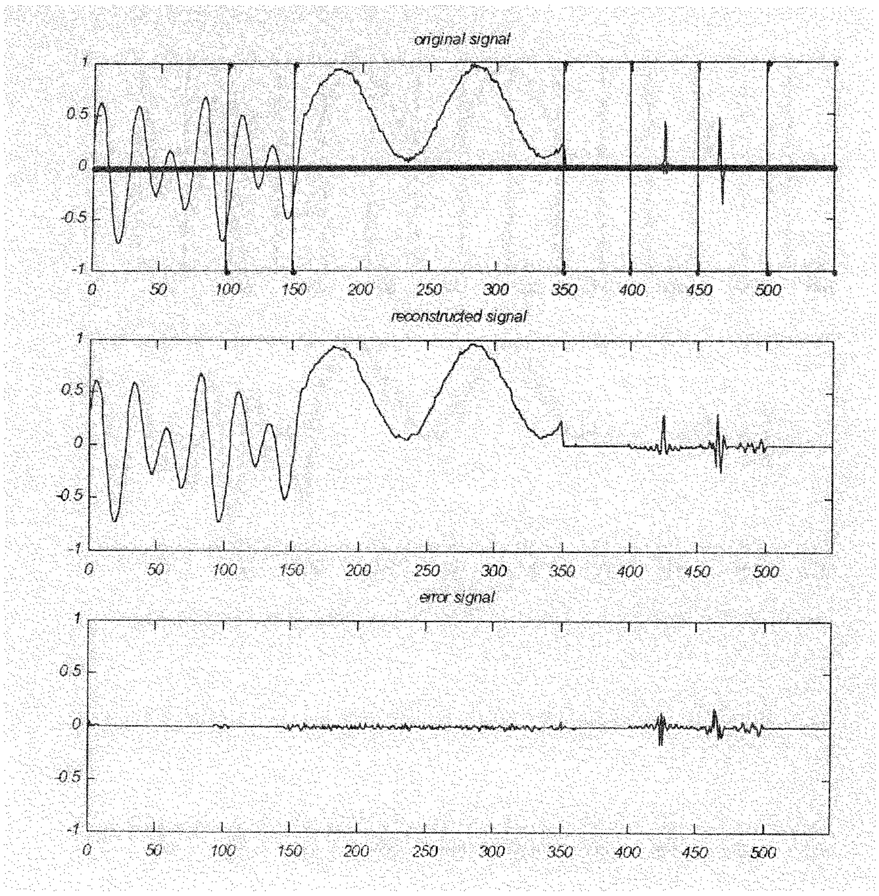

Figure 5.9 Example 5 of lossy compression of x_1

$N_U = 50$

$SNR_{MSE} = 23.9982$ dB

$SNR_{MAE} = 27.635$ dB

$R = 4.1667$

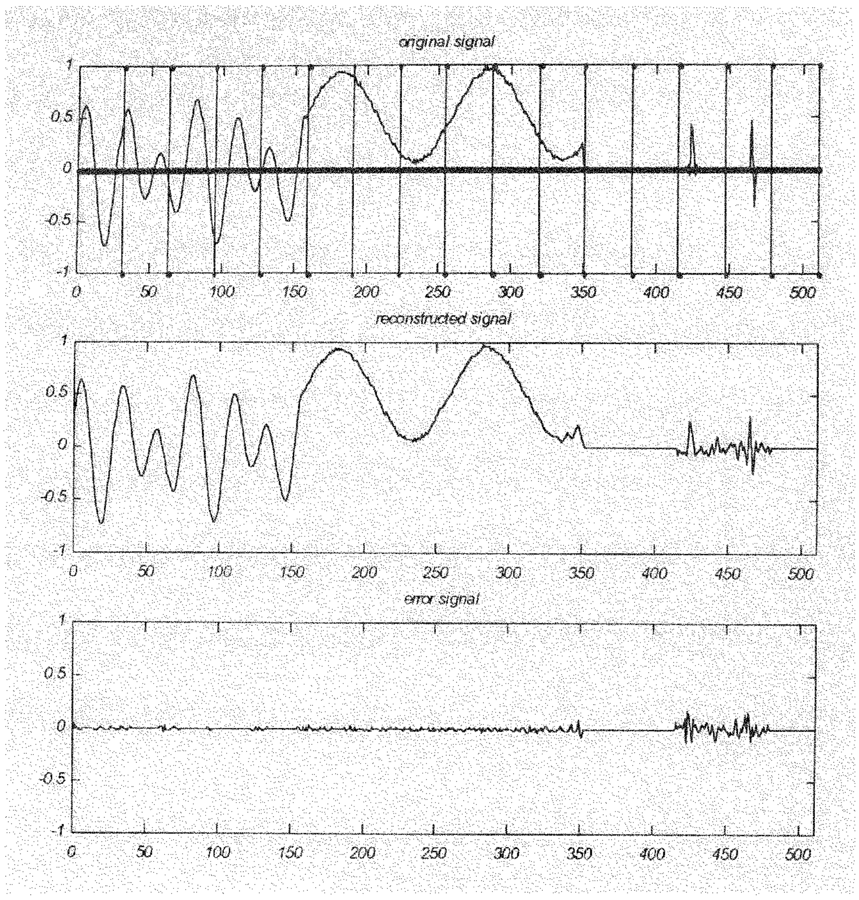

Figure 5.10 Example 6 of lossy compression of x_1

$N_U = 32$

$SNR_{MSE} = 23.2314$ dB

$SNR_{MAE} = 26.9347$ dB

$R = 4$

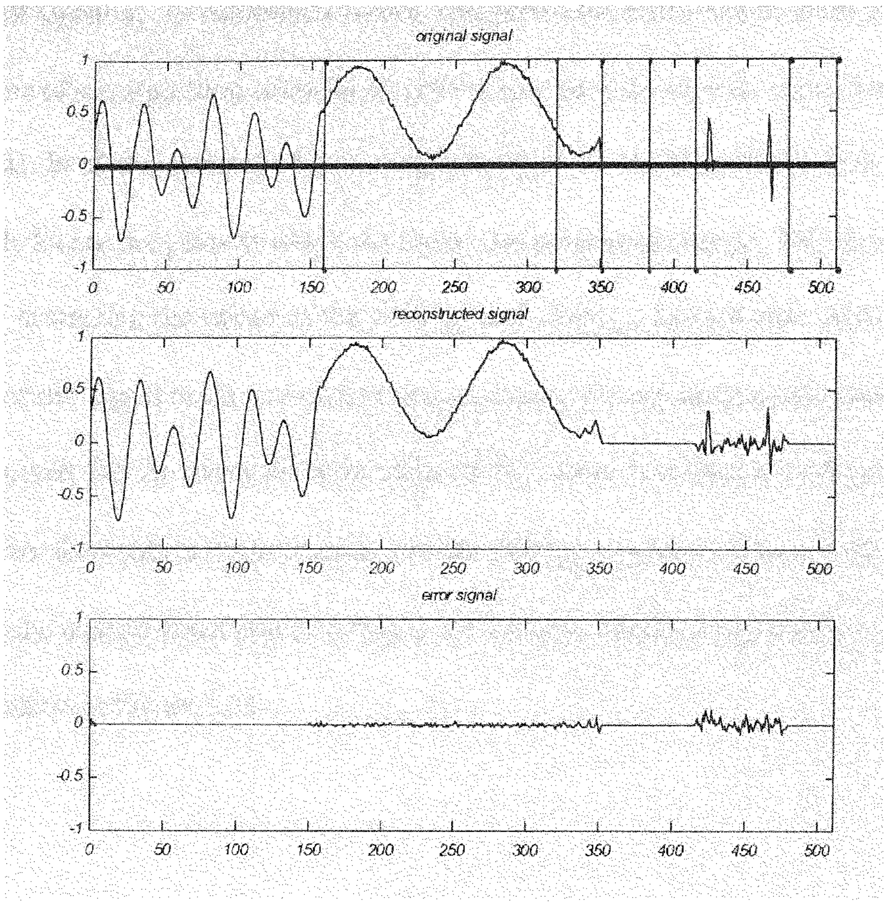

Figure 5.11 Example 7 of lossy compression of x_1

$N_U = 32$

$SNR_{MSE} = 24.8609$ dB

$SNR_{MAE} = 29.1472$ dB

$R = 4$

Test signal x_1, as mentioned before, was generated artificially in order to examine the behavior of the algorithm when faced with simple periodic signals, noise, and transient components. In all the cases of adaptive segmenting of x_1, the algorithm appears to successfully locate the point at which the signal characteristics change. This success is verified by examining the values of the SNR_{MSE} and SNR_{MAE} in each case. Although the simplicity of this signal tends to restrict the opportunity for any great performance gain, it can be observed that for every different value of N_U, there is at least a 1 dB gain in the SNR_{MSE} over the uniform segmentation, and the SNR_{MAE} gains more than 4 dB when comparing the unsegmented signal in Figure 5.5 with the adaptive segmentation at the finest resolution in Figure 5.11.

