DISTRIBUTED AUDIO CODING WITH EFFICIENT CORRELATION EXTRACTION

BY

SANDEEP KUMAR MATTA, B.E.

A thesis submitted to Graduate School

in partial fulfillment of the requirements

for the degree

Master of Science

Major Subject: Electrical and Computer Engineering

New Mexico State University

Las Cruces New Mexico

August 2008

"Distributed Audio Coding with Efficient Correlation Extraction," a thesis prepared by Sandeep Kumar Matta in partial fulfillment of the requirements for the degree, Master of Science, has been approved and accepted by the following:

Dr. Linda Lacey

Dean of the Graduate School

Dr. Charles D. Creusere

Chair of the Examining Committee

Date

Committee in Charge:

Dr. Charles D. Creusere, Chair

Dr. Deva K. Borah

Dr. Joe Lakey

DEDICATION

I dedicate this work to my family and my advisor Charles D. Creusere.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr.Charles D. Creusere, for his support, encouragement and knowledge. Also, I would like to thank NMSU professors who taught me all the basics and helped me in completing this thesis. I also thank Vijendra Raj Apsingekar for guidance. Last but not least, I am very much thankful to Dr. Srivatsavan kandadai for his continuous guidance on TWIN-VQ.

VITA

August 12th, 1985	Born at Godavarikhani, A.P, India
2002-2006	B.E., Osmania Universtiy, Hyderabad, India
2006-2008	M.S., New Mexico State University, Las Cruces, New Mexico

PUBLICATIONS

[1] C. D. Creusere and Sandeep K. Matta, " Efficient correlation extraction for Distributed Audio coding" Accepted for 42nd Asilomar Conference on Signals and Systems, 2008.

[2] C. D. Creusere and Sandeep K. Matta, " Distrubuted Audio Coding with Efficient Correlation Extraction" Submitted to 13th DSP/SPE workshop, 2009.

Field of Study

Major Field: Electrical Engineering

Digital Signal Processing

ABSTRACT

DISTRIBUTED AUDIO CODING WITH EFFICIENT CORRELATION EXTRACTION

BY

SANDEEP KUMAR MATTA, B.E

Master of Science in Electrical Engineering

New Mexico State University

Las Cruces, New Mexico, 2008

Dr. Charles D. Creusere, Chair

Wireless sensor networks form the backbone of many surveillance applications. Distributed source coding is one of the enabling technologies for efficient bandwidth utilization in wireless sensor networks and is consequently of great current interest. This thesis studies its application to audio signals, using a transform weighted interleaved vector quantization (TWIN-VQ) framework and allowing sensor nodes to passively receive and use information from neighboring sensors that is being transmitted to the common joint decoder. Specifically, we use the linear predictor coefficients generated as side information by TWIN-VQ for one source to determine its frame-by-frame correlations with another source. We, then develop a low

complexity algorithm to conditionally encode the transform coefficients of one source given the transform coefficients of another source. Specifically, we use a context based adaptive binary arithmetic coder to losslessly compress the final flattened transform coefficients from TWIN-VQ.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Wireless sensor networks are the keystone for many audio and video surveillance applications. In the sensor networks, sensors are generally deployed densely in an unattended manner. One of the major constraints at each sensor node is its power. In order to reduce the power consumption and computational complexity, these sensor nodes are often not allowed to communicate with each other. Instead, these sensor nodes send their compressed outputs to common decoder for joint decoding [16]. This is the classical Distributed source coding (DSC) approach. In a wireless sensor network, outputs of the adjacent sensor nodes are almost always correlated. Distributed source coding is considered as one of the efficient encoding techniques, since it exploits this correlation in a computationally efficient manner. As there is a tradeoff between the bit rate and power consumption at the sensor node, decreasing the bit rate of sensor's output could give savings in power consumption.

In a classical Distributed Source coding (DSC) paradigm, nodes are precluded to communicate amongst themselves and the decoder must extract the correlated frames without any assistance from the encoders. The results of Slepian and Wolf and Wyner and Ziff show us how such correlations can be exploited to reduce transmission rates for lossless and lossy coding systems respectively [17],[18]. These algorithms coupled with the recent interest in Distributed sensor networks have lead to a resurgence interest in this research area. The specific application of these research concepts to audio signals is what we term Distributed Audio coding (DAC). The main

objective of the distributed audio coding is to decrease the computational complexity, bandwidth requirements and power consumption required to communicate audio signals acquired by a distributed sensor array.

To create a practical implementation of a distributed audio coding system, we break the basic rule of DSC and allow nodes to 'overhear' the transmissions being sent from other nodes to base station where the joint decoding is performed—i.e. nodes are allowed to passively receive the information from nearby nodes. Thus, a degree of communication is allowed between sensor nodes. With this passively received information (sensed signal from adjacent sensor node) we estimate the inter-sensor correlations and thus adapt to changing operating conditions. One of the major difficulties with extracting inter-sensor correlation is encoder complexity; specifically, the sensor node has to spend power and computational bandwidth to acquire the signal from nearby sensor. Generally for the sensor to listen nearby node, it has to receive the RF signal and then demodulate the received signal, spending the power in the process. Next, the sensor has to decode the sensed signal to reconstruct the audio frames. Once decoded, these frames have to be compared with the local frames to determine any correlation. Finally the correlated frames have to be conditionally encoded which again requires extra computational steps.

Here we explore portions of the above stated problem, namely the efficient correlation estimation of the sensor outputs and encoding one output with respect to the other. Specifically, we determine the correlation between the compressed outputs

of sensor nodes with the help of the correlation between side information. This approach has many advantages, the most significant one being that the need to decode the sensed signal from nearby node is eliminated and moreover that less information is processed for correlation extraction, thereby reducing the power consumption and computational complexity.

Let us consider the problem of classical distributed compression of sensor outputs where the decoder has to reconstruct the original source signal from the side information and bit stream of the other source nearby the original source. To be more precise, consider the two discrete sources *X* and *Y*. From Shannon's source coding theorem [4] one can encode the source *X* at the minimum of *H(x)* bits/sample where *H(x)* is the entropy of the source *X*. When *X* and *Y* are correlated, discrete –alphabet, independent and identically distributed (i.i.d) sources, we still can encode at the conditional entropy rate given by *H(x/y)* [20] (where *H(x/y)* is the uncertainty remained in *X* given *Y*) with classical DSC approach where the encoders of *X* and *Y* do not communicate. In this case, the encoder of *X* does not know *Y*. This is the Slepian-Wolf coding theorem [17]. As we have state earlier, Wyner and Ziv have extended this to lossy case where they have assumed *X* and *Y* as correlated i.i.d Gaussian random variables [18].

The case with which we are dealing is the simplest one since we have access to *Y* (the sensed signal) at the encoder of *X*. Hence, we can conditionally encode the source *X* given the signal *Y*; specifically, we extract here the correlation between sources *X* and

Y with the help of side information from X and Y, respectively. Consider the outputs of two different sensor nodes X and Y with a simple correlation structure:

$$Y = A*X + N \tag{1.1}$$

where A is an attenuation constant, N is the continuous zero mean Gaussian noise. X and Y are continuous independent and i.i.d. In other words, the sensed signal can be realized as a gain adjusted and corrupted version of the locally captured signal X. Note more general correlation structures are given in [4] while Pradhan *et al.* adopt a simplified variation in [5], assuming that correlation $Y = X + N$ where Y is a corrupted version of X; X, Y , & N are continuous and *i.i.d*.

Our implementation of a distributed audio coding system is based on the TWIN-VQ algorithm as described in the MPEG-4 standards document [1] as well as in [2] and [3], and the following chapter gives the complete details of TWIN-VQ coder. In this thesis, all the experiments are performed on two sources X and Y. Specifically, X is considered as the original audio signal and Y is the attenuated and corrupted version of X as stated in equation (1.1).

2. TRANSFORM DOMAIN WEIGHTED INTERLEAVED VECTOR QUANTIZATION

2.1 INTRODUCTION.

Transform domain weighted interleaved vector quantization (TWIN-VQ) is one of the audio compression algorithms specified in MPEG 4 standards document [1]. This audio compression algorithm is found to be most effective at very low bit rates [2] [3] and its basic structure is shown in Figure 2.1.

