APPLICATIONS OF ANALYSIS AND SYNTHESIS TECHNIQUES FOR COMPLEX SOUNDS

XINGLEI ZHU

(B.E. (Hons.), USTC, CHINA)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2004

Acknowledgements

I am indebted to my supervisors, Dr. Lonce Wyse and Dr. Ye Wang, for their extraordinary support and guidance throughout the past one and a half years.

I would like to thank my friends, Zhicheng Zhou, Rong Zhang and Rui Li for their encouragement that inspired me so much.

I am grateful to Dr. Qibin Sun who constantly helped me in the last two years in Singapore, in my study and daily life.

I am also grateful to the National University of Singapore and Institute for Infocomm Research for awarding me a Research Scholarship.

Finally, I would like to thank my parents who initially leaded me to the way of science.

Table of Contents

List of Figures

Summary

In this thesis we present two applications of sound modeling/synthesis in sound texture modeling and packet loss recovery. In both applications we build a model for specific sounds and resynthesize them. The modeling/synthesis process provides extreme low bit representation of the sound and generates perceptually similar sounds.

In sound texture modeling, we build a model for specific kind of sounds that contains a sequence of transients, such as fire burning sound. We use a Poisson distribution to simulate the occurrence of transients and time-frequency linear predictive coding to capture the time and frequency spectrum contour of each event.

Another application of sound modeling/synthesis is packet loss recovery. We improve the content based unequal error protection (C-UEP) scheme, which uses a percussive codebook to recover the lost packet containing percussive sounds. Our solution is an unequal error protection scheme that gives more protection to drum beats in music streaming due to the perceptual importance of the musical beat. We make a significant improvement on the codebook construction process by codebook modeling and reduce the redundant information to only 1% of the previous C-UEP system.

We make evaluations for both applications and discuss the limitations of the

current system. We also demonstrate the other possible applications and future work.

Chapter 1 Introduction

1.1 Motivation

Sound is everywhere in our daily life. In the real world, sounds are made by physical process and have different characters by themselves. Digital recorded real sounds are usually in the form of *Pulse Code Modulation* (PCM), which is formed by sampling analog signals at regular intervals in time and then quantizing the amplitudes to discrete values. Such a representation is storage consuming and the sound characters, such as pitch and timbre, are usually inconvenient to change.

Sound modeling/synthesis provides a means to present sounds in a low bit rate. A “sound model” is a parameterized algorithm for generating a class of sounds and a “synthesizer” is an algorithm to regenerate a specific class of sounds using sound model parameters. Sound models can provide extremely low bit rate representations, because only model parameters need to be communicated over transmission lines. That is, if we have class-specific decoder/encoder pairs, we can achieve far greater coding efficiencies than if we only have one pair that is universal [Scheirer]. An example of using a class-specific representation for efficiency is speech coded as phonemes. The problem is that we do not yet have a set of models with sufficient coverage of the entire audio space, and there exist no general methods for coding an arbitrary sound in terms of a set of models. The process is generally lossy and the “distortion” is difficult to quantify. However, there are specific application domains where this kind of model-based codec

strategy can be very effective. For example, Chapter 4 describes a packet loss recovery method for transients in music using a “beat” model that vastly reduces the amount of necessary redundant data for error recovery. Another example might be sports broadcasting where a crowd sound model could be used for low bit-rate encoding of significant portions of the audio channel.

If generative sound models are used in a production environment, the same representation and communication benefits exist. Ideally, all audio media could be parametrically represented just as music is currently with MIDI (musical instrument digital interface) control and musical instrument synthesizers. In addition to coding efficiency, interactive media such as games or sonic arts could take advantage of the interactivity that generative models afford. For example, sound textures are an important class of sounds for interactive applications, but in a raw or even compressed audio form they have significant memory and bandwidth demands that restrict their usage. Building specific models for sound texture is very useful in such applications due to the storage requirements of sound models.

Sound models also provide variety in synthesized sounds, which is hard to implement or memory-consuming for recorded sounds. Because in sound models what we preserve for a specific class of sound is only parameters, we can change the synthesized sounds by changing the parameters. This kind of flexibility is hard to apply directly to recorded sounds without sound models. Consider a *virtual reality* (VR) environment where we need different water flowing sounds in different parts and these sounds need to change when some specific event happens. To implement it we need a large collection of recorded

water sounds if we use recorded sounds. The situation is quite different when we have a model of water sounds, what we need to do is only to transfer a new set of parameters and change some of them when needed. Another example is digitally synthesized music. By building physical models of musical instruments, we can synthesize music sounds virtually or even create some new sounds that could not be played by traditional music instruments.