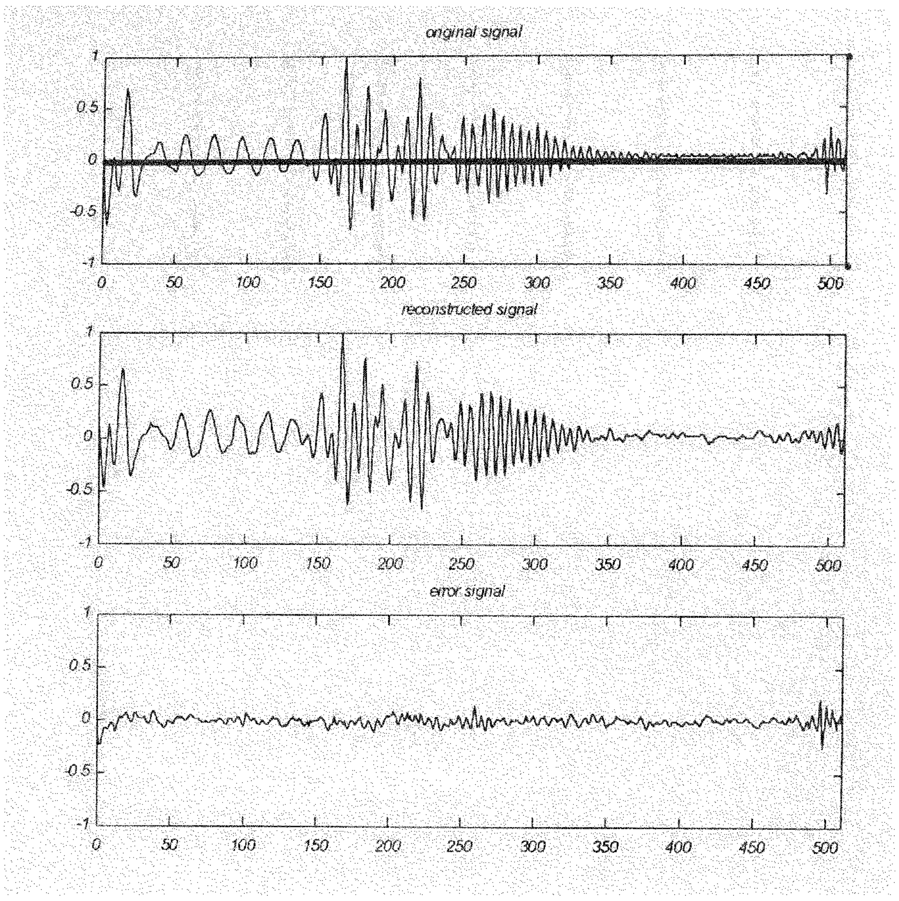

Figure 5.12 Example1 of lossy compression of x_2

$N_U = 64$

$SNR_{MSE} = 13.3586$ dB

$SNR_{MAE} = 13.0909$ dB

$R = 4$

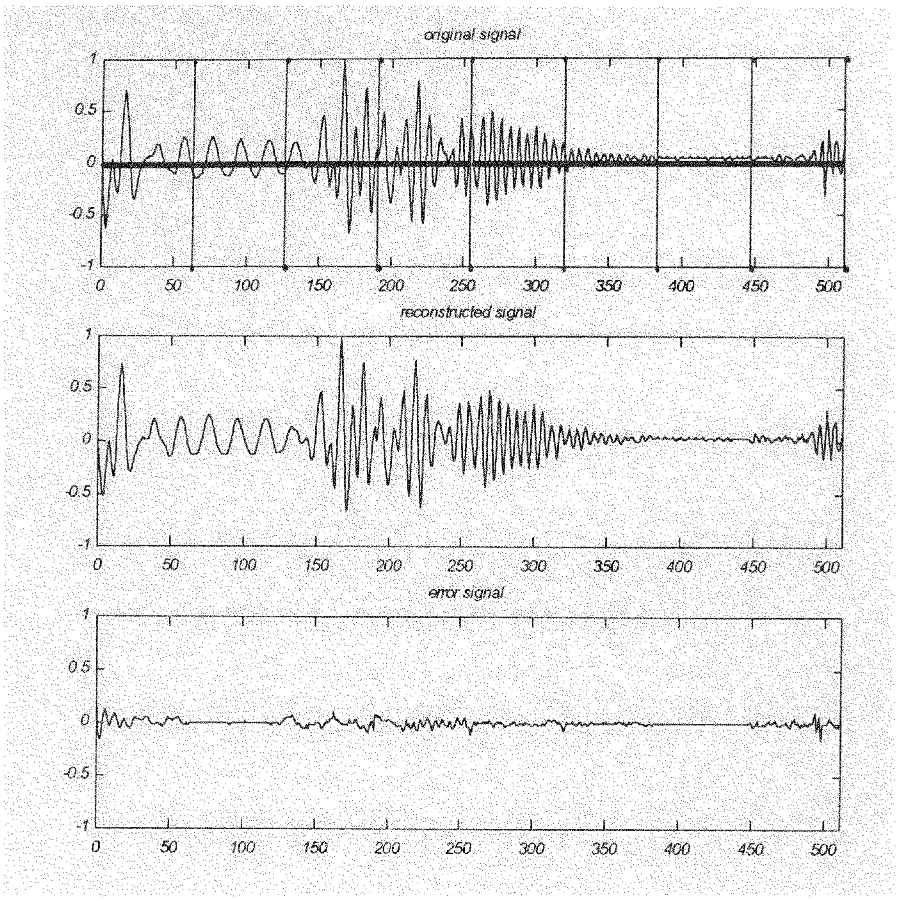

Figure 5.13 Example 2 of lossy compression of x_2

$N_U = 64$

$SNR_{MSE} = 16.218$ dB

$SNR_{MAE} = 17.0779$ dB

$R = 4$

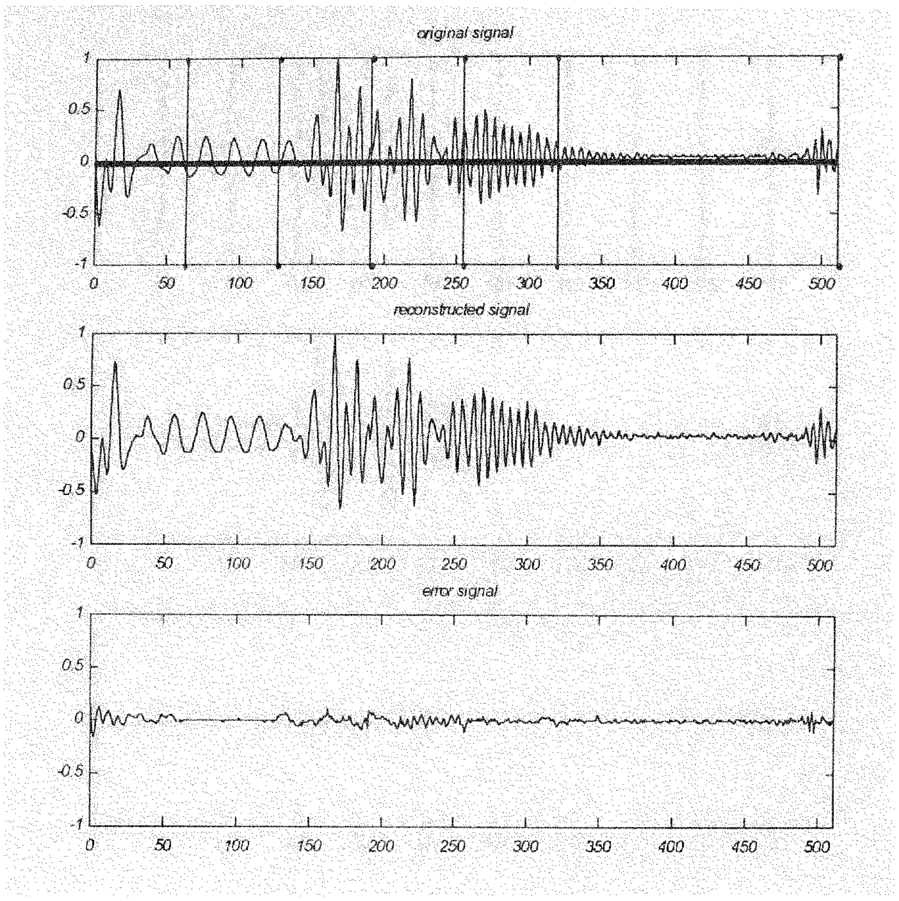

Figure 5.14 Example 3 of lossy compression of x_2

$N_U = 64$

$SNR_{MSE} = 16.325$ dB

$SNR_{MAE} = 16.8592$ dB

$R = 4$

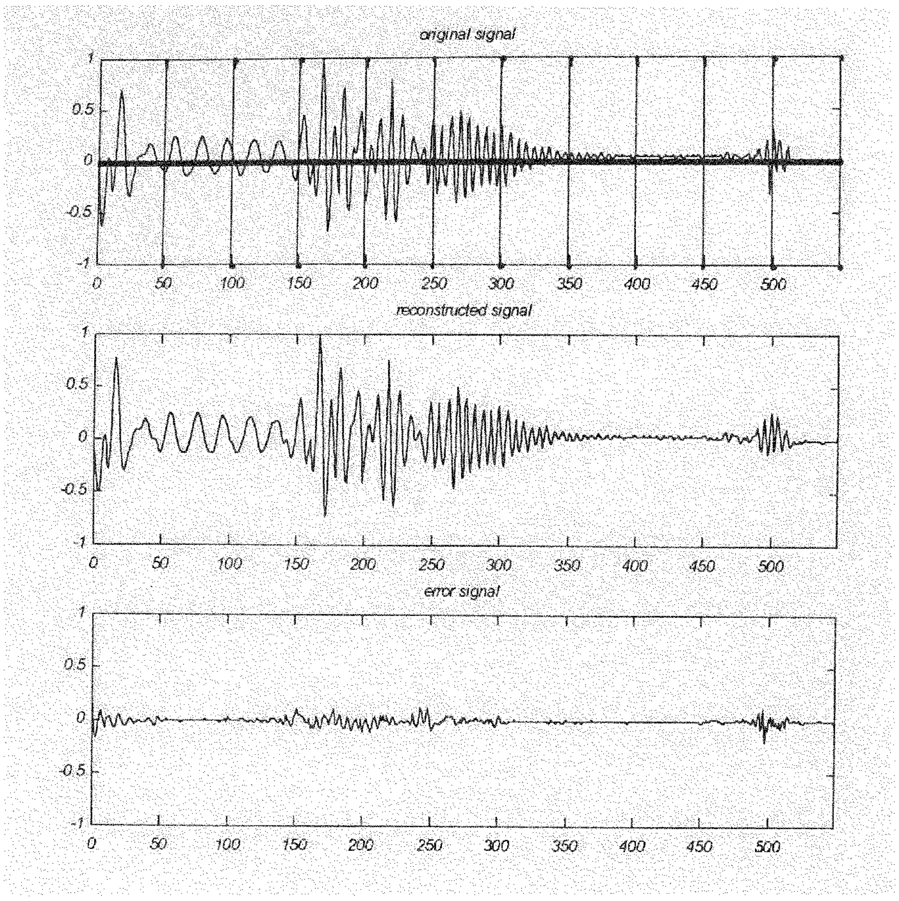

Figure 5.15 Example 4 of lossy compression of x_2

$N_U = 50$

$SNR_{MSE} = 15.488$ dB

$SNR_{MAE} = 16.606$ dB

$R = 4.1667$