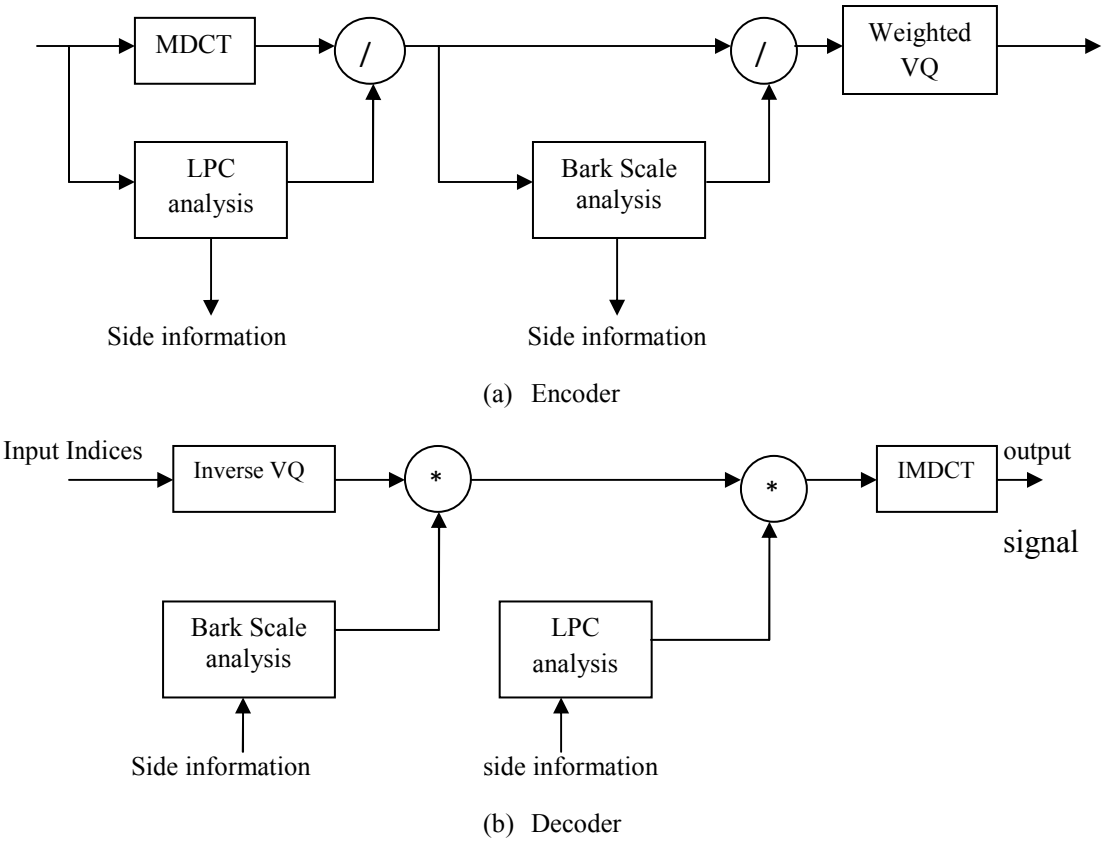

Figure 2.1: Basic Structure of TWIN-VQ (a) Encoder (b) Decoder

In TWIN-VQ algorithm, first the input signal is divided into time frames of 1024 samples. These input audio frames take two paths as shown in Figure 2.1 [5]. Along the first path, a 1024-point MDCT (modified discrete cosine transform) is directly applied to single frames of the audio signal and along the other path a linear predictive filtering is done. The former path transforms the input audio signal into the frequency domain while the latter calculates the linear predictive spectral envelope which gives the rough representation of the spectrum. This spectral envelope is used to normalize the MDCT coefficients through division as shown in the figure. This process is equivalent to performing linear predictive filtering in time domain and removing the predictable portions of the signal. Audio signal generally contain a lot of harmonic structure, so the first stage flattening does not completely flatten the MDCT spectrum. Using Bark scale analysis, the rough shape of the spectrum is estimated and this spectrum is used to further normalize the MDCT coefficients, improving the quantization performance of the algorithm. Finally, a two channel conjugate weighted vector quantization is applied on the flattened MDCT coefficients and the indices of the final quantized coefficients are transmitted to the decoder. The quantized linear prediction coefficients and Bark scale coefficients are also transmitted to the decoder as the side information.

The decoder as shown in Figure 2.1(b) inverts the operations done by the encoder. It reconstructs the quantized and flattened MDCT coefficients from the VQ indices and the reconstructed Bark scale and linear predictive coefficients are multiplied

respectively with the reconstructed spectrum to reconstruct a quantized version of original spectrum. An inverse MDCT (IMDCT) is then applied to this quantized original spectrum to reproduce the time domain signal.

The above introduction gives the brief outline of TWIN-VQ. Now let us discuss the block in more detail.

2.2 MODIFIED DISCRETE COSINE TRANSFORM (MDCT)

Modified Discrete Cosine Transform is derived from Discrete Cosine Transform (DCT). DCT is an efficient block-based transform coding technique, but the main disadvantage with DCT is that it creates artifacts at the block boundaries at low bit rates [4]. One of the popular overlapping block based technique which eliminates such artifacts and has gained wide importance in audio compression algorithms is the Modified Discrete Cosine Transform (MDCT) [5].

When we transform the signal using a Discrete Fourier Transform (DFT), a small overlap between the adjacent blocks increases the data rate of the transformed coefficients. With 50% of overlap between blocks, the data rate would be doubled prior to performing any quantization. An overlapping transform which overcomes this problem with the advantage of removing blocking artifacts is MDCT.

Cosine modulated pseudo quadrature mirror filter banks are mainly used for M channel filter banks for subband decomposition with almost perfect reconstruction properties [5]. These pseudo quadrature mirror filter (PQMF) banks uses bandpass

filters and have been applied to perceptual audio coding in such algorithms as MPEG audio layer 3, more widely known as MP3. As these bandpass filters are orthogonal, the synthesis filters are just the time reverse of the analysis filters hence forcing the mirror image condition—i.e.

$$g_k[n] = h_k[L-1-n], \tag{2.1}$$

where $h_k[n]$ is the impulse response of k-th analysis filter and $g_k[n]$ is the impulse response of the corresponding synthesis filter.

To cancel the aliasing in the neighboring subbands, low pass prototype filter in PQMF is subjected to the following constraint.

$$|W(e^{jw})|^2 + |W\left(e^{\left(j\left(\frac{\pi}{N}\right)-w\right)}\right)|^2 = 2, \quad 0 < |w| < \frac{\pi}{2N} \tag{2.2}$$

$$\left|W(e^{jw})\right|^2 = 0, \quad |w| > \frac{\pi}{N} \tag{2.3}$$

where $W(e^{jw})$ is the frequency response of the lowpass prototype filter used to design the PQMF. These two equations lead to a precise frequency domain relationship between the analysis and synthesis filters, $H_k(z)\ and\ G_k(z)$. The filter bank analysis filters are the cosine modulations of $w[n]$ given by

$$h_k[n] = 2w[n]\cos\left(\frac{\pi}{M}(k+0.5)\left(n-\frac{(L-1)}{2}\right)+\theta_k\right) \tag{2.4}$$

and synthesis filters are given by

$$g_k[n] = 2w[n]\cos\left(\frac{\pi}{M}(k+0.5)\left(n-\frac{(L-1)}{2}\right)-\theta_k\right) \tag{2.5}$$

where the phase difference is given by

$$\theta_{k+1}-\theta_k = (2r+1)\frac{\pi}{2} \tag{2.6}$$

and $w[n]$ is a lowpass prototype window of length L with linear phase and finite impulse response. Here, our specific interest is MDCT which is given as cosine modulated filter bank with $L = 2M$ where L is the length of filter banks and M is the number of analysis filters. Therefore, the impulse response of the analysis filter for the MDCT is given by

$$h_k[n] = w[n]\sqrt{\frac{2}{M}}\cos\left(\frac{(2n+M+1)(2K+1)\pi}{4M}\right). \tag{2.7}$$

Since the transform is orthogonal, the synthesis filters are given by the time reverse version of analysis filters, i.e.

$$g_k[n] = h_k[2M-1-n]\,. \tag{2.8}$$

The above equations (2.7) and (2.8) also give the impulse responses of analysis and synthesis filters, respectively, for the MDCT, but the MDCT used in perceptual audio coders is implemented typically in a block based manner.

The MDCT analysis filter bank is practically implemented using a block transform of size 2M samples where M is number of input samples and the output of MDCT gives

M samples with a 50% overlap between two input sample blocks. Thus, MDCT is a critically sampled transform. For $2M$ input samples of $x[n]$ the transformed coefficients $X[k]$ for $0 \leq k \leq M-1$ are given by

$$X[k] = \sum_{n=0}^{2M-1} x[n]h_k[n]. \tag{2.9}$$

The above equation performs the inner product between the input $x[n]$ and the filter impulse response $h_k[n]$. Unlike the DFT, the IMDCT uses two transformed blocks to recover a single time domain block of input signal. The first *M* samples of the k^{th} filter impulse response $h_k[n]$ for $0 \leq n \leq M-1$, are weighted by the k^{th} coefficient of current block, $X[k]$ at the same time the second *M* samples of the k^{th} impulse response for $M \leq n \leq 2M-1$ are weighted with the previous block frequency coefficients $X'[k]$. These weighted filter coefficients are overlapped and added to recover the single time domain block. These operations of overlapping and adding cancel the time domain aliasing which would normally occur due to critical sampling in transformed domain. Thus, the inverse MDCT is given by

$$x[n] = \sum_{k=0}^{M-1}(X[k]h_k[k] + X'[k]h_k[n+m]). \tag{2.10}$$

The above equation requires an *M* size buffer to store the previous set of transformed coefficients. Recall that the impulse responses of the analysis and synthesis filters are derived from the lowpass prototype filter $w[n]$. The most commonly used prototype filter or window function is

$$w[n] = \sin\left[(n + 0.5)\frac{\pi}{2M}\right] \tag{2.11}$$

for $0 \leq n \leq 2M - 1$. This window function has the property of perfect reconstruction.

2.3 LINEAR PREDICTIVE CODING

Linear predictive analysis is carried out on the input audio signal to coarsely estimate its spectrum. TWIN-VQ uses the LPC spectrum since the Line spectral pairs (LSP) calculated from LPC are easy to quantize and send to the decoder as the side information. The decoder simply reconstructs the rough spectra from LSP parameters and the reconstructed spectrum is used to denormalize the flattened MDCT coefficients.

In general, a linear predictive coder can be modeled as an Auto Regressive Moving Average (ARMA) model which is given for any input sample *x[n]* as

$$x[n] = \sum_{k=1}^{p} a_k x(n-i) + G \sum_{l=0}^{q} b_l u(n-l) \tag{2.12}$$

where the input sample $x[n]$ is represented as the sum of scaled version of p past samples and scaled version of q excitation samples. For (2.12), G is the Gain factor for the input samples, a_k and b_l are the filter coefficients. Equivalently, the filter transfer function for the equation (2.12) in z- domain is

$$H(z) = G\ \frac{1 + \sum_{l=1}^{q} b_l z^{-1}}{1 - \sum_{k=1}^{p} a_k z^{-1}} \tag{2.13}$$

where *H(z)* is the pole-zero model in which the polynomial roots of numerator and denominator are the zeros and poles of the system, respectively.

When a_k= 0 for $1 \leq k \leq p$ the system *H(z)* is termed as all-zero or moving average system where the output $x[n]$ is given as the weighted value of p prior inputs. Conversely when $b_l = 0\ for\ 1 \leq l \leq q$, *H(z)* reduces to all-pole or Auto regressive (AR) system since its moving average (MA) term is zero. In this case the transfer function is given as

$$H(z) = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-1}} = \frac{1}{A(z)}. \tag{2.14}$$

TWIN-VQ uses an all-pole model to estimate the LPC spectrum. Since the spectrum is itself estimated from linear prediction coefficients, it is vital to accurately estimate the prediction coefficients. There are two methods for estimating the Linear Prediction Coefficients:

1. Autocorrelation method,
2. Covariance method.

TWIN-VQ uses the autocorrelation method. In both of the methods, a least squares technique is used to minimize the residual error *E* with respect to coefficients a_k as given by

$$E = \sum_{n=-\infty}^{\infty} e(n)^2 = \sum_{n=-\infty}^{\infty} (x(n) - \sum a_k x(n-k))^2 \ . \qquad (2.15)$$

The LP coefficients which minimize E in (2.15) are obtained by partial derivation of E with respect to each of the prediction coefficient a_k and equating them to zero. Thus, we get the following p equations with p unknown variables a_k:

$$\sum_{k=1}^{p} a_k \sum_{n=-\infty}^{\infty} x(n-i)x(n-k) = \sum_{n=-\infty}^{\infty} x(n-i)x(n)\,, \ \ 1 \leq \ i \leq p. \qquad (2.16)$$

As we consider an input signal $x[n]$ of finite length, we can introduce an autocorrelation function for the above equation, specifically

$$R(i) = \sum_{i=0}^{N-1} x(n)x(n-i)\,, \qquad 0 \leq i \leq p. \qquad (2.17)$$

applying (2.17),equation (2.16) becomes

$$\sum_{k=1}^{p} R(|i-k|)a_k = R(i), \qquad 1 \leq i \leq p\,. \qquad (2.18)$$

The above set of linear equations can be expressed concisely in matrix form as

$$\begin{bmatrix} R(0) & \cdots & R(p-1) \\ \vdots & \ddots & \vdots \\ R(p-1) & \cdots & R(0) \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} R(1) \\ \vdots \\ R(p) \end{bmatrix} \qquad (2.19)$$

or $$Ra = r. \qquad (2.20)$$

The resulting matrix *R* is an autocorrelation matrix and is therefore a symmetrical Toeplitz matrix where all the elements along its main and sub diagonals are equal. The equation (2.20) can be solved using Levinson-Durbin algorithm.