1.2 Contribution

In this thesis we present two applications of sound modeling/synthesis. The first application is to build a model for specific class of sounds which consists of transient sequences. The second one is building codebook model in packet loss recovery to reduce redundant information. In both applications, the sound model/synthesis strategy greatly reduces the requirement of memory and provides variety of sounds.

1.3 Thesis Organization

The remaining parts of this thesis are organized as follows. In Chapter 2, we introduce some background knowledge, including sound synthesis technology, linear predictive coding (LPC) and hidden Markov model (HMM). Chapter 3 presents an application of sound modeling/synthesis in specific sound texture. Chapter 4 gives details of the application of sound modeling/synthesis in packet loss recovery. Finally, in Chapter 5 we draw some conclusion and discuss future work.

Chapter 2 Background

In this chapter we present some background information that will be used in the later chapters of this thesis. In section 2.1, we briefly present two kinds of sound synthesis technologies, additive sound synthesis and subtractive synthesis. Section 2.2 gives more details about *linear predictive coding* (LPC), a kind of subtractive synthesis methods. In section 2.3 we show the concept of the *hidden Markov models* (HMM).

2.1 Sound Synthesis Technology

Sound Synthesis, together with sound source modeling technologies, provide a wide applicable means to model and recreate audio signal. In this section we present an overview of two kinds of general used sound synthesis methods: additive sound synthesis and subtractive sound synthesis.

2.1.1 Additive Sound Synthesis

Additive synthesis, also called Fourier synthesis, is a straight forward method of sound synthesis. It is a type of synthesis which produces a new sound by adding together two or more audio signals. The sources added together are simple waves such as sine waves and are in the simple frequency ratios of harmonic series. The

resultant absolute amplitude is the sum of the amplitudes of the individual signals. The resulting sound is the sum of the individual frequencies taking into account.

According to the Fourier theory, any periodic sound can be created by combining multiple sine waves at different frequency bins, phase angles and amplitudes. For non-periodic sounds, windows are applied to the sounds to cut frames out from the sounds. Each frame is considered as one period of an infinite periodic sound and the same Fourier theory works. In practice, most instrumental sounds include rapidly varying and stochastic components so that there are thousands of partials with different frequency and phase. Thus additive synthesis is not applicable to synthesize actual sound in physical instruments due to the great number of partials to be implemented. To make a practical implementation, some simplifications were proposed. One of them is to group the partials into bundles of mutually harmonic partials so that Fast Fourier Transform can be used to generate each group separately and efficiently.

Additive synthesis is computationally expensive and it generally requires a great amount of control data, even in reduced form. Thus the psychoacoustical significance of a single parameter is quite limited. Furthermore, additive synthesis performs badly in the presence of stochastic components and highly transient signals.

2.1.2 Subtractive Sound Synthesis

Subtractive synthesis reflects the opposite process of the additive synthesis. While additive synthesis works from bottom up, subtractive synthesis takes a top-down

scheme.

Subtractive synthesis starts with a basis waveform, which is rich in frequency partials. Then we subtract frequencies from this basis waveform. This step is usually done by using filters and the filters we use need to be time-variable.

Subtractive synthesis is a very workable method. Because low−order filtering is very intuitive, subtractive synthesis is easy and rewarding to use. Most of its parameters also have psychoacoustical semantics—timbre is created by taking a proper starting waveform and shaping its spectrum with filters. Modulation is then applied to the sound to make the sound more lively and organic. Subtractive synthesis also has some disadvantages, accurate instrument simulations are surprisingly difficult to create because of the simplicity of the synthesis engine. The synthesized sounds often do not sound very good without extensive modification and addition of features.

2.2 Linear Predictive Coding (LPC)

As a kind of pole-zero filters, linear predictive coding (LPC) is one of the most powerful audio signal processing techniques, especially in speech processing domain. First introduced in the 1960's, LPC is an efficient means to achieve synthetic speech and speech signal communication [Schroeder]. LPC captures the frequency spectrum contour of the original signal and provides an extreme economical model of the original signal.

For speech signal, the LPC implements a type of vocoder [Arfib], which is an analysis/synthesis scheme where the spectrum of a source signal is weighted by the spectral components of the signal analyzed. In the standard formulation of LPC, an all-pole filter is applied to the original signal and a set of LPC coefficients and a residual is obtained. In the synthesis process, a source-excitation process is pursued. In speech synthesis, the source signals are either a white noise or a pulse train, thus resembling voiced or unvoiced excitations of the vocal tract, respectively.

The later sections of this section are arranged as follow: section 2.2.1 introduces the general pole-zero filter; section 2.2.2 introduces the concept of transfer function; section 2.2.3 shows how to calculate LPC coefficients; section 2.2.4 shows how audio signal is modeled and synthesized by LPC filter; section 2.2.5 shows the reflection coefficients, another presentation mean of LPC coefficients.