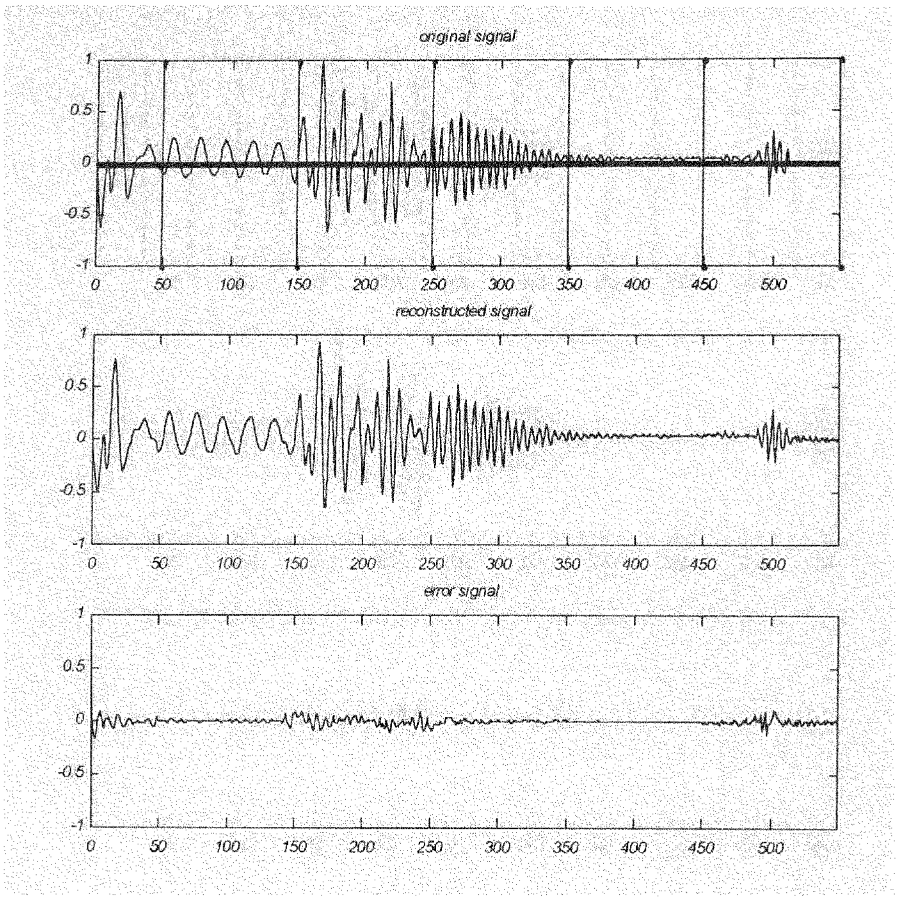

Figure 5.16 Example 5 of lossy compression of x_2

$N_U = 50$

$SNR_{MSE} = 16.0119$ dB

$SNR_{MAE} = 16.7696$ dB

$R = 4.1667$

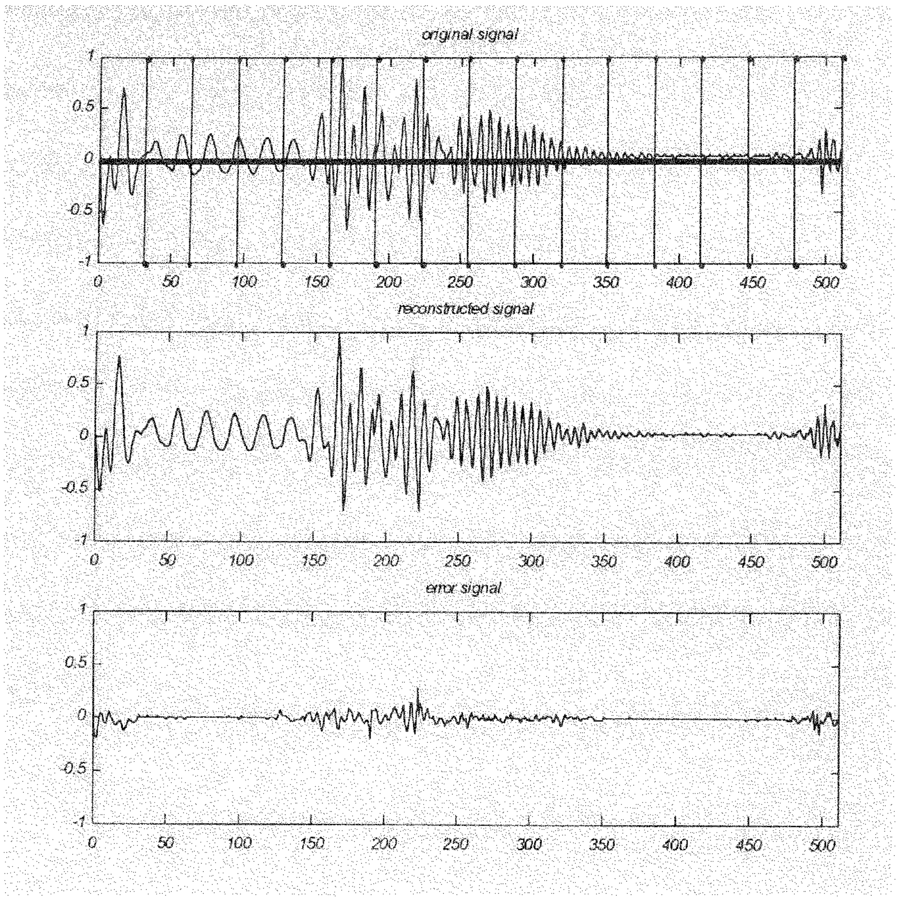

Figure 5.17 Example 6 of lossy compression of x_2

$N_U = 32$

$SNR_{MSE} = 14.3085$ dB

$SNR_{MAE} = 16.2024$ dB

$R = 4$

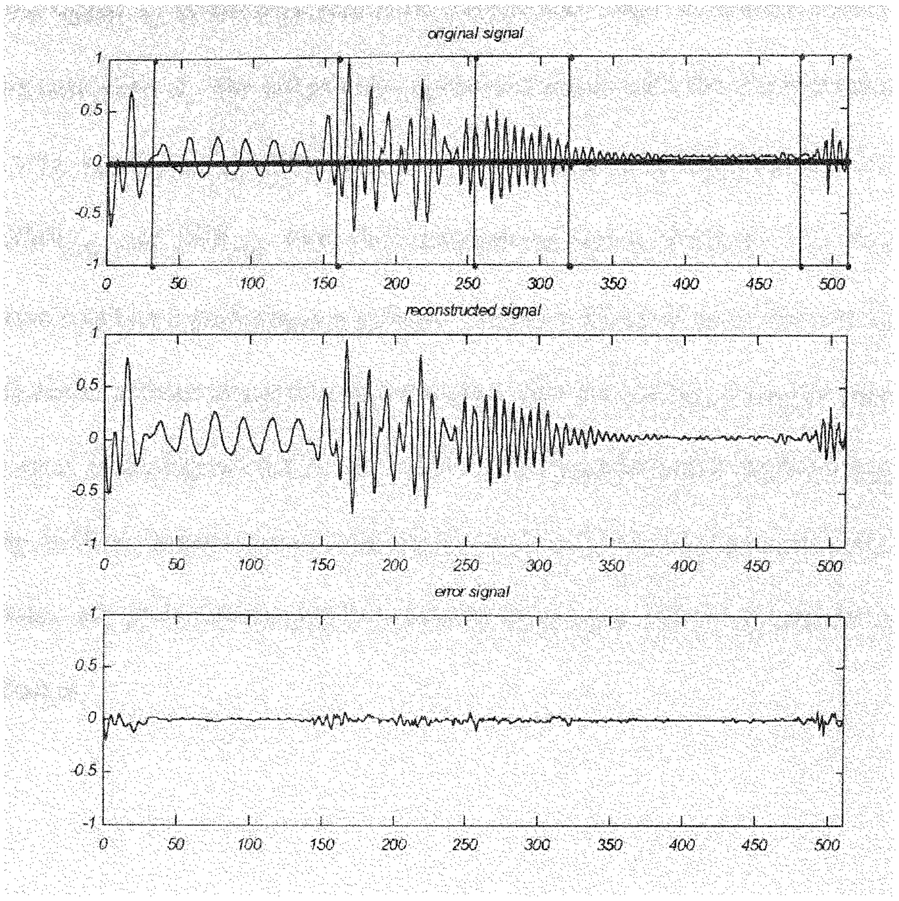

Figure 5.18 Example 7 of lossy compression of x_2

$N_U = 32$

$SNR_{MSE} = 17.3508$ dB

$SNR_{MAE} = 18.7347$ dB

$R = 4$

Test signal x_2 is more indicative of a signal that might be encountered in practice. As was the case with x_2, the adaptively segmented signal with the finest resolution, shown in Figure 5.18, has the best performance; more than double the signal power in both in terms of SNR_{MSE} and SNR_{MAE} over the unsegmented signal of Figure 5.12. In fact, this segmentation exhibits a performance gain of 1 dB to 3 dB over ever other variation listed. Again, it is obvious from inspection that the algorithm is locating the appropriate transition areas in the signal, which is in itself a positive affirmation of the adaptive segmenting method. It may also be noticed, for example, that as the search resolution becomes finer, the performance gap between the uniformly and the adaptively segmented signal increases.