In TWIN-VQ the linear prediction coefficients calculated directly from Levinson-Durbin algorithm are not used to estimate the spectrum; instead 41 cepstral coefficients are calculated from LP coefficients. The LPC coefficients are then recalculated for cepstral coefficients. Sklar states that the LPC coefficients calculated from cepstrum are always better than the LPC coefficients directly calculated through autocorrelation method.

In the final step, line spectral pairs are calculated from the LSP polynomials. These LSP polynomials are obtained from the conventional LP polynomial *A(z)* given in (2.14).

2.4 BARK SCALE ANALYSIS:

Linear prediction does not completely flatten the MDCT coefficients as shown in Figure 2.2 as where we observe some remaining undulations in the transform coefficients. In some cases, if the input audio contains lot of harmonic structure, the

amplitude of the MDCT coefficients after the first stage flattening may be increased, thereby reducing the quantization efficiency [6]. So a second stage flattening is necessary to decrease the overall amplitude range of MDCT coefficients. To further flatten the MDCT coefficients, i.e. to reduce the overall amplitude range, a rough

shape of the spectrum is estimated through Bark scale analysis. The final flattened MDCT coefficients provide finer (having low range or variance) representation than previous MDCT coefficients [6]. Figure 2.3 shows reduction in the amplitude of peaks after normalization with Bark scale envelope.

.

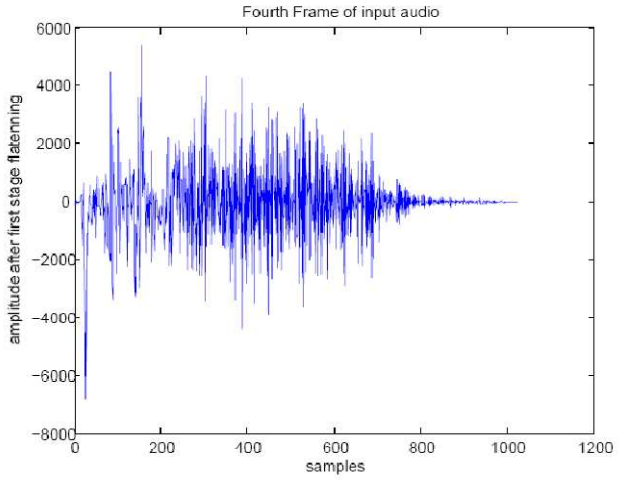

Figure 2.2: Transform Coefficients of one of the frame after first stage flattening

The Bark scale envelope is found such that it represents the fine undulations in the input audio. To find the Bark scale envelope, the MDCT coefficients after the first stage flattening are split into subbands based on bark scale and a scale factor for each subband is calculated. Bark scale is nothing but the first 24 critical bands of human hearing [6].

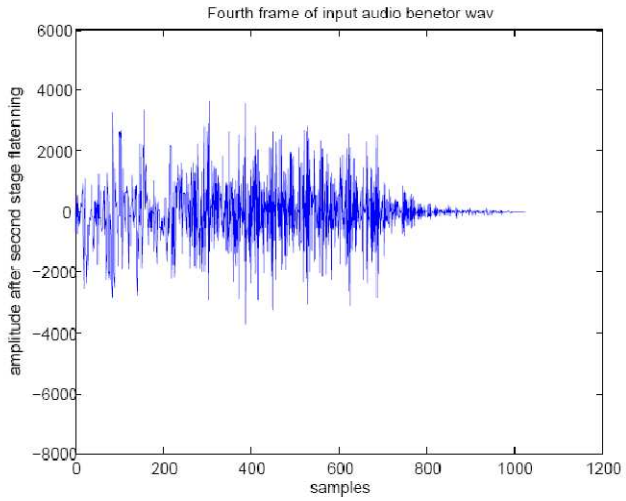

Figure 2.3: Transform coefficients for one of the frame after second stage flattening for benetor.wav

The procedure to calculate the Bark scale envelope is shown in Figure 2.4. First, the amplitude of each MDCT coefficient is calculated following the first stage flattening and the MDCT coefficients are transformed from linear to bark scale using the following equation

$$B = 13\tan^{-1}(0.76f) + 3.5\tan^{-1}(\frac{f}{7.5}) \quad (2.21)$$

where B is the Bark scale frequency and f is the frequency in linear scale. The following table shows the transformation table to subbands. The frequency axis is split into subbands corresponding to the Table 2.1 . The average value of MDCT coefficient in each subband is calculated and the sequence of amplitudes in each subband provides the bark scale envelope. Hence, the rough spectra of flattened MDCT coefficients are represented by fewer coefficients, and it is easier to transmit this spectrum to the decoder.

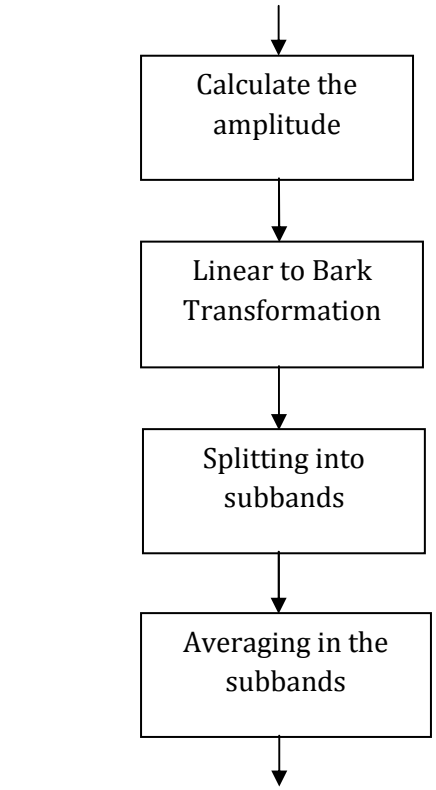

Figure 2.4: Calculation of Bark scale envelope

Table 2.1: Transformation table to subbands

Subbands	Linear frequency scale (htz)
1	3
2	8
3	13
4	18
5	24
6	30
7	36
8	43
9	51
10	60
11	71
12	83
13	99
14	119
15	145
16	180
17	228
18	298
19	406
20	596
21	1024

2.5 WEIGHTED VECTOR QUANTIZATION:

In many transform coders for speech [7], adaptive bit allocation is used to code the transformed coefficients and this bit allocation dictates the quantization errors so as to minimize the distortion of the reconstructed signal. This optimal bit allocation strategy is determined by the spectral envelope and this spectral envelope is efficiently encoded and sent to the decoder as side information [8]. The main disadvantage of adaptive bit allocation is that it is not resilient to channel errors—i.e. any error in the side information forces the decoder to erroneously decode bitstream. Since adaptive bit allocation is equivalent to weighted vector quantization (WVQ) [8], we can have more control over the distribution of quantization error through weighted euclidean distance measure in transformed domain without sacrificing performance for noisy channels.

We now discuss the quantization of the LPC residue in the transformed domain. A linear predictive coder can be realized with an FIR filter having filter coefficients given by $h^T = [h_1, h_2, \dots\dots, h_N]$. Considering an input signal **x**, quantization is performed by finding the best code vector c which minimizes the distortion **D** between the input signal **x** and the synthesized code vector **Hc**, as shown in Figure 2.5 where **H** is the impulse response matrix given by LPC filter coefficients:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{m-1} & h_{m-2} & \cdots & h_1 & 1 \end{bmatrix}.$$

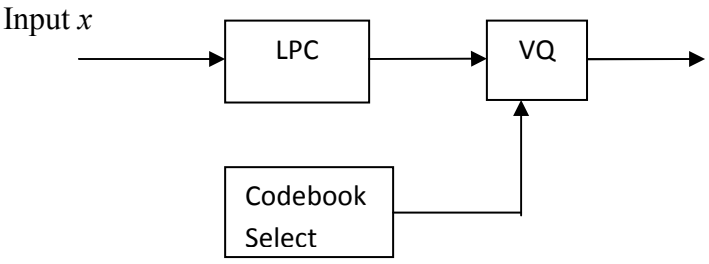

Figure 2.5: Quantization of LPC residue

If the Euclidean distance is used as a measure of similarity, then the distortion $\boldsymbol{D}$ is given by

$$\boldsymbol{D} = \|\mathbf{x} - \mathbf{Hc}\|^2. \tag{2.22}$$

Should the power of **x** is normalized so that the residual power is unity, **x** can be written as

$$\boldsymbol{x} = \boldsymbol{He} \tag{2.23}$$

where **e** is the LPC residual vector. Substituting (2.13) into (2.12) we get

$$\boldsymbol{D} = \left\|\mathbf{He} - \mathbf{Hc}\right\|^2 = (\boldsymbol{e}-\boldsymbol{c})^{\boldsymbol{T}}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H}(\boldsymbol{e}-\boldsymbol{c}) \tag{2.24}$$

where *H* is the factor of correlation matrix *R* and *R* is obtained by the spectral factorization of *E*(**xx'**).