2.2.1 Pole-Zero Filter

Generalized pole-zero filter can be represented as:

$$\tilde{s}(n) = \sum_{k=1}^{p} a_k s(n-k) + G\sum_{l=1}^{q} b_l u(n-l), 1 \le k \le p,\ 1 \le l \le q$$

where s(n) are inputs and u(n) are outputs of the system.

When $a_k = 0,\ 1 \le k \le p,\ \ \tilde{s}(n) = G\sum_{l=1}^{q} b_l u(n-l),\ 1 \le l \le q$ is called all-zero model or moving average model (MA-model); when $b_l = 0,\ 1 \le l \le q$, $\tilde{s}(n) = \sum_{k=1}^{p} a_k s(n-k), 1 \le k \le p$ is called all-pole model or autoregressive model (AR-model). LPC filter is a kind of all-pole model.

2.2.2 Transfer Function

For linear time invariant (LTI) system, the output y from a linear time-invariant filter with input x and impulse response h is given by the convolution of h and x, i.e., $y(n) = h(n) * x(n)$, where "*" means convolution. Take the z-transform of both sides of $y(n) = h(n) * x(n)$ and we get $Y(z) = H(z)X(z)$.

The transfer function (or system function) $H(z)$ of a linear time-invariant discrete-time filter is defined as $Y(z)/X(z)$, where $Y(z)$ denotes the z-transform of the filter output signal $y(n)$, and $X(z)$ denotes the z-transform of the filter input signal $x(n)$. The transfer function provides an algebraic representation of a linear, time-invariant (LTI) filter in the frequency domain.

The transfer function of the pole-zero filter is:

$$H(z) = \frac{S(z)}{U(z)} = G\frac{1+\sum_{l=1}^{q} b_l z^{-l}}{1+\sum_{k=1}^{p} a_k z^{-k}}, \text{where } S(z) = \sum_{n=-\infty}^{\infty} s_n z^{-n},\ G \ \textit{is gain.}$$

For all-zero model, the transfer function is:

$$H(z) = \frac{S(z)}{U(z)} = G(1+\sum_{l=1}^{q} b_l z^{-l}), \text{where } S(z) = \sum_{n=-\infty}^{\infty} s_n z^{-n},\ G \ \textit{is gain.}$$

For all-pole model, the transfer function is $H(z) = G\frac{1}{1+\sum_{k=1}^{p} a_k z^{-k}}, G \ \textit{is gain.}$

2.2.3 Calculation of LPC

To estimate LPC coefficients (a_k), use short-term analysis technique and for each segment, minimize the total prediction error by calculating the minimum squared error

$$E_m = \sum_n e_m{}^2[n] = \sum_n \left(x_m[n] - \tilde{x}_m[n] \right)^2 = \sum_n \left(x_m[n] - \sum_k^p a_k x_m[n-k] \right)^2$$

Take the derivative to the above equation and set it to zero, we can get the Yule-Walker equations: $\sum_{k=1}^{p} a_k \phi_m[i,k] = \phi_m[i,0]$, where $\phi_m[i,j] = \sum_n [x_m[n-i] x_m[n-j]]$.

This equation can be solved by autocorrelation method or by covariance method.

2.2.4 LPC Analysis and Synthesis Process

In analysis process, an all-pole LPC filter coefficients are calculated as described in previous section. By applying the LPC filter on the original signal $S(n)$, we get the residual

$$e(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k)$$

In synthesis process, a source signal $e'(n)$, usually white noise, is used instead of the residual to excite the LPC filter. So we get the resynthesized signal

$$s'(n) = e'(n) + \sum_{k=1}^{p} a_k s'(n-k)$$

The regenerated signal has the similar frequency spectrum as the original audio signal.

2.2.5 Reflection Domain Coefficients

LPC coefficients are not robust to change. The stability is hard to judge and is easily affected by a small change of the filter coefficients value. This problem no longer exists by translating LPC coefficients into reflection domain by on Levinson's recursion [Kay]. The stability of reflection coefficients is very easy to check: it is stable iff the absolute value of all the reflection coefficients are smaller than 1. Another advantage of reflection coefficients is that it can be interpolated, as long as the results are still stable filter coefficients. But when near 1 and -1, it is sensitive to errors.

2.3 Hidden Markov Models (HMM)

In this section, we briefly introduce the concept of Hidden Markov Models (HMM). HMM is widely used in serial process modeling, such as speech synthesis.