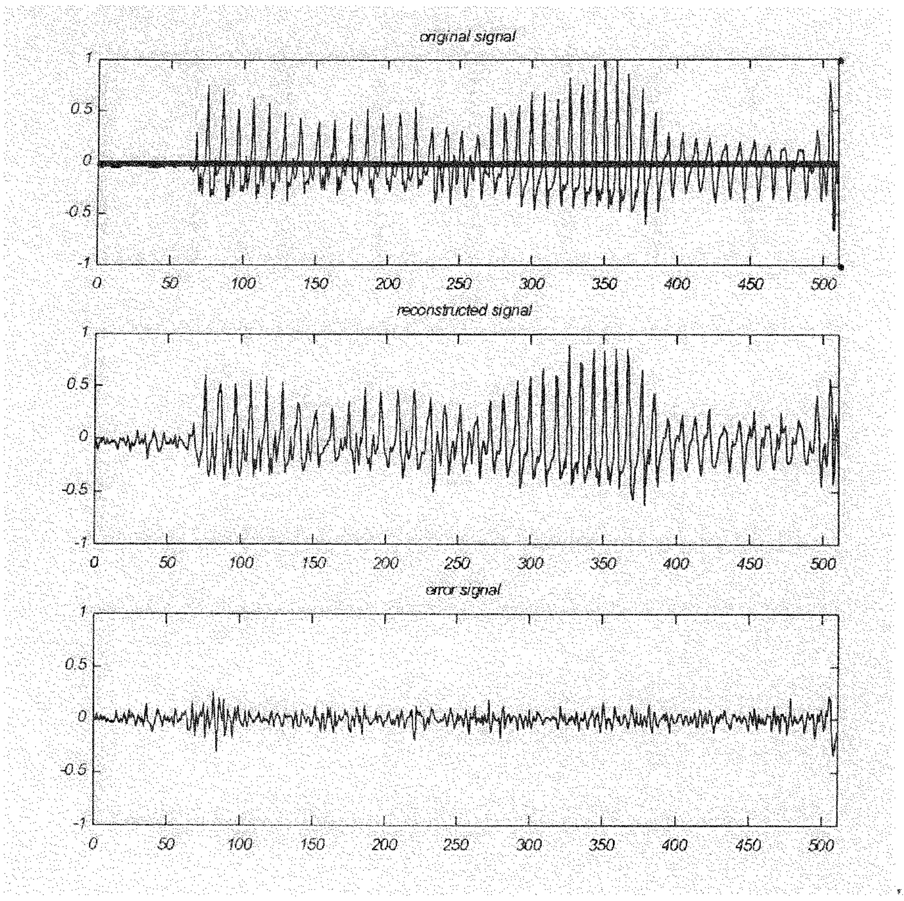

Figure 5.19 Example 1 of lossy compression of x_3

$N_U = 64$

$SNR_{MSE} = 11.087$ dB

$SNR_{MAE} = 11.0404$ dB

$R = 4$

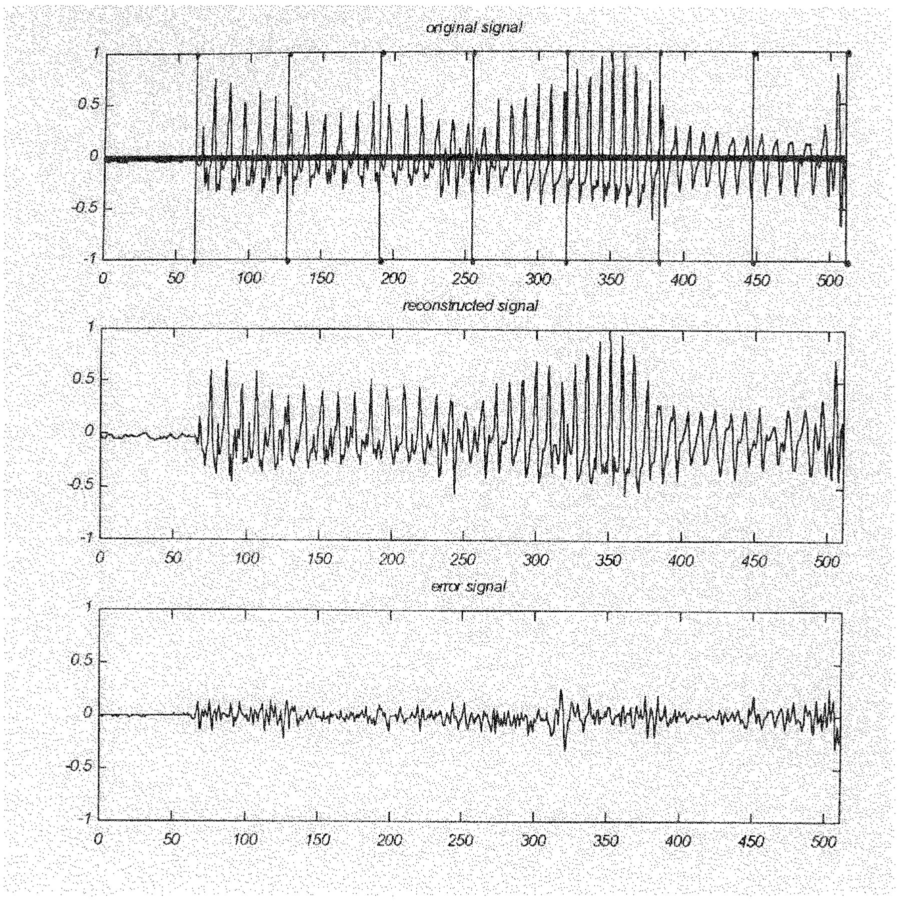

Figure 5.20 Example 2 of lossy compression of x_3

$N_U = 64$

$SNR_{MSE} = 11.1956$ dB

$SNR_{MAE} = 11.8459$ dB

$R = 4$

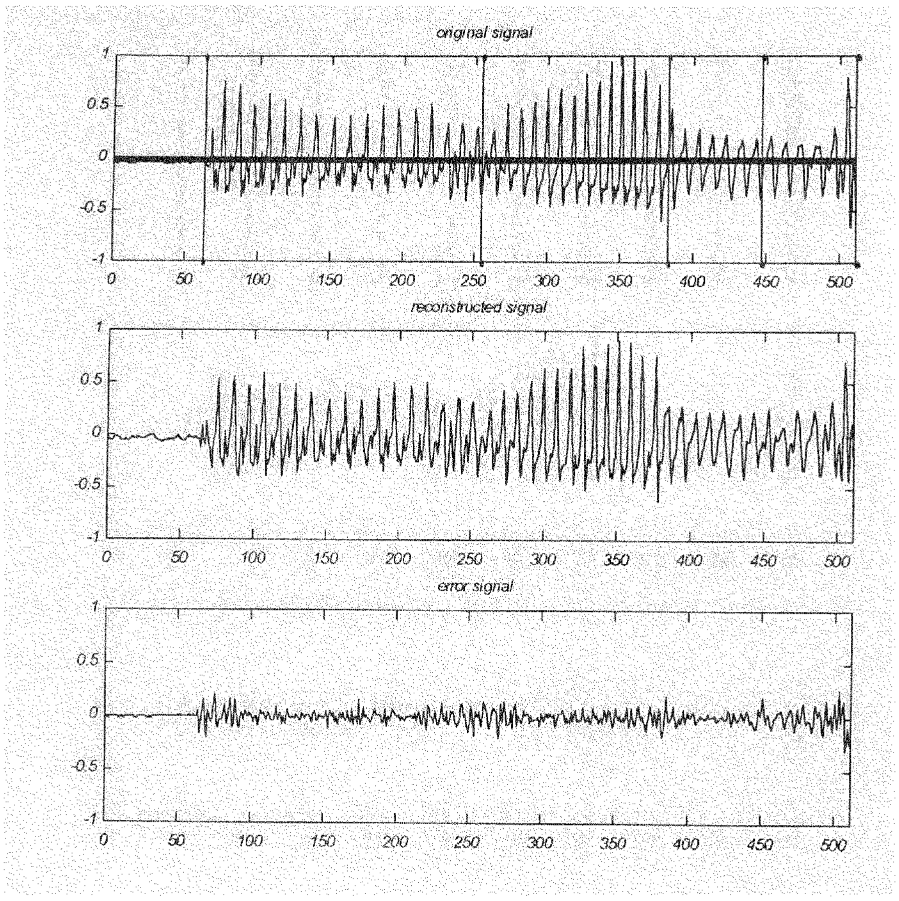

Figure 5.21 Example 3 of lossy compression of x_3

$N_U = 64$

$SNR_{MSE} = 12.037$ dB

$SNR_{MAE} = 12.5444$ dB

$R = 4$

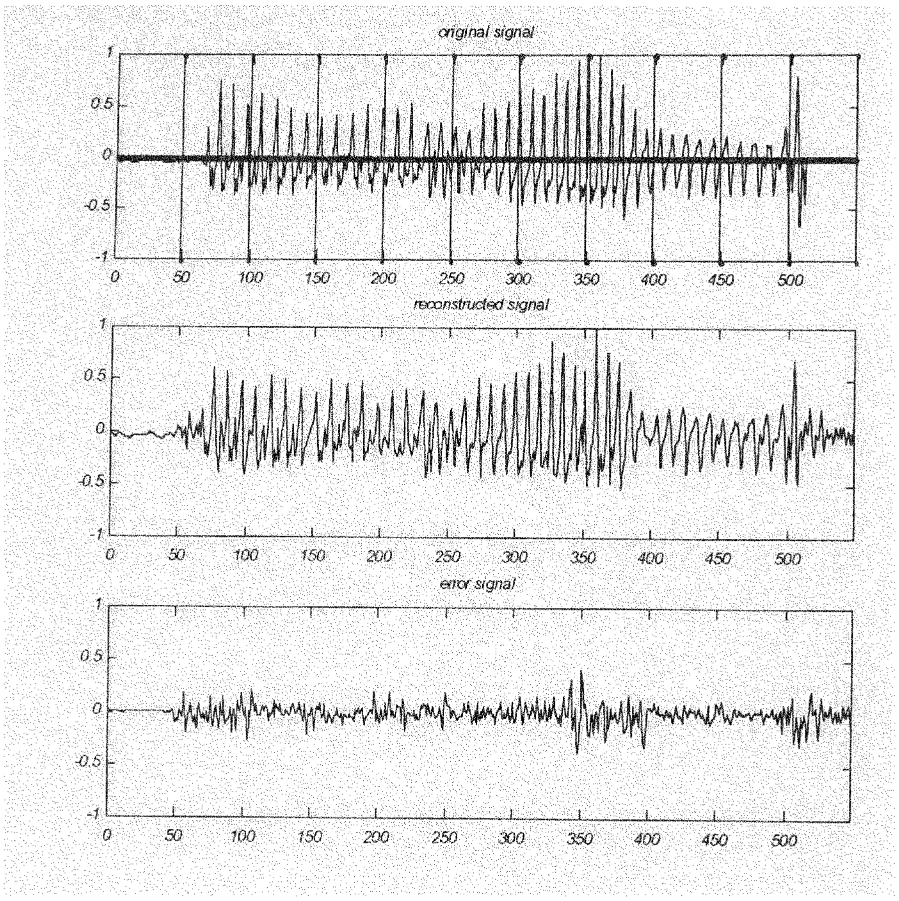

Figure 5.22 Example 4 of lossy compression of x_3

$N_U = 50$

$SNR_{MSE} = 9.1976$ dB

$SNR_{MAE} = 9.4733$ dB

$R = 4.1667$

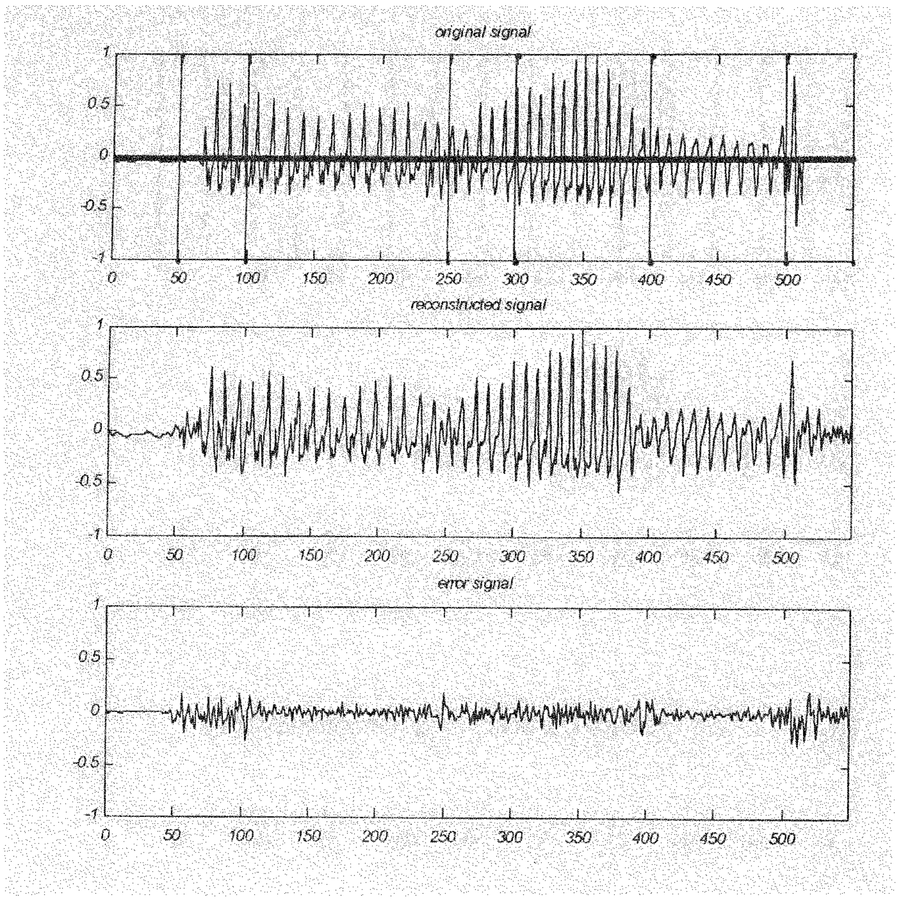

Figure 5.23 Example 5 of lossy compression of x_3

$N_U = 50$

$SNR_{MSE} = 11.4635$ dB

$SNR_{MAE} = 11.5543$ dB

$R = 4.1667$

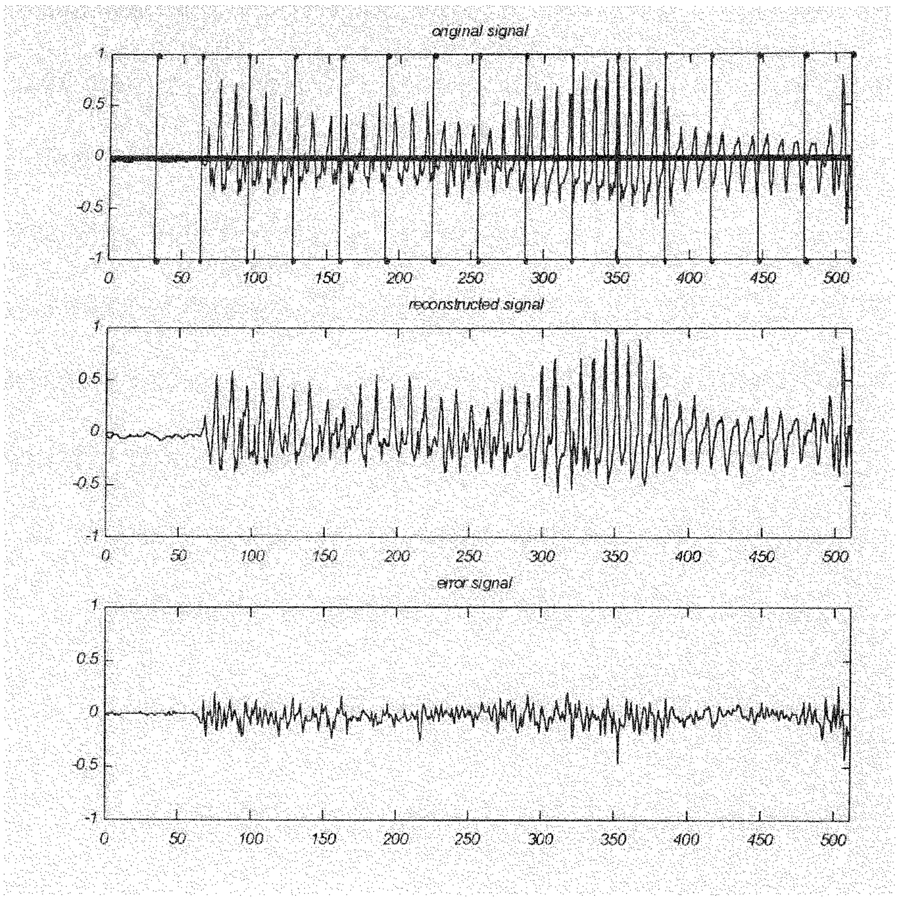