Since R is a positive definite matrix, there exists a matrix U formed by the eigen vectors of R which can diagonalize the matrix R. Let the diagonal matrix be Λ.

$$\mathbf{\Lambda} = \mathbf{U^T R U} = \text{diag}[\lambda_1, \ldots\ldots\ldots \lambda_n] \tag{2.25}$$

where λ_i represents the nonnegative eigen values of R. It also represents the variance of each random variable in $\mathbf{x}$. Since U is a Unitary matrix, R can be written as

$$\boldsymbol{R = H^T H = U \Lambda U^T}. \tag{2.26}$$

Substituting (2.26) into (2.24), the distortion D is given by

$$D = (\boldsymbol{e - c})^{\boldsymbol{T}}\, \boldsymbol{U \Lambda U^T (e - c)} = \sum_{i=1}^{m} \lambda_i\, (E_i - C_i)^2. \tag{2.27}$$

The above equation shows that the quantization error is the weighted sum of Euclidean distance between transformed residual vector E_i and the transformed code vector C_i. The above transformation is regarded as Karhunen-Loeve transfo rmation for an auto regressive source. Thus, the quantization error in transformed domain equation (2.27) is same as the quantization error in time domain (2.22).

Let us quantify the quantization gain achieved by the weighted vector quantization. The quantization gain due to WVQ is same as the quantization gain for the adaptive bit allocation. Consider the source $\boldsymbol{x} = [x[1], x[2], \ldots\ldots\ldots, x[m]]^T$ to be an m-dimensional uniformly distributed vector with variances $\lambda_1, \lambda_2, \ldots\ldots \lambda_m$. Also consider

the two codebooks, C_u and C_w, one for uniform vector quantization and the other for weighted vector quantization with both having same number of codewords and operating at the same bit rate of $\bar{b}$. For the uniformly distributed vector $\boldsymbol{x}$, if each component x_i forms a hyper cube of edge length L_i so all of the VQ bins form the hyper cube of edge length $L_i/\bar{N}$ where $\bar{N} = 2^{\bar{b}}$ and the corresponding centroid is the codeword c_u. The average distortion for this case is given as

$$d_u = \int_0^{\frac{L_1}{\bar{N}}} \dots \int_0^{\frac{L_m}{\bar{N}}} \sum_{i=0}^{m} (x_i - c_{ui})^2 \, dx_1 \dots dx_2. \tag{2.28}$$

Equation (2.28) is nothing but the variance of each VQ bin normalized by its total volume. The average distortion in terms of individual variances λ_i can be written as

$$d_u = k \sum_{i=1}^{m} \lambda_i \tag{2.29}$$

and

$$k = \frac{1}{3} \prod_{i=1}^{m} {\lambda_i}^{1/2} \tag{2.30}$$

In designing the codebook C_w for weighted vector quantization, the procedure is the same. Here we use C_w, the lattice of uniform scalar quantizers as mentioned above, but with different number of bits b_i per dimensional component x_i. The optimal value for b_i is given in [4] as

$$b_i = \bar{b} + \frac{1}{2} \log_2 \frac{\lambda_i}{(\prod_{i=1}^{m} \lambda_i^{\frac{1}{m}})} \tag{2.31}$$

and the number of quantization levels for i^{th} variable is $N_i = 2^{b_i}$. Assuming uniform quantization, the edge size of VQ bin is given by

$$\frac{L_i}{N_i} = 2^{-\bar{b}} L_i \frac{\prod_{j=1}^{m} \lambda_j^{\frac{1}{2m}}}{\lambda_i^{\frac{1}{2}}}. \tag{2.32}$$

The distortion is no longer the simple Euclidean distance, so we use weighted Euclidean distance as a distortion measure instead. Thus, the distortion is given by

$$d_w = \int_0^{\frac{L_1}{N}} \ldots \int_0^{\frac{L_m}{N}} \sum_{i=0}^{m} \lambda_i (x_i - c_{wi})^2 \, dx_1 \ldots dx_2 \tag{2.33}$$

which on simplification yields

$$d_w = km \sum_{i=1}^{m} \lambda_i^{\frac{1}{m}}. \tag{2.34}$$

Dividing (2.29) by (2.34) yields the gain for the weighted vector quantization:

$$G = \frac{d_u}{d_w} = \frac{\sum_{i=1}^{m} \lambda_i}{m \sum_{i=1}^{m} \lambda_i^{\frac{1}{m}}}. \tag{2.35}$$

Equation (2.35) is also the quantization gain for adaptive bit allocation. Hence, the quantization gain is same for both adaptive bit allocation and weighted VQ.

2.6 INTERLEAVING

The above analysis shows the gain *G* obtained using WVQ. To improve the performance of VQ, it is vital to know the relationship between transform size *m* and the coding gain. As the correlation matrix *R* is a symmetrical Toeplitz matrix, the SNR gain is given as

$$SNG = 10log_{10}G. \tag{2.36}$$

The above equation can be explicitly expressed in terms of transform size *m* and PARCOR parameters k_j as [8] follows

$$SNG = -10log_{10}\left\{\prod_{i=1}^{P}(1 - k_i^2)^{(\frac{m-i}{m})}\right\} \tag{2.37}$$

Furthermore, it can easily be shown that the linear predictive gain equals the above SNR gain for the transform size of infinity. Equation (2.37) above gives the relationship between the transform size *m* and the SNR gain. It also holds for the time domain quantization procedure of (2.22) and can therefore be used to estimate the performance of coders with the vector dimension.

It is shown that high SNR gain is obtained for large transform size *m* but there are disadvantages having such a large transform sizes. At a fixed bit rate, the computational complexity and the codebook size is exponentially related to vector dimension, so it is generally better to have small transform sizes. The transform size,

on the other hand, should be chosen without sacrificing for coding gain. One method to decrease the transform size is to split the code vector **c** into subvectors. If it is split into *J* l-dimensional subvectors, SNR gain is given by

$$SNG_s = 10log_{10}\left[\frac{\left(\sum_{i=1}^{m}\frac{\lambda_i}{m}\right)}{\left(\sum_{j=1}^{J}\frac{d_j}{J}\right)}\right] \tag{2.38}$$

where d_j is the geometrical mean of weighting factors for each subvectors and is given as follows:

$$d_j = \prod_{k\epsilon M_j} \lambda_k^{1/n} \tag{2.39}$$

where M_j denotes the index set of the elements which belong to the j[th] subvector.

In general, $SNG_s \leq SNG$, since the denominator of (2.38) is nothing more than the arithmetic mean of d_j while that of (2.35) is expressed as the geometric mean of d_j. The equality between these two equations holds when d_j is uniformly independent of j. In other words, we can state that, even if the transformed components are split into subvectors, the SNR gain remains unchanged when the geometric mean of the weighting factors for each subvectors can be made equal.

The transform we use in TWIN-VQ is the MDCT. Although the MDCT is suboptimal for arbitrary input audio signals, it is better than the transforms like KLT which are particular to AR process. If the normalized MDCT coefficients were directly input to

the VQ, the lengths of the vectors would be between 256 and 1024. Quantizing such large vectors is not feasible since the computational complexity would be too high and it is also impractical to design an efficient codebook using algorithms like generalized Lloyd algorithm. As stated above, we can overcome this problem by splitting the MDCT coefficient vector into sub-vectors such that the geometric mean of weighting coefficients corresponding to each sub-vector remains the same. This is done by sub dividing the MDCT coefficient vector into sub-vectors. Figure 2.6 shows how a 12-dimensional vector is interleaved into two 6-dimensional subvector.

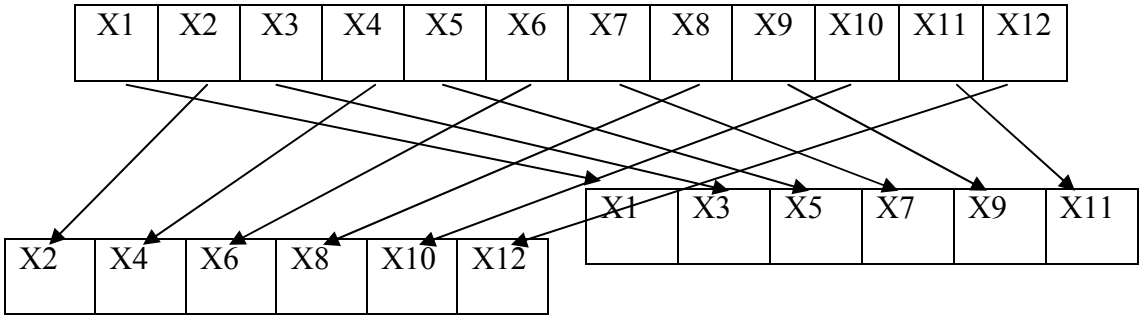

Figure 2.6: Interleaving a 12 dimensional vector into two 6 dimensional subvectors [5]

2.7 TWO-CHANNEL CONJUGATE VECTOR QUANTIZER (TC-VQ):

Twin VQ uses an efficient Vector Quantization scheme to quantize the normalized MDCT coefficients. Specifically, it uses Two-Channel Conjugate Vector Quantization (TC-VQ) which is found to be very robust against channel errors. TC-

VQ uses two codebooks that in some sense are "conjugates" of each other [9]. TC-VQ encodes the input vector by selecting a code vector from each codebook, calculates the average and finds the distortion *D* between the input vector and the averaged vector using perceptually weighted mean square distortion measure given by [5]

$$D = \sum_{i=1}^{N}\sum_{j=1}^{N}\left(x - \frac{y_{1i} + y_{2j}}{2}\right)^{T} W(x - \frac{y_{1i} + y_{2j}}{2}). \qquad (2.40)$$

In the above equation, $\boldsymbol{x}$ is the input vector, $\boldsymbol{y}_{1i}$ and $\boldsymbol{y}_{2i}$ are the vectors in the codebook *Y1* and *Y2* respectively and *W* is the diagonal matrix containing perceptual weights as its diagonal elements. As in a general VQ, we transmit indices of codebook which minimizes distortion *D* to the decoder; in a similar fashion here we send the indices of two codebook vectors to the decoder. TC-VQ has a advantage with respect to storage requirements— the effective number of code vectors needed in normal VQ is N^2 where as in TC-VQ the total number of code vectors is 2N. In this case, the length of the channel codeword *b* is reduced by half and the probability that both the output indices are destroyed in transmission is significantly reduced.

In contrast, if the two codebooks are independent, the distortion considerably increases even for small errors in transmission. SNR performance for Gaussian source is approximately proportional to the bit rate *r*. The proportional constant in dB is given by $10log_{10}(4)$ or around 6 dB/bit. If the bit rate is reduced from *r* to *r*/2 the

SNR will decrease by 3r dB, so it is important to have two conjugate codebook design to reduce distortion.

Conjugate codebooks are designed in similar fashion to normal VQ codebooks using generalized Lloyd algorithm [4]. This is an iterative algorithm which locally minimizes the sum of the quantization distortion for a given set of training sequences. Any random codebook can be used to start the process. The convergence of algorithm is guaranteed since the sum of quantization distortion is minimized in each iteration and this distortion converges to local minima. If the degree of improvement becomes less than the threshold, the codebooks are considered to have converged to the optimum.