Hidden Markov Models (HMM) are first introduced in [Baum] and later implemented for speech processing by Baker [Baker]. HMM is a discrete-time, discrete-space dynamical system that utilizes a Markov chain that emits a sequence of observable outputs: one output (observation) for each state in a trajectory of such states. The result is the output of a model for the underlying

process. Alternatively, given a sequence of outputs, HMM infers the most likely sequence of states. HMM can be used to predict a continue sequence of observations and also can be used to infer underlying states according to the outputs so that they are widely used in speech recognition.

Mathematically, HMM is a five-tuple $(\Omega\_x, \Omega\_O, A, B, \pi)$, where

$\Omega\_x = \{q_1, ...q_N\}$ is the finite set of possible states with N as total number of states;

$\Omega\_O = \{v_1, ...v_M\}$ is the finite set of possible observations with M as total possible observations;

$A = \{a_{ij}\}$ is the set of transition probabilities;

$a_{ij} = \Pr(X\_t + 1 = q_j \,|\, X\_t = q_i)$ is the transition probability from state i to j;

X_t is the state at time t;

$B = \{b_i\}$ is the set of observation probabilities;

$b_i = \Pr(O\_t = v_k \,|\, X\_t = q_i)$ is the observation probability of v_k when state is i at time t;

O_t is the observation at time t;

$\pi = \{\pi_i\}$ is the initial state distribution;

$\pi_i = \Pr(X\_0 = q_i)$ is the probability of that the initial state is q_i

Chapter 3 Application Scenario 1. Sound Texture Modeling

In this chapter we present an application of sound modeling/synthesis in specific sound texture modeling. We use a Poisson distribution to simulate the occurrence of events and use time-frequency linear predictive coding (TFLPC) to capture both the time and frequency spectrum contour inside each event. This method is applicable to non-regular distributed transient-events texture, such as the crackling of fire sounds.

3.1 Problem Statement

Sound textures are sounds for which there exists a window length such that the statistics of the features measured within the window are stable with different window positions. That is, they are static at “long enough” time scales. Examples include crowd sounds, traffic, wind, rain, machines such as air conditioners, typing, footsteps, sawing, breathing, ocean waves, motors, and chirping birds. Using this definition, at some window length any signal is a texture, so the concept is of value only if the texture window is short enough to provide practical efficiencies for representation. Since all the temporal structure exists within a determined window size, if we have a code to represent that structure for that length of time, the code is valid for any length of time greater

than the texture window size.

If a statistical description of features is valid, (e.g. the density and distribution of “crackling” events in a fire), the variance in the instantiations for a given parameter value would be semantically equivalent, if not perceptually so. That is, one might be able to perceive the difference between two reconstructed texture windows since the samples have a different event pattern, but if density is the appropriate description of the event pattern, then the difference is unimportant. We must thus identify structure within the texture window that can be represented statistically as well as structure that must be deterministically maintained.

Texture modeling does not generally result in models that cover a particularly large class of sounds. It is more appropriate for generating infinite extensions with semantically irrelevant statistical variation than it is at providing model parameters for interactive control or for exploring a wider space of sound around a given example.

In this Chapter, we focus on synthesizing continuous, perceptually meaningful audio stream based on single audio example. The synthesized audio stream is perceptually similar to the input example and not just a simple repetition of the audio patterns contained in the input. The synthesized audio stream can be of arbitrary length according to the needs.

3.2 Review of Existing Techniques

Sound texture modeling is a comparatively new research area and no much works

has been done in this area, although the corresponding topic in graphic research area, graphic texture analysis, has been studied for many years. According to our survey, almost all the methods utilize some statistical feature to model sound texture.

Generally, different time frames are used for texture analysis. The texture window length is signal-dependent, but typically on the order of 1 second. If the window needs to be longer in order to produce stable statistics when time shifted, then the sound would be unlikely to be perceived as a static texture. An LPC analysis frame is typically on the order of 10 or 20 ms. The frequency domain LPC (FDLPC) technique, which is an important part of our system, is called “temporal wave shaping” in its original context [Herre], and it specifies the temporal shape of the noise excitation used for synthesis on a sub-frame scale.

Tzanetakis and Cook [Tzanetakis] use both analysis and a texture window. In recognition that a texture can be composed of spectral frames with very different characteristics, they compute the means and variances of the low-level features over a texture window of one second duration. The low level features include MFCCs, spectral centroid, spectral rolloff (the frequency below which lays 85% of the spectral “weight”), spectral flux (squared difference between normalized magnitudes of successive spectral distributions) and time-domain zero crossing. Dubonov [Dubnov] used a wavelet technique to capture information at many different time scales. St. Arnaud [Arnaud] developed a two-level representation corresponding roughly to sounds and events, analogous to Warren and Verbrugge’s “structural” level [Warren1988] describing the object source and the “transformational” level corresponding to the pattern of events caused by breaking

and bouncing.