Figure 5.24 Example 6 of lossy compression of x_3

$N_U = 32$

$SNR_{MSE} = 9.7965$ dB

$SNR_{MAE} = 10.0091$ dB

$R = 4$

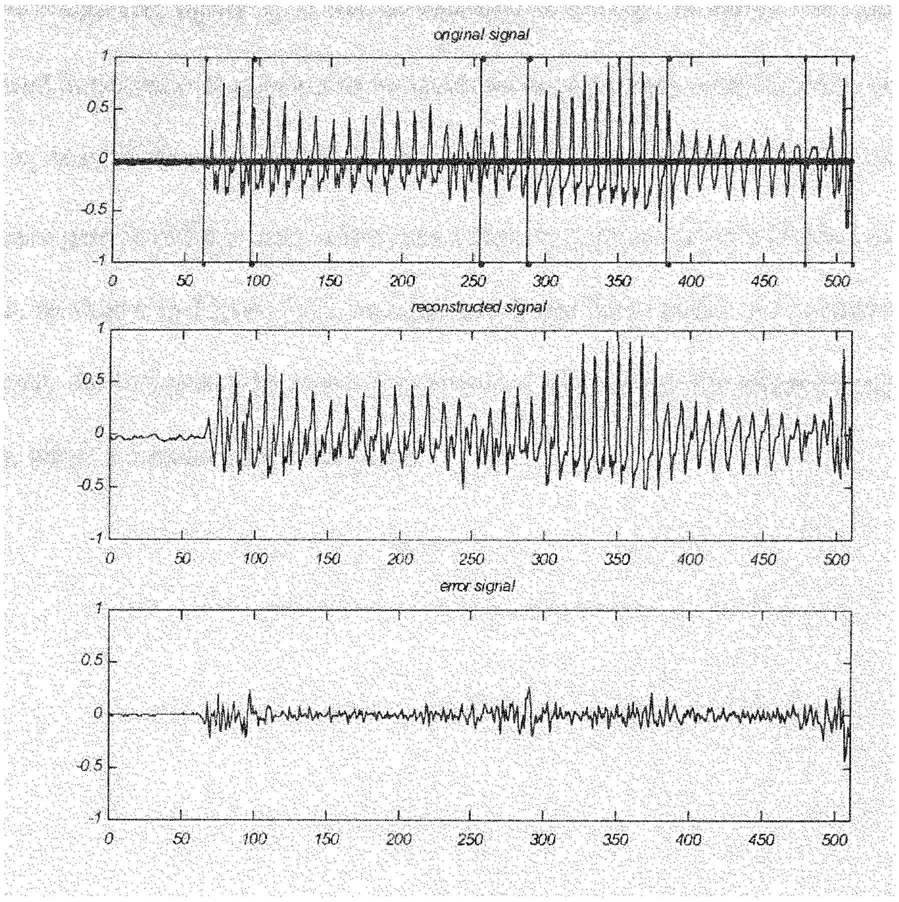

Figure 5.25 Example 7 of lossy compression of x_3

$N_U = 32$

$SNR_{MSE} = 11.1653$ dB

$SNR_{MAE} = 12.1112$ dB

$R = 4$

Although test signal x_3 is also an example of the type of signal that might be encountered in practice, it appears to be much more stationary over its entire support, whereas x_2 was relatively non-stationary, varying widely over time. For this reason, the performance gain is slightly less; in fact, the adaptive segmentation with the coarsest resolution, as shown in Figure 5.21, actually offers the best results. As in each previous case, though, there is generally some improvement when using the adaptive segmenting algorithm, albeit a smaller one in this case.