1. Quantization of $\boldsymbol{t_i}$. For each training vector $\boldsymbol{t_i}$ the best pair of intermediate reconstruction vectors $\boldsymbol{y_{1m}}$and $\boldsymbol{y_{2n}}$ is selected so as to minimize *d* from given codebooks $Y_1$and Y_2.
2. Renew the codebook Y_1 under the constraint of fixed Y_2. The new intermediate reconstruction vectors $\boldsymbol{y_{1n}}$ is derived from the equation which expresses the partial derivative of the average distortion D_n by $\boldsymbol{y_{1n}}$ equals zero. Consider $\psi_n$to be the number of training vectors in D_n.

$$D_n = \sum_{j=0}^{N-1} \sum_{u_i \epsilon (y_{1n} \otimes y_{2j})} \left\| u_i - \frac{(y_{1n} + y_{2j})}{2} \right\|^2, \qquad (2.41)$$

$$y_{1n} = \psi^{-1} \sum_{j=0}^{N-1} \sum_{u_i \epsilon (y_{1n} \otimes y_{2j})} (2u_i - y_{2j}), \quad (2.42)$$

where $$\psi = \sum_{j=1}^{n} \sum_{u_i \epsilon (y_{1n} \otimes y_{2j})} 1. \quad (2.43)$$

3. Renew codebook Y_2 under the constraint of fixed Y_1. The above partial differentiation is repeated in the same way to renew the code vectors y_{2m} with subscript labels in (2.40-2.42) interchanged.
4. The average distortion is calculated and the process is stopped if the difference in the distortion falls below the threshold.
5. Using the same training set vectors, the process is repeated for every iteration till the difference in distortion becomes less than the threshold.

   The following Figure [2.7] shows the flow chart for the conjugate codebook design.

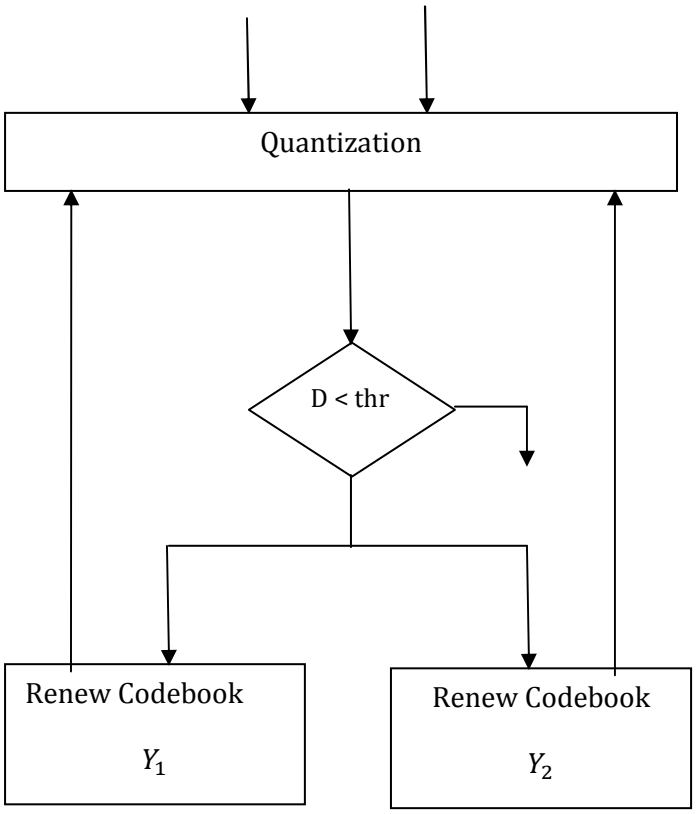

Figure 2.7: Flow chart for conjugate codebook design

3. CORRELATION EXTRACTION

3.1 Introduction

Now that the reader is familiar with TWIN-VQ and DSC. For Distributed Audio Coding, it is first necessary to extract the correlation between the side information and the temporal audio signal being encoded. The goal of the correlation extraction process is to determine the frames of the audio signal which have correlation. It has been observed that TWIN-VQ uses Linear Prediction (LP) Coefficients and Bark Scale Coefficients as side information. In this chapter, we study the correlation between the LP Coefficients and original audio signal. As stated in Chapter 1, we consider two sensor outputs X and Y, where the sensor output Y is given as a gain shifted and corrupted version of X—i.e.

$$Y = (A * X) + N \tag{3.1}$$

Here, the output Y is dependent on sensor output X—i.e. X is an independent source and Y is a dependent source.

Metrics like Mean Square Error (MSE) and Structural Similarity Measures (SSM) [10] between Linear Prediction coefficients of X and Y do not really give very precise frame correlation estimates. As we stated in Chapter 2, linear predictive coding models the audio with an AR (auto regressive) process and we are interested in finding the time domain clustering of frames from the coefficients of this AR model. It has been proven in [11] that the metric which gives the best time series clustering

for AR model is the cepstral distance measure. This cepstral distance measure is nothing more than the weighted mean square error between the cepstral coefficients. The calculation of Cepstral Coefficients from LP coefficients will be explained in detail in sections that follow. Our experiments show that the time frames extracted through cepstral distance measure show a high degree of correlation with the temporal audio frames.

3.2 CALCULATION OF CEPSTRAL COEFFICIENTS

Cepstral Coefficients matching forms the basis for correlation extraction in the proposed Distributed Audio Coding system. The block diagram of Figure 3.1 shows the procedure for calculating the Cepstral coefficients. To calculate the

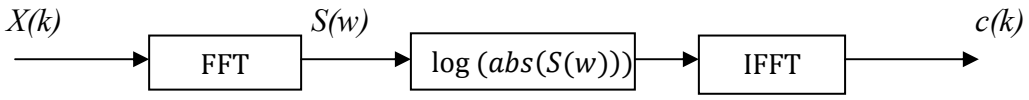

Figure 3.1: Block diagram for the calculation of cepstral coefficients

cepstral coefficients for any input stochastic process *X(k)*, a Fast Fourier transform is first applied directly to *X(k)* which results in the power spectrum *S(w)* of the input signal. Next, the logarithm of transformed sequence is obtained which is called the log spectra. Finally, the inverse Fast Fourier transform of log spectra provides an estimate of the cepstrum. The cepstrum represents a rough spectra of the input signal, and the precision of this estimate improves with the number of cepstral coefficients.

These coefficients form a good analysis space for both speech and audio, especially in noisy environments [15]. The cepstrum has been used for the detection of echoes in seismological data [14] and it has also been used to measure the distance between two signals [11]. Kalpakis *et al* used Euclidean distance between cepstral coefficients to cluster the time series of ARIMA (Auto Regressive Integrated Moving Average) models.

In TWIN-VQ Cepstral Coefficients are calculated in the process of calculating Linear Prediction coefficients and these Cepstral Coefficients represent the LPC cepstrum of the input time domain signal. Without increasing the encoding complexity, we can use these cepstral coefficients to identify the correlated frames of input audio signals *X* and *Y*. The Cepstral Coefficients for the p^{th} order Auto Regressive (AR) model can be calculated from prediction coefficients $\{a_k\}$ as follows:

$$c(n) = \begin{cases} -a(1), & for\ n = 1 \\ -a(n) - \sum_{m=1}^{n-1} \left(1 - \frac{m}{n}\right) a(m)c(n-m), & for\ 1 < n \leq p \\ -\sum_{m=1}^{p} \left(1 - \frac{m}{n}\right) a(m)c(n-m), & for\ n > p \end{cases} \tag{3.3}$$

These Cepstral Coefficients have many interesting properties for stationary time series generated by different models [13]. To list a few properties:

1) Cepstral coefficients are good at distinguishing models and they are effective as similarity measure between models. A small change in AR coefficients results in a considerable change in cepstral distance.
2) The coefficients decay rapidly to zero. Thus, it is sufficient to retain only the first few coefficients and still have most of the information about the signal. This property has the capacity to overcome curse of dimensionality. The rate at which the cepstral coefficients decay is related to location of poles and zeros of a given AR model. If the poles and zeros are closer to unit circle, the decay is slower and it is necessary to retain more coefficients.
3) When used as feature vectors, these coefficients have high discriminatory power compared to other feature vectors.
4) They are invariant to amplitude translation, amplitude scaling and time shifting. Invariance to amplitude translation provides the ability to classify time series which have same pattern but different means. Invariance to amplitude scaling says that cepstral coefficients are unchanged when time series is multiplied by a constant.
5) Convolution of two time series gives the addition of corresponding time series in cepstral domain. This property reduces the computational complexity involved with convolution in time domain.

The Cepstral distance is the distance between the logarithm of the two spectra and is given by [14]

$$d(\log(H^1), \log(H^2)) = \sum_{k=1}^{\infty} w_k (c^1(k) - c^2(k)) \tag{3.2}$$

where H^1 and H^2 are the system functions and $c^1(k)$ and $c^2(k)$ are the cepstral coefficients of H^1 and H^2, respectively. In the (3.2), w_k indicates the weight for the cepstral coefficients. The magnitude of the cepstral coefficients decays to zero for higher dimensions, hence the cepstrum can also be used as a dimensionality reduction technique along with other techniques like Principal Component Analysis (PCA). Since the higher dimensional cepstral coefficients are less significant, it is pertinent to weight them in some sense. There are many weighting functions proposed in [12]. In this thesis, we use the index weighting function given as $w_k = i^2$ where i is the descending index of the cepstral coefficient with the lower coefficient having the largest index.

The cepstral distance can be calculated instead without using the cepstral coefficients for two AR models. Consider the system functions $H^x(z)$ and $H^y(z)$ of the two sources *X* and *Y,* respectively. Let p and q be the order of functions $H^x(z)$ and $H^y(z)$ with poles $\alpha_1, \dots\dots\dots, \alpha_p$ and $\beta_1, \dots\dots\dots, \beta_p$ respectively. Then the cepstral distance between the two AR models [14] is given by

$$d(\log(H^x(z)), \log(H^y(z))) = log \frac{\prod_{i=1}^{p} \prod_{j=1}^{q} |1-\alpha_i \overline{\beta_j}|^2}{\prod_{i,j=1}^{p} (1-\alpha_i \overline{\alpha_j}) \prod_{i,j=1}^{q} (1-\beta_i \overline{\beta_j})}. \tag{3.4}$$

3.3 USE OF CEPSTRAL COEFFICIENTS FOR DISTRIBUTED AUDIO CODING

In TWIN-VQ prediction, coefficients are included as side information in bitstream. Thus, we extract the temporal correlation of audio using this side information as a similarity measure. It has been found that the cepstral distance is the similarity measure that is highly effective for extracting correlated audio frames.