One of the objectives in model design is to reduce the amount of data necessary to represent a signal in order to better reveal the structure of the data. The TFLPC approach achieves a dramatic data reduction with minimal perceptual loss for a certain class of textures. Athineos and Ellis [Athineos] used this representation to achieve excellent parameter reduction with very little perceptual loss using 40 Time Domain LPC (TDLPC) coefficients and 10 Frequency Domain LPC (FDLPC) coefficients per 512-sample or 23ms frame of data resulting in a 10x data reduction. In this process, the compression is lossy although perceptual integrity is preserved and the range of signals for which this method works is restricted. This is a coding method rather than a synthesis model, although it achieves excellent data reduction. We can not, for example, generate perceptual similar sounds of arbitrary length using this method, which greatly restricts the applications.

To construct a generative model, we want to connect the Time domain (TD) signal representation to a perceptually meaningful low-dimensional control. We have hope of doing this because the signal representation is already very low dimensional. We still need to "take the signal out of time" by finding the rules that govern the progression of the frame data vectors.

3.3 Certain Sound Texture Modeling

3.3.1 System Framework

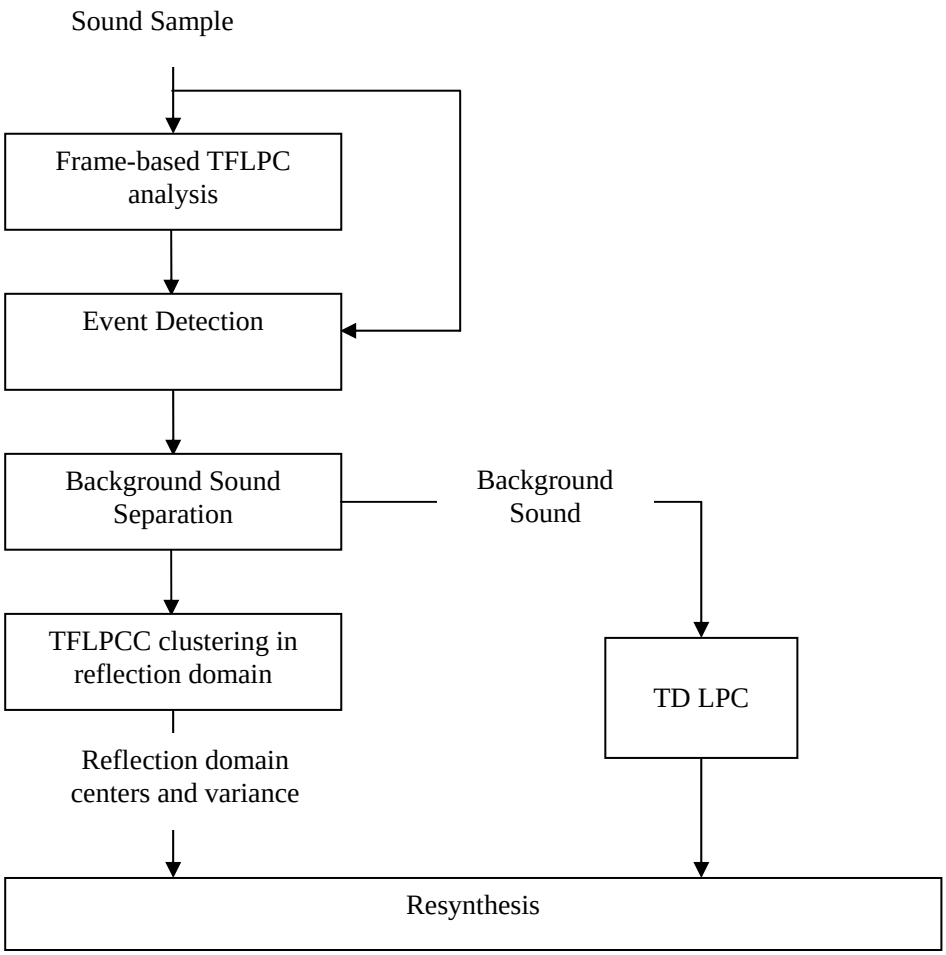

Figure 3.1 Texture Modeling System Framework

The framework of the system is shown in Figure 3.1. There are five basic steps in the framework: frame-based TFLPC analysis, event detection, background sound separation, TFLPCC clustering in reflection domain and resynthesis. The first four steps are the process of modelling the sound texture, and the last step is to synthesize sound of arbitrary length.