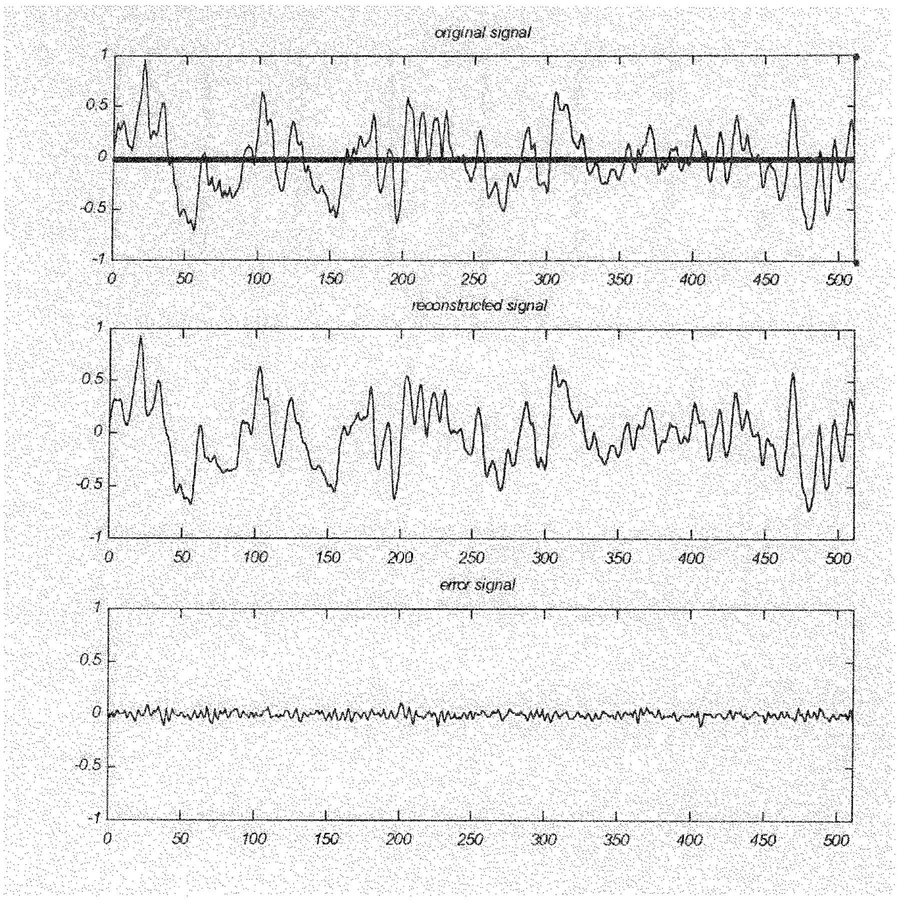

Figure 5.26 Example 1 of lossy compression of x_4

$N_U = 64$

$SNR_{MSE} = 19.0081$ dB

$SNR_{MAE} = 19.0474$ dB

$R = 4$

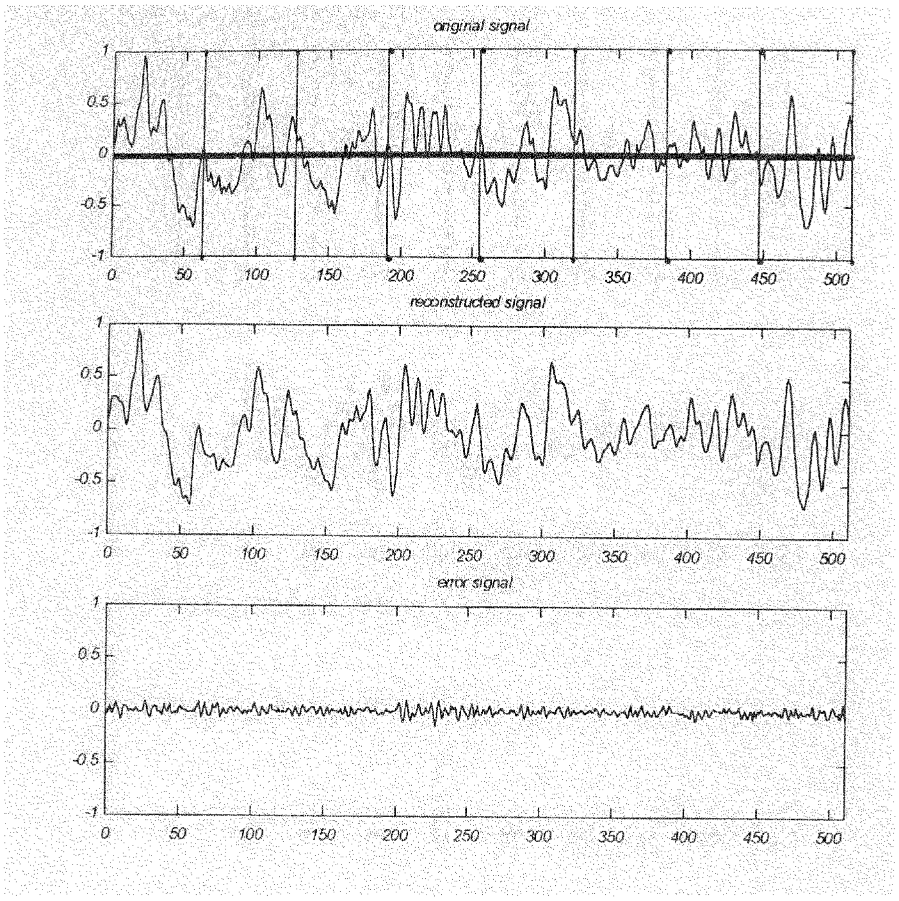

Figure 5.27 Example 2 of lossy compression of x_4

$N_U = 64$

$SNR_{MSE} = 18.5343$ dB

$SNR_{MAE} = 18.9459$ dB

$R = 4$

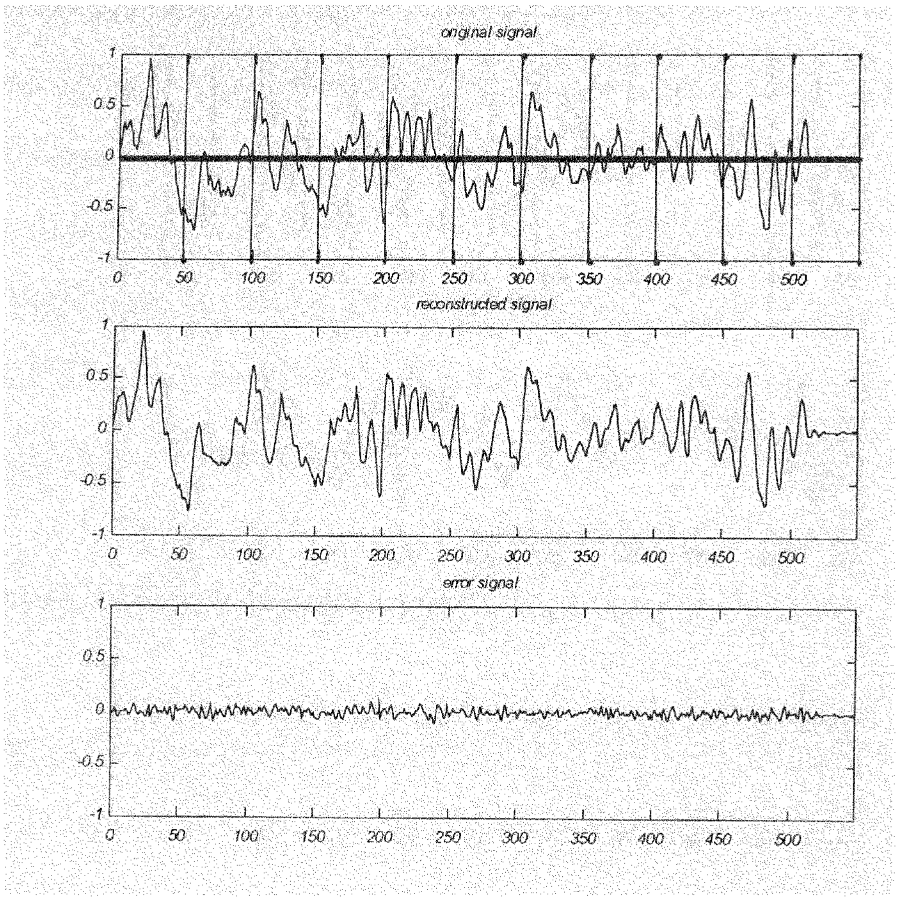

Figure 5.28 Example 3 of lossy compression of x_4

$N_U = 50$

$SNR_{MSE} = 18.3155$ dB

$SNR_{MAE} = 18.2627$ dB

$R = 4.1667$

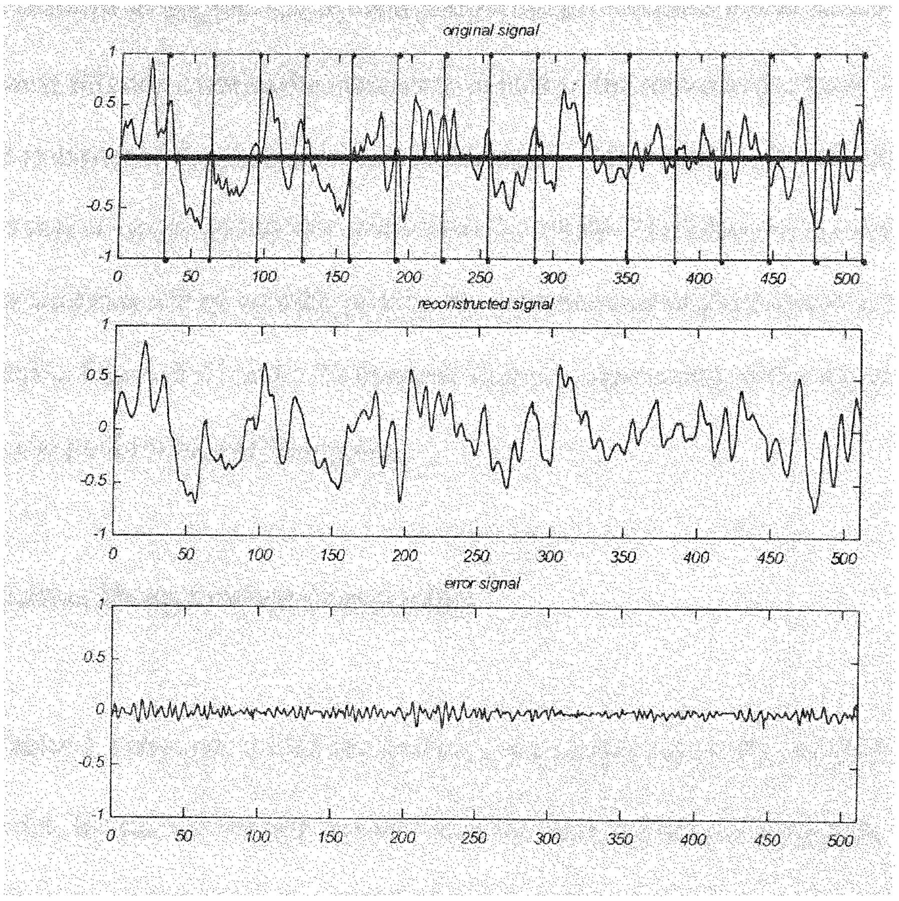

Figure 5.29 Example 4 of lossy compression of x_4

$N_U = 32$

$SNR_{MSE} = 17.3213$ dB

$SNR_{MAE} = 17.487$ dB

$R = 4$

As stated in Section 5.2, x_4 was chosen not just because it was non-stationary, but also because this non-stationarity appears to extend to the microscopic level. In other words, it is expected that no particular segmentation of this signal will yield more segments that are more locally stationary than the whole. The adaptive segmenting algorithm confirms this by actually picking the unsegmented original signal as the optimal segmentation. Figure 5.27 and 5.29 illustrate that any segmenting of the signal actually degrades the performance of the system.

5.4 Evaluation Using Lossless Compression

Tables 5.1 through 5.4 list the lossless compression ratios R_H, as given in Equation 5.4, for each of the test signals. Each has been segmented adaptively using the same parameters as in Section 5.3, and is then compared with a uniform segmentation and no segmenting whatsoever.

type of segmentation	compression ratio R_H
adaptive	3.4429
uniform	3.1429
none	2.5375

Table 5.1 Results for lossless compression of x_1

type of segmentation	compression ratio R_H
adaptive	2.7018
uniform	2.5347
none	1.9194

Table 5.2 Results for lossless compression of x_2

type of segmentation	compression ratio R_H
adaptive	2.0338
uniform	1.5182
none	1.6280

Table 5.3 Results for lossless compression of x_3

type of segmentation	compression ratio R_H
adaptive	2.7198
uniform	2.2141
none	2.7198

Table 5.4 Results for lossless compression of x_4

Examining the results in the lossless case reveals that in some cases, the performance gain is relatively minimal, nevertheless, there is still a gain. In the case of x_1, the adaptive segmentation yields an improvement of approximately 36% over no segmentation and

10% over uniform segmentation. In the case of x_2, the improvement over no segmentation is 41%, but there is only a 7% gain over uniform segmentation. For x_3, in which the signal is already relatively stationary over its entire support, the gain over no segmentation was only 25%, however, the gain over uniform segmentation was 34%. Finally for x_4, the adaptive segmenting algorithm picked no segmentation as the best segmentation, thus there is a 0% gain over no segmentation, but there is a 23% gain over uniform segmentation. In particular, this last result illustrates that the arbitrary uniform segmentation incorporated by many existing techniques may sometimes reduce, instead of improve, the performance of the system.

Chapter 6: Conclusions and Recommendations

6.1 Conclusion

In conclusion, it has been established that by using the adaptive segmenting process, data compression performance can be improved. In the case of lossy compression, it was shown that at a fixed compression ratio R, the signal-to-noise ratio can often be doubled or made even greater when adaptive segmenting was used. Inserted into another larger data compression scheme, it is quite reasonable to expect that adaptive segmenting can yield even greater improvement. Thus for a fixed R, more accurate representation can be achieved using adaptive segmenting, or for a fixed signal-to-noise ratio or other measure of reconstruction error, a increase in R can be attained.

For lossless compression, e.g. those employing entropy-based schemes, a significant reduction was achieved in the Huffman length of transform sequences by using adaptive segmenting in place of uniform segmenting. For some of the test signals, this meant an increase in the compression ratio of as much as 36%. Again, only the general case was presented, and it seems quite feasible that even more substantive improvement could be made within the context of a specific algorithm.

6.2 Recommendations for Further Study