The following Figure 3.2 shows the scatter plot of cepstral distance for a typical audio sequence between sources *X* and *Y* in temporal domain versus the cepstral distance in prediction domain, where *X* and *Y* are related by (3.1) with A =1 and the additive Gaussian noise having a standard deviation of 0.1.

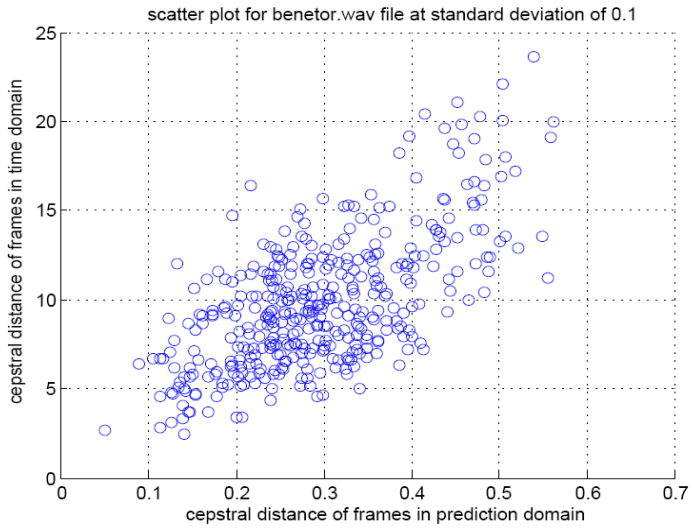

Figure 3.2: Relationship between cepstral distance in time and prediction domain

The Figure 3.2 clearly shows that there exists some degree of correlation between sources X and Y in cepstral domain that we can potentially exploit for Distributed Audio Coding. The Figure 3.3 is similar to Figure 3.2 but plotted at a standard deviation of 0.01. This figure shows that prediction cepstral distance is less at a standard deviation of 0.01 than the cepstral distance at 0.1. In the Figure 3.3 we can observe that most of the frames have very small amount of prediction cepstral distance. Note that this is not true as often for the temporal cepstral distance. This mean that there is a high degree of correlation in prediction domain.

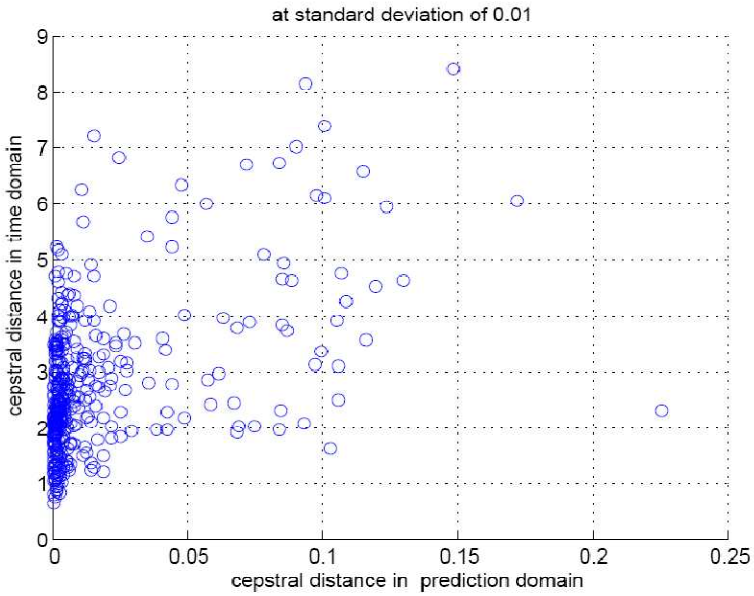

Figure 3.3: Scatter plot of cepstral distance in time and prediction domain at an additive noise standard deviation of 0.01

Studying Figure 3.3, we evaluate the frames which have cepstral distance greater than 0.05 as outliers. On considering the particular properties of outliers it has been observed that the mean of the total energy of outliers is considerably less than the mean of total energy of the frames which cluster along zero.

The utility of using cepstral distance in prediction domain can be seen even more clearly in Figure 3.4 where we show that on the average this distance increases in an approximately linear fashion with increasing noise power.

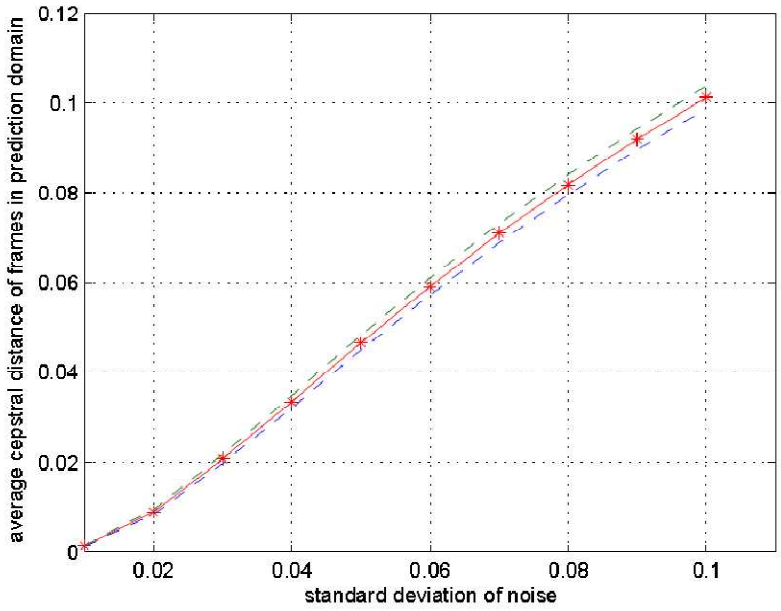

Figure 3.4: Average cepstral distance of Linear prediction coefficients vs noise power. Dashed lines indicate 90% confidence intervals.

To further illustrate the point that the correlation exist in cepstral prediction domain, we plot in Figure 3.5 and Figure 3.6 the mean of the cepstral predictor distance

between sources *X* and *Y* versus the gain constant *A* in [3.1] at standard deviations of 0.01 and 0.1 respectively.

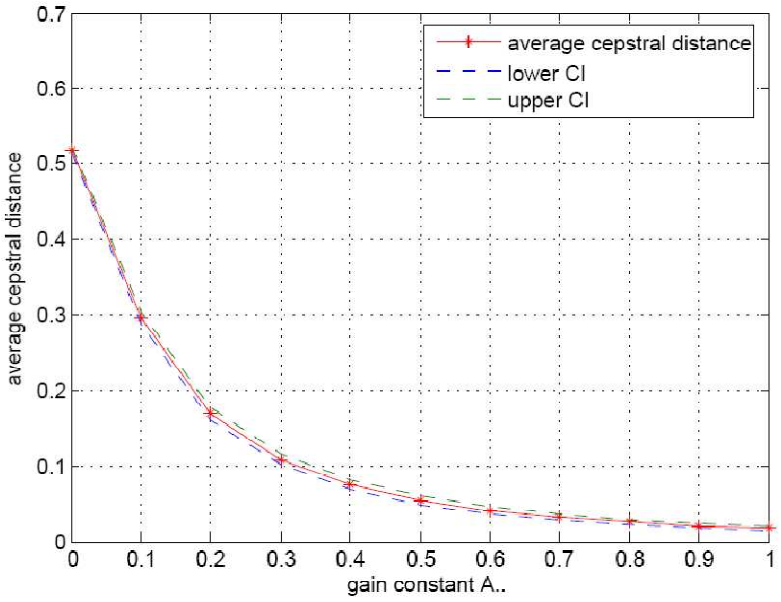

Figure 3.5: Average Cepstral Predictor distance versus Gain (A) at a standard deviation of 0.01 dashed lines show 90% Confidence intervals.

In Figure 3.6 noise power is fixed at a standard deviation of 0.1 and the gain is varied from 0 to 1. This Figure shows a more linear relationship between gain A and cepstral predictor distance. Both Figures 3.5 and 3.6 show that the cepstral predictor distance monotonically decreases as the gain constant *A* approaches 1.

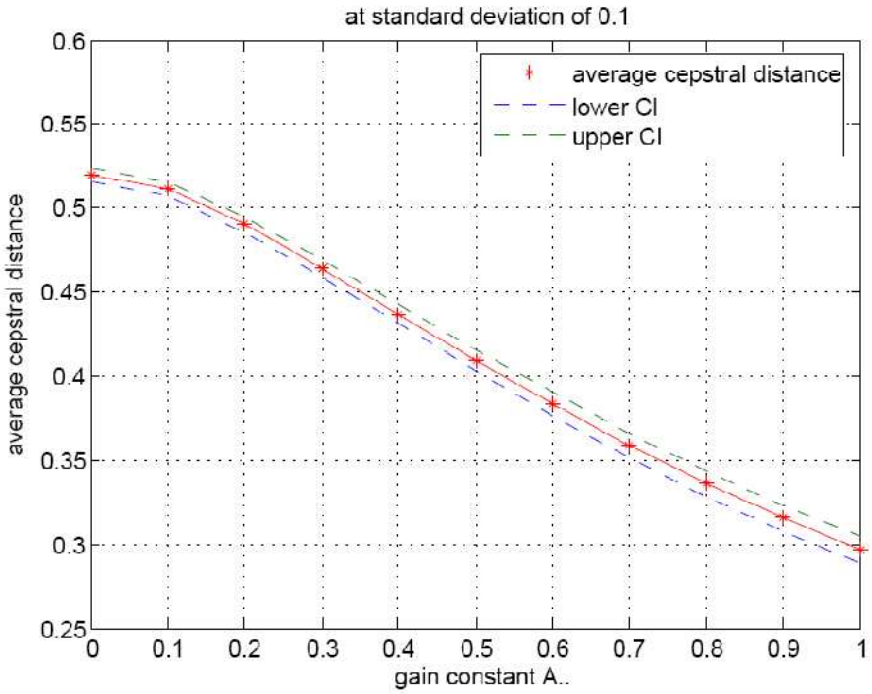

Figure 3.6: Average cepstral distance of predictor coefficients vs Gain constant (A) at a standard deviation of 0.1 dashed lines indicate 90% confidence intervals.

4. ALGORITHM

4.1 ALORITHM OF CONDITIONAL ENCODING

The algorithm for conditional encoding of dependent source *Y* given independent source X is shown in Figure 4.1. Initially, a weighted cepstral distance for prediction coefficients is calculated for all the frames of input audio.