3.3.2 Frame Based TFLPC Analysis

A frame-based time and frequency domain LPC analysis is first applied to the sound for further event extraction and reflection domain clustering, as shown in Figure 3.2. Such an analysis is essentially the same as the method in [Athineos].

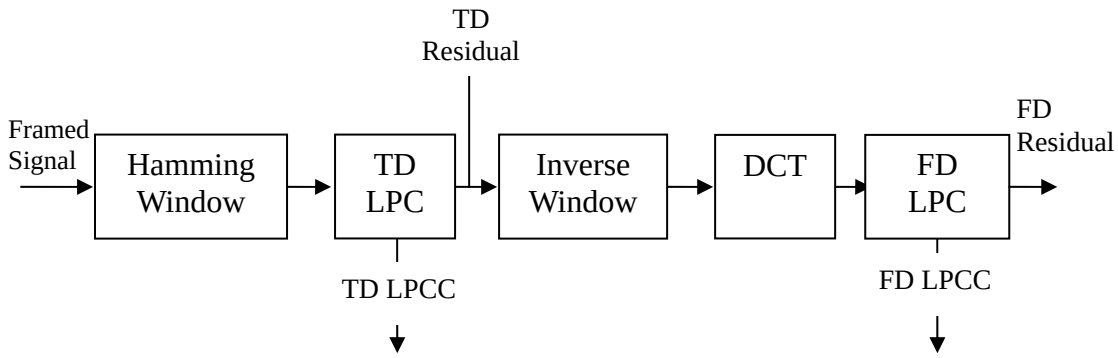

Figure 3.2 TFLPC Analysis

Each frame in the signal is first multiplied by a hamming window. Following the time domain linear prediction (TDLP), 40 LPC coefficients and a whitened residual are obtained. Then the TD residual is multiplied by an inverse window to restore the original shape of the frame. We use a discrete cosine transform (DCT) to get a spectral representation of the residual and then apply another linear prediction to this frequency domain signal. This step is called frequency domain linear prediction (FDLP), which is the dual of TDLP in frequency domain. We extract 10 FDLPC coefficients for each frame.

3.3.3 Event Detection

The detection of events is shown in Figure 3.3.

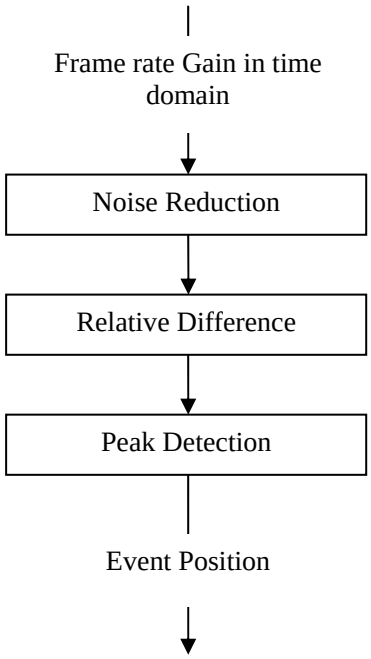

Figure 3.3 Event Extraction

The gain of time domain LPC analysis in the frame-based TFLPC indicates the energy of frames so that it can be used to detect events. The gain is first compared with a threshold (20% of the average of the gain over the whole sound sample) to suppress noise and small pulses in gain. A frame-by-frame relative difference is calculated and the peak position of the result is recorded as the onset of an event. To detect the offset of each event, we use the average of the gain between adjacent event onsets as an adaptive threshold. When the event gain is less than the adaptive threshold, the event is considered as over. The length of most events in our collection of fire sounds vary from 5-7 overlapped frames, or 60-80ms.

The event density over the duration of the entire sound is calculated as a statistical

feature of the sound texture and this density is used in synthesis to control the occurrence of events.

3.3.4 Background Separation

After we segment out the events, we are left with the background sound we call ‘din’ containing no events. We concatenate the individual segments and pre-emphasize the high frequency part and then apply a 10-order time domain LPC filter to this background sound to model it. The pre-emphasis is to better capture the high frequency character. The TDLPC coefficients we obtain here are used to reconstruct the background sound in the resynthesis process.

3.3.5 TFLPC Coefficients Clustering

In this step, we cluster the TDLPC coefficients and FDLPC coefficients to further reduce the data amount. The process is as follows.

1) We first transform each of the TDLPC coefficient (TDLPCC) and FDLPC coefficient (FDLPCC) vectors into the reflection domain. The filters represented by the LPC coefficients are not generally stable under perturbation [Atal], so such a transform is necessary.

2) Then we determine the number of clusters of TDLPCC and FDLPCC separately. This is an issue of validity in unsupervised clustering. Here we use the K-means method in clustering and the criterion function of minimization of ratio of within-cluster scatter-matrix’s norm and total scatter matrix’s norm [Halkidi] to

determine the proper cluster number.