The adaptive segmenting process was introduced as a general framework, so an obvious extension of the methodology would be towards more application-specific areas. The type of transform used could be modified for a more specific class of signals, the cost function adapted more closely to the exact coding technique, and so on. The best-basis paradigm of Coifman, et al. [17] could be applied to the transform choice, such that an appropriate basis is chosen for each segmentation or even each segment.

An additional area of refinement involves finding ways to reduce the computational costs of the search mechanisms and the cost functions. The dynamic programming approach employed in Chapter 4 to reduce the computational cost of an exhaustive search can be taken further by employing, for example, a parallel search algorithm, or one of the many other variations on this theme. Consider, for instance, a routine in which the dynamic programming search begins from both ends of the signal simultaneously. For the case of the cost functions themselves, methods for estimating the autocovariance matrices and other necessary statistics could be substituted for the explicit and costly calculations now in place.

It is also to important to note that the utility of the adaptive segmenting algorithm is not confined to the area of data compression alone. Time-frequency analysis, and general analysis of non-stationary signals, could also make use of this technique. This area has advanced from the Fourier transform and its complete lack of local time-domain information, to the uniform segmenting of the short-time Fourier transform, to the partial flexibility of the aforementioned local trigonometric transforms, and their frequency domain dual, wavelet packets. At the pinnacle of their flexibility, however, all these methods are constrained to the binary tree form discussed in Section 1.3. The approach presented in this thesis could possibly serve to improve the accuracy and performance of these methods, notably in the area of shift-invariance.

WORKS CITED

[1] Y. K. Lee, K. C. Kim and H. S. Lee, "An Efficient Coding of LSP Parameters Using Multiple Type Frame Segmentation," Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing 2 (1996): 753-756.

[2] Deepen Sinha and James D. Johnston, "Audio Compression at Low Bit Rates Using a Signal Adaptive Switched Filterbank," Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing 2 (1996): 1053-1056.

[3] John Watkinson, Compression in Video and Audio (Oxford, UK: Focal Butterworth-Heinemann, 1995) 115-117.

[4] Dana H. Brooks, et al., "Best Basis Segmentation of ECG Signals Using Novel Optimality Criteria," Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing 5 (1996): 2750-2753.

[5] Zixiang Xiong, et al., "Flexible Tree-Structured Signal Expansions Using Time Varying Wavelet Packets," IEEE Transactions on Signal Processing 45 (1997): 333-345.

[6] Ronald R. Coifman, et al., "Signal Processing and Compression with Wavelet Packets," Numerical Algorithms Research Group, Yale University (1990).

[7] Henrique S. Malvar, Signal Processing with Lapped Transforms (Boston: Artech, 1992).

[8] Ken C. Pohlmann, Principles of Digital Audio 3rd ed. (New York: McGraw-Hill, 1995) 429-436.

[9] N. S. Jayant and Peter Noll, Digital Coding of Waveforms: Principles and Applications to Speech and Video (Englewood Cliffs: Prentice-Hall, 1984) 535 546.

[10] K. R. Rao and P. Yip, Discrete Cosine Transform: Algorithms, Advantages, and Applications (Boston: Academic, 1990).

[11] Allen Gersho and Robert M. Gray, Vector Quantization and Signal Compression (Boston: Kluwer, 1992).

[12] Brani Vidakovic and Peter Müller, "Wavelets for Kids: A Tutorial Introduction," (1991) 16-17.

[13] Richard W. Hamming, Coding and Information Theory (Englewood Cliffs: Prentice-Hall, 1980) 112-113.

[14] David A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," Proceedings of the Institute of Radio Engineers 40 (1952): 1098-1101.

[15] Massih Hamidi and Judea Pearl, "Comparison of the Cosine and Fourier Transforms of Markov-1 Signals," IEEE Transactions on Acoustics, Speech, and Signal Processing (October 1976) 428-429.

[16] Hideo Kitajima, "Energy Packing Efficiency of the Hadamard Transform," IEEE Transactions on Communications November 1976 1256-1258.

[17] Ronald R. Coifman and Mladen Victor Wickerhauser, "Entropy-based Algorithms for Best Basis Selection," Numerical Algorithms Research Group, Yale University.