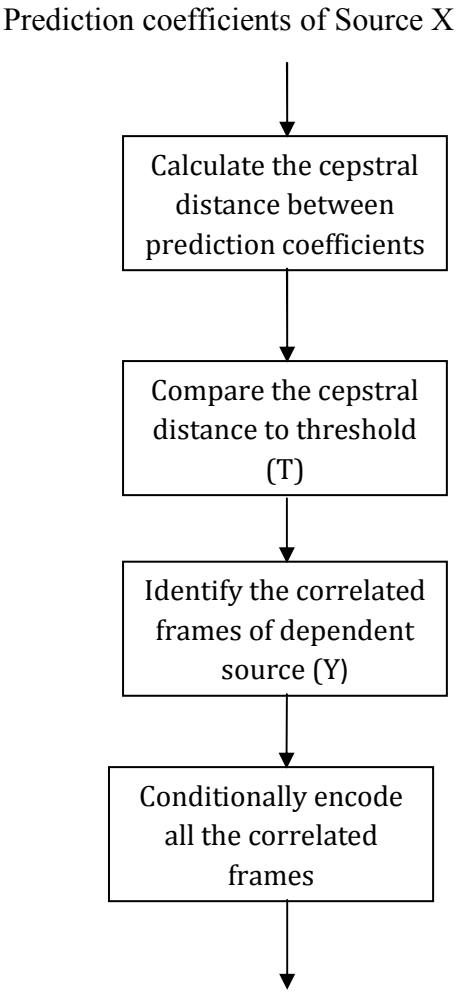

Figure 4.1: Steps to conditionally encode the dependent source (Y)

A comparison of this cepstral distance to a threshold *T* is performed to identify the correlated frames. We define the frames which exhibit the cepstral distance less than the threshold (*T*) as being correlated frames. Finally, we conditionally encode all of the correlated frames.

To test the accuracy of the correlated frames identified by this algorithm, the correlation coefficient for all the frames which have the cepstral distance less than the threshold (*T)* is calculated in temporal domain for the benetar.wav audio file and compared to the correlation coefficient of frames which have higher cepstral distance than threshold (*T*). It has been observed that the frames which have lower cepstral distances have high correlation coefficient. Table 4.1 shows the correlation coefficient for frames classified as common versus that for frames classified as uncommon frames.

Table 4.1: Correlation coefficient of frames in temporal domain for benetar.wav.

	Common Frames	Uncommon Frames
Number of Frames	112	281
Mean of Correlation Coefficient (ρ)	0.8328	0.5838

In the above table, correlation coefficient ρ is evaluated for two sources X and Y where Y is given by (3.1) with $A = 1$ and $N = 0.1$. Correlated frames are identified by the algorithm using a threshold T set to 40% of maximum cepstral distance.

To test the potential of conditional encoding, it is necessary to estimate the conditional entropy of dependent source Y given the independent source X. Here, we estimate the 1st order entropies of the MDCT coefficients of dependent source Y using the following equation:

$$H(Y|X) = -\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} P(y_j|x_i)\,P(x_i)log_2P(y_j|x_i) \tag{4.1}$$

The 1st order entropies are calculated on single frames of input audio. First, the MDCT coefficients of both the sources X and Y are scalar quantized to a particular number of levels. Let $x_0, x_1, \ldots\ldots x_{N-1}$ be the quantization levels or the alphabet of source X and $y_0, y_1, \ldots\ldots y_{M-1}$ be the quantization levels or the alphabet of source Y. To calculate the conditional entropy given in (4.1), it is necessary to estimate the conditional probability $P(y_j|x_i)$. The conditional histogram $P(y_j|x_i)$ is evaluated as follows: at the given values of x_i for the independent source X, the total number of y_j's for the dependent source Y are calculated. The above process is repeated for every alphabet of source X.

After identifying the correlated frames of input audio, we then estimate the 1st order conditional entropies of the MDCT coefficients in these common frames. These

entropies along with their 90% confidence intervals are given in Table 4.2 for a representative audio sequence.

Table 4.2: Sample mean of 1st order entropies at 8kbps with N=0.1 and A =1.

Quantization levels (L)	Entropy of independent source *(H(x))*	Entropy of dependent source *H(y)*	Conditional Entropy *H(y/x)*
4	0.4992±0.0022	0.4952±0.0023	0.2595±0.0046
8	0.5676±0.0028	0.5739±0.003	0.2963±0.0053
16	0.6358±0.0028	0.6448±0.0026	0.3080±0.0051
32	0.7024±0.0021	0.7111±0.0031	0.3081±0.0052
64	0.8256±0.0057	0.8504±0.0042	0.3475±0.0058
128	1.0400±0.0088	1.0779±0.0054	0.3824±0.0070

The above Table 4.2 is given at the encoding rate of 8kbps for the beneter.wav file and we note that the conditioning of *Y* on *X* reduces the entropy by between 60% and 64% at higher levels of quantization. This is a considerable decrease in conditional entropy of *Y* upon conditioning on *X*. This shows that there exists high degree of correlation between sensor outputs at low bit rates which can be potentially exploited to reduce the encoded bit rate.

Table 4.3: Sample mean of entropies at 16 kbps for bene.wav with A = 1 and N = 0.1 at 90% C.I.

Quantization levels (*L*)	Entropy of independent source (*H(x)*)	Entropy of dependent source *H(y)*	Conditional Entropy *H(y/x)*
4	0.8606±0.0022	0.8612±0.0014	0.5660±0.0054
8	0.9786±0.0034	1.0209±0.0029	0.6781±0.0052
16	1.1557±0.0044	1.1985±0.0029	0.7294±0.0073
32	1.3124±0.0062	1.3686±0.0034	0.7645±0.0077
64	1.5269±0.0092	1.6322±0.0061	0.8529±0.0084
128	1.9373±0.0164	2.1430±0.0084	1.0053±0.0102

Examining Table 4.3 where the encoding rate is 16kbps, the decrease in entropy of dependent source *Y* upon conditioning on *X* ranges from 45% to 53%. This shows that there is a decrease in gain of conditional entropy from 8 kbps to 16 kbps.

In order to calculate the conditional entropy estimate, the values of the MDCT coefficients of correlated frames must be binned. For all of the quantization levels considered in Table 4.2 and 4.3 the number of bins are equal to number of quantization levels (L). Let us consider the case of unequal bin sizes. Tables 4.4 and 4.5 show the variation in conditional entropy with bin size. For the results in Table

4.4, the number of bins for *Y* is kept constant at 8 bins. Here, we increase the number of bins for *X*, the source which we are conditioning on, and we note that conditional entropy increases montonically. This is obvious as we are conditioning *Y* on more number of bins it adds for extra bit of information to *H(y/x)*. To see this, consider how the product $P\left(\frac{y_j}{x_i}\right)P(x_i)$ in 4.1 changes with an increase in the number of bins in *X*. Specifically $P(x_i)$ increases while the conditional pdf $P\left(\frac{y_j}{x_i}\right)$ decreases, but the increase in $P(x_i)$ is faster than the decrease in $P\left(\frac{y_j}{x_i}\right)$, so the conditional entropy increases with increase in number of bins of *X*. Hence, we can achieve some gain in conditional encoding by using a small number of bins for *X* relative to the number of quantization levels used for *Y*. This is more evident from Table 4.5.

Table 4.5 shows the variation in conditional entropy *H(y/x)* with the change in bin size of *Y*. Here, the transform coefficients of *X* are kept constant at 8 bins. The conditional entropy *H(y/x)* decreases with increase in number of bins for *Y*. This is exactly the opposite of what occurred in Table 4.4.

Table 4.4: Change in Conditional entropy (*H(y/x))* with bin size of *X* at the encoding rate of 8kbps at the standard deviation of 0.1.

Number of bins of *X*	*H(x)*	*H(y)*	*H(y/x)*
4	0.5463	0.6321	0.1840
8	0.6183	0.6321	0.2459
16	0.6837	0.6321	0.3006
32	0.7865	0.6321	0.3480
64	0.9804	0.6321	0.4043

Table 4.5: Change in Conditional entropy (*H(y/x))* with bin size of *Y* at the encoding rate of 8kbps at a standard deviation of 0.1.

Number of bins of *Y*	*H(x)*	*H(y)*	*H(y/x)*
4	0.6183	0.5619	0.2539
8	0.6183	0.6321	0.2459
16	0.6183	0.7015	0.2371
32	0.6183	0.8207	0.2236
64	0.6183	1.0477	0.2037

Table 4.6 shows the effect of noise power on entropy estimates with fixed number of quantization and binning levels. This table shows that there is a gradual increase in entropy of dependent source with noise power but a dramatic increase in the conditional entropy this as expected. The entropy estimate of independent source *X* remains constant since it is independent of noise variance.

Table 4.6: Entropy estimates with noise power at quantization levels L = 8.

Standard deviation of Noise	*H(x)*	*H(y)*	*H(y|x)*
0.001	1.1322± 0.0041	1.1294± 0.0040	0.2875± 0.0129
0.01	1.1322± 0.0041	1.1446± 0.0040	0.5167± 0.0036
0.05	1.1322± 0.0041	1.1642± 0.0044	0.5592± 0.0027
0.1	1.1322± 0.0041	1.1704± 0.0049	0.5692± 0.0029
0.5	1.1322± 0.0041	1.2241± 0.0041	0.6064± 0.0024

4.2 ENCODING WITH ARITHMETIC CODER

Arithmetic coding is considered to be a very efficient lossless compression technique. This coder is preferred to Huffman coding particularly when the input source has a small number of alphabets (binary sources) and the probability distribution of the alphabet symbols is highly skewed. Arithmetic coding techniques are also useful in that they separate the model from the actual coder [4].

In an arithmetic coder, a unique number or a tag is assigned to a sequence of symbols produced by the input source. This tag is chosen from the set of real numbers on the interval between 0 and 1 and the size of this set is infinity since we have infinite number of possibilities. The tag generation is completely based upon the symbol

probabilities generated by the model, and the arithmetic encoder uses cumulative distribution functions to map sequence of random inputs onto the unit interval. The tag itself is chosen based on these cumulative distribution functions. Finally, this tag is converted into a binary sequence. Higher probability (more likely) symbols reduce the tag's numeric range by less than the lower probability symbols; hence, higher probability symbols add fewer bits to the message. The complete details and the practical implementation of an arithmetic coder is given in [21]

In order for the decoder to decode the binary representation of the tag, it has to have the access to the model. Practically, this implementation of the arithmetic coder requires side information for the model to be sent to the decoder since the distribution of source in the form of probability of model is not generally know *a priori*. A more practical approach is to use an adaptive arithmetic coder. In this implementation, we start with a single count for each symbol and update the symbol count as we encounter the symbol. The decoder also employs the same updating procedure to stay in synchrony with encoder. Any difference in updating procedure between the encoder and decoder leads to erroneous decoding of encoded symbol.