The criteria function is defined as:

$$F = \frac{\det(S_w)}{\det(S_T)} \tag{1}$$

where $S_w = \sum_{i=1}^{c} \sum_{x \in X_i} (x - m_i)(x - m_i)^T$ is the within-cluster scatter matrix, X_i is the i^{th} cluster, c is the total number of clusters, $m_i = mean(x \mid x \in X_i)$ is the mean vector of the i^{th} cluster; $S_T = \sum_{x} (x - m)(x - m)^T$ is the total scatter matrix; *m=mean(x)* is the mean of all the vectors. We limit the number of cluster to be in the range from 2 to 20 and then calculate the criterion function F for different candidate cluster numbers in this range. Then we calculate the change rate of F with increase of cluster number c. When the change rate is very small (less than 1/1000), which means the criteria function changes slowly, the current number is considered as the optimal one.

3) The center vector and variance of each cluster is calculated and recorded for resynthesis. Based on the assumption that each dimension of the LPCC vector is independent, we calculate the variance of each dimension separately so that we get a variance vector for each cluster. Instead of the original frame-based TFLPCC sequence of each event, the cluster index of each sequence and the cluster center and variance are recorded. We also record the time domain LPC gain sequence and the cluster number sequence of each event for resynthesis. We record these parameters to preserve the original order of frames, which is critical in our system.

3.3.6 Resynthesis

In the resynthesis process, we generate the background sound and event sequence separately and mix them together in the final step.

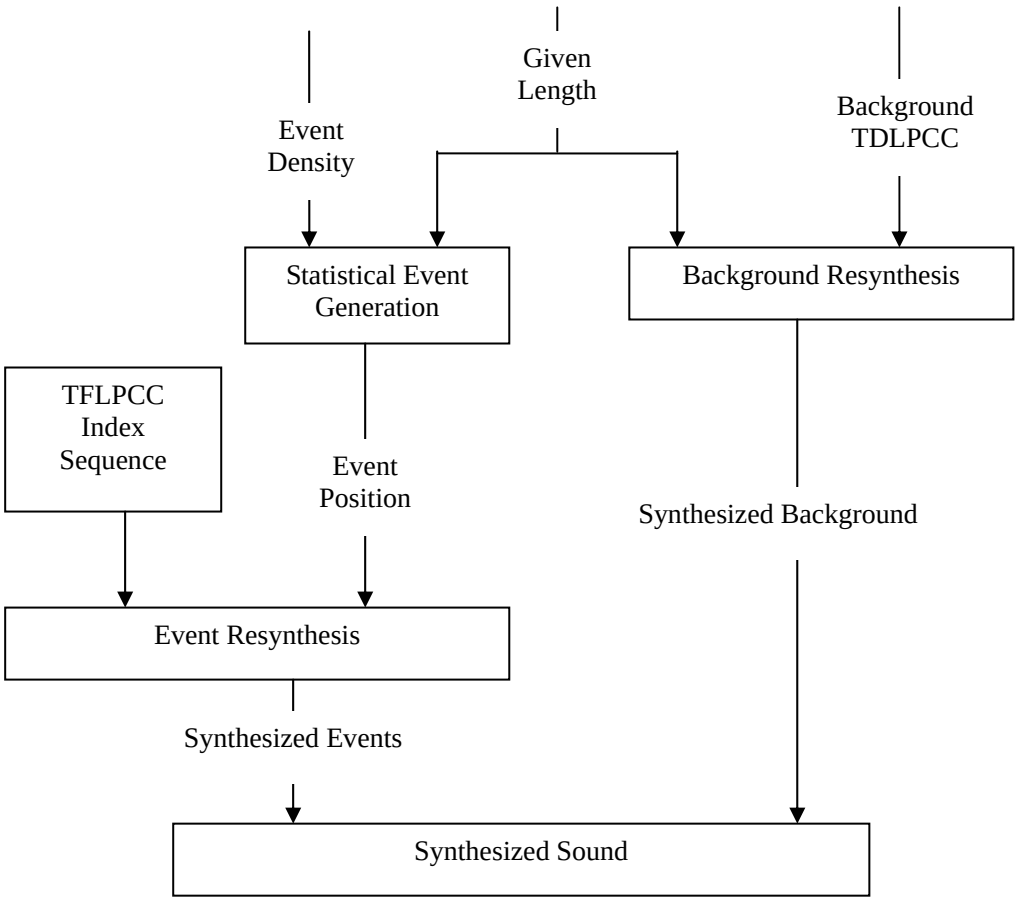

Figure 3.4 Resynthesis

Given a desired sound length, we use a noise excited 10-order background TDLPC filter to generate the background sound and de-emphasize the high frequency part. For the foreground sound, the resynthesis process is shown in

Figure 3.4 and described below.