In this work, we use an arithmetic coder to encode the bit planes of each flattened MDCT coefficient. First, the most significant bit plane (MSB) is encoded. The bit rate of the bit planes increases from MSB to LSB since the uncertainty increases from most significant bit plane to least significant bit plane. As we are using arithmetic coder to encode binary words, it is called a binary arithmetic coder. In order to avoid

sending side information to the decoder, we use an adaptive model with this binary arithmetic coder.

For comparison with the results of conditional encoding of dependent source Y, first we encode both X and Y independently using single context binary arithmetic coder. Here, the model is updated after encoding each zero. To conditionally encode the dependent source Y given the independent source X, we use multiple contexts and the context selection is based upon the corresponding MDCT coefficient of X.

The arithmetic coder we use is implemented in MATLAB and accepts 8, 16, 32 or 64 bit unsigned integer as input. The MDCT coefficients, however, are theoretically real values. To make the final flattened MDCT coefficients compatible with arithmetic coder, it is crucial to rescale them. First, the sign bit is removed by taking only the absolute value of MDCT coefficient. Next, the sign removed MDCT coefficient is normalized by dividing it with the maximum value. Finally, the normalized values are rounded (quantized) and converted to the range of unsigned 8, 16, 32 or 64 bit integer.

Table 4.7 shows the conditional coding results for benetar.wav at an encoding rate of 8kbps and with a noise standard deviation of 0.1 and 0.01 in (3.1).

Table 4.7: Conditional coding at 8kbps and scaling transform coefficients to 8 bit unsigned integer, excluding sign bit coding.

Noise Power	Encoding rate for X in bits/sample	Encoding rate for Y in bits/sample	Conditional coding rate for Y in bps	Number of bins or contexts used
0.1	2.3109	2.4281	1.5336	2^8
0.01	2.3109	2.3519	1.4652	2^8
0.1	2.3109	2.4269	2.2303	2^4
0.01	2.3109	2.3438	2.1102	2^4

Studying the Table 4.7, we note that there is a greater degree of reduction in bit rate for large number of contexts at different noise levels but that the reduction is not so significant for small number of contexts. The reason for this is that the correlation improves when taking more number of bins into account because we can more accurately classify the bins. Moreover, the bit savings obtained in Table 4.7 is 37% which is less than what we expected from the conditional entropy of Table 4.2 where the reduction should be around 56%. The reason can be attributed to loss in sign bit due to scaling—specifically, the correlation between the sign of X and Y may be lost when scaling is done for the compatibility.

Table 4.8 shows that with increase in precision of bit planes (16 bit unsigned int) there is an increase in gain of conditional encoding. At half the number of contexts out of total number of contexts possible, the reduction in bitrate at 0.1 standard deviation for 8 bit unsigned integer transform coefficients is 8% from Table 4.7 but at the same noise power and for 16 bit unsigned integer transform coefficients the reduction is 16% from Table 4.8.

Table 4.8: Conditional coding of Y at encoding rate of 8kps and rescaling of transform coefficients to 16 bit unsigned integer.

Noise Power	Encoding rate for X in bits/sample	Encoding rate for Y in bits/sample	Conditional encoding rate in bits/sample	Number of bins or contexts used
0.01	4.2848	4.4181	3.5585	2^8
0.1	4.2848	4.5869	3.8383	2^8

Table 4.9: Change in encoding rate with respect to number of bitplanes.

Number of bitplanes	Bitrate of source *X*	Bitrate of source *Y*	Bitrate of source *Y* given *X*	Number of contexts
2	0.3275	0.3356	0.2519	2^2
3	0.5481	0.5730	0.4239	2^3
4	0.9047	0.9163	0.7184	2^4
5	1.5015	1.5387	1.2652	2^5
6	2.3923	2.4506	2.0077	2^6

From the Table 4.9 we can observe that with the increase in number of bitplanes there is a decrease in percentage difference between conditional and non-conditional coding and the table shows that there is a potential of scalable coding within conditional coding.

Table 4.10 Encoding the sign bit of dependent source *Y* for benetar.wav file at 8 and 16 kbps at noise power of 0.01

Encoding rate	Coding independently	Coding dependently
8 kbps	0.4964	0.2891
16 kbps	0.8610	0.6357

Table 4.10 shows that with an increase in encoding rate, the percentage of reduction in conditional coding of sign bit is reduced from 41.7 % to 26.1 % since with increasing encoding rate there is more uncertainty about the sign bits. More over the sign bits of two sources X and Y are clearly correlated otherwise, there would be no reductions at all using context-based coding for sign bits of Y with respect to X.

CONCLUSIONS AND FUTURE WORK:

We have introduced a new approach for detecting correlation in temporal domain between two audio sequences. This correlation detection is performed using only the TWIN-VQ side information, which is obtained with minimal decoding of passively received information from nearby sensor nodes. In particular, we use a cepstral decomposition of linear prediction coefficients. We have observed the resulting cepstral coefficients are excellent features for time series clustering and hence are useful for extracting the correlated temporal frames between two sensed audio signals. We also introduced an algorithm for conditionally encoding the audio. This algorithm conditionally encodes the locally captured signal with respect to the passively received signal. This conditional encoding reduces the bit rate of audio sequence.

In order to find the 1^{st} order entropies, we performed uniform scalar quantization on the final flattened coefficients. These reconstruction levels are termed as bins. Better quantization bins can be created by employing pdf-optimized quantization scheme instead of uniform scalar quantization since the distribution of flattened transform coefficients is highly peaked at origin. But this also entails to transmission and coding of side information. To further get reductions in the bit rate, it is vital to jointly encode sign bits with bitplanes using the context-selection based on sign bits and the values of the MDCT coefficients produced by the independent source. Moreover using the context-based binary arithmetic coder on bitplanes we have scalable

compression—as we have observed, each bitplane has different encoding rates so that scalable compression can be realized by only sending the most significant bit planes at lower bit rates. The tradeoff is of course, a loss in audio quality. To faithfully reproduce the reconstructed signal, all the bitplanes must be encoded. Here, we are achieving the scalability by conditional bitplane coding, but better results might be obtained by optimizing the scalability for the independent source.

REFERNCES:

[1] (1998(E)) Mpeg-4 standards document. [Online]. Available: ISO/IEC FCD 14496 - 3 subpart 4.

[2] N. Iwakami, T. Moriya, and S. Miki, "High quality audio coding at less than 64 kbits/s by using transform domain weighted interleave vector quantization (twinvq)," in *Proc. of the Internation Conf. on Acoustics Speech and Signal Processing*, May 1995, pp. 3095–3098.

[3] N. Iwakami and T. Moriya, "Transform domain weighted interleaved vector quantization (twin vq)," in *Proc. 101st convention of the audio engineering society*, p. preprint 4810.

[4] Khalid sayood, *Introduction to data compression*. Morgan Kaufmann Publishers, 2006

[5] Srivatsan A Kandadai, "Scalable audio coding that scales to low bit rates" *Ph.D dissertation*, New Mexico State University, May 2007.

[6] Naoki Iwakami, Takehiro Moriya, Satoshi Miki, Kazunaga Ikeda, and Akio Jin, "Audio Coding Using Transform-Domain Weighted Interleave Vector Quantization (Twin VQ)" *Electronics and Communications in Japan*, Part 3, Vol. 81, No.3, 1998.

[7] M. Tribolet and R. E. Crochiere, "Frequency domain coding of speech," *IEEE Trans. Acoust. , Speech, Signal Processing*, vol ASSP-27, pp, 512-530, 1979.

[8] Takehiro Moriya and Masaaki Honda, "Transform coding of speech using a weighted vector quantizer", *IEEE journal on selected areas in communications*, vol. 6, No. 2, February 1988.

[9] T.Moriya, "Two Channel Conjugate Vector Quantization for noisy channel speech coding", *IEEE Journal on selected areas in communications*, vol 10, Issue 5, June 1992.

[10] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli, "Image Quality Assessment: From Error Measurement to Structural Similarity" *IEEE Transactions on Image Processing*, vol.13, 2004.

[11] Richard J Martin, " A Metric for ARMA Process" *IEEE Transactions on Signal Processing*, vol 48, No 4, April 2000.

[12] Zheng Fang, Wu Wenhu and Fang Ditang, "A log-indexed Weighted Cepstral Distance Measure for Speech recognition" *Journal of computer science and technology*, 12(2), pp 177-184, March 1997.

[13] Konstantinos Kalpakis, Dhiral Gada and Vasundhara Puttagunta, "Distance Measure for Effective Clustering of ARIMA Time Series" *Proceedings of ICDM 2001* , San Jose, pp 273-280.

[14] Jeroen Boets, K. De Cock, M. Espinoza and B. De Moor, " Clustering Time Series, Subspace Identification and Cepstral Distances" *Communications in Information and Systems*, Vol. 5, No.1, pp. 69-66,2005

[15] L.Gu and K.Rose, "Perceptual Harmonic Cepstral Coefficients for Speech Recognition in Noisy Environment," in *IEEE Int. Conf on Acoustics, Speech and Signal Processing*, Salt Lake City, UT, May 2001, pp. 125 - 128

[16] Zixiang Xiong, Angelos D. Liveris and Samuel Cheng, "Distributed Source Coding for Sensor Networks", *IEEE Signal Processing Magazine*, vol. 4, pp. 80 – 94, September 2004.

[17] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inform. Theory*, vol. 19, pp. 471-80, July 1973.

[18] A. Wyner and J. Ziv, "The rate distortion function for source coding with side information at the decoder," *IEEE Trans. Inform. Theory*, vol. 22, pp. 1-10, Jan. 1976.

[19] Q.Zhao and M.Effros. "Broadcast system source codes: A new paradigm for data compression," in *Proc. 33rd Asilomar Conf. Signals, Systems and Computers*, Pacific Grove, CA, 1999, pp.337-341

[20] S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense microsensor network," *IEEE Signal Processing Magazine*, vol. 19, pp. 51-60, March 2002.

[21] Ian H. Witten, Radford M. Neal and John G. Cleary, "Arithmetic coding for Data Compression," Communications of the ACM, vol 30, No 6, June 1987.