1) Use the event density, which is the average number of events per second, as the parameter of a Poisson distribution to determine the onset position of each event in the resynthesized sound.

2) Randomly select an event index. According to the TFLPCC sequence, use the reflection domain TFLPCC cluster centers and the ½ of the corresponding variance as the parameters to a Gaussian distribution function in each dimension to generate the reflection domain TFLPCC feature vector sequence for the event. Here we multiply a factor of ½ to the variance to make sure the generated LPC coefficients do not differ too much from the originals.

3) Transform the reflection domain coefficients into the LPC domain.

4) Do the inverse TFLPC. This is just a reverse process of the TFLPC analysis, as shown in Figure 3.5.

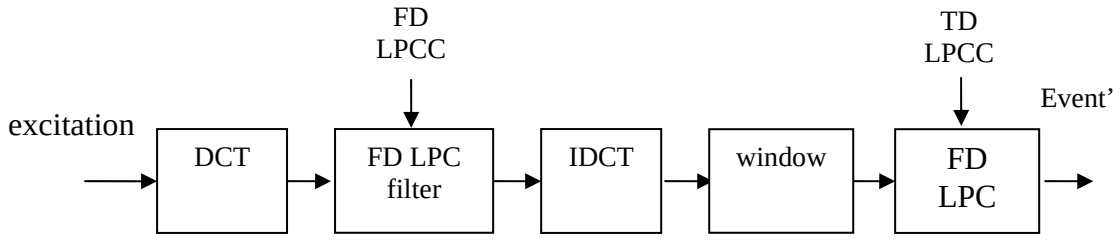

Figure 3.5 TFLPC Synthesis

We first get the DCT spectrum of the excitation signal and then filter it using the FDLPC coefficients to get the excitation signal in the time domain. Figure 3.6 shows the residual and the regenerated excitation in time domain. FDLPC captures the sub-frame contour shape well. Then we filter the time domain excitation using the TDLPC filter to get the time domain frame signal.

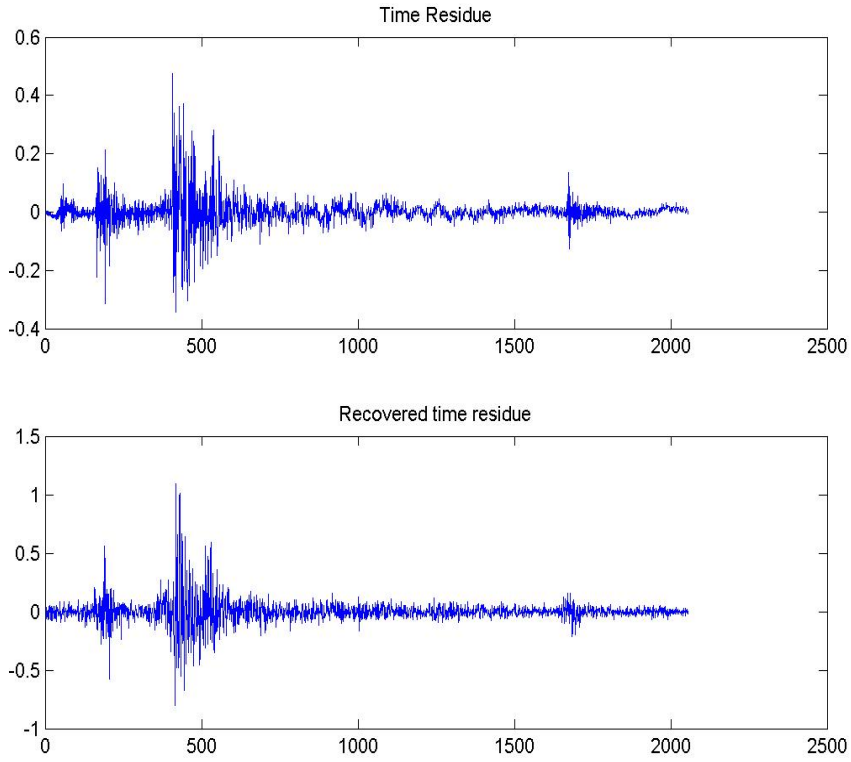

Figure 3.6: Time domain residual (above) and recovered excitation signal (below). Here we plot 7 overlapping frames to show the structure of one event.

5) Repeat step 4 for all the frames inside one event and then overlap and add to reconstruct the event.

6) Repeat 2-5 until we generate events for all the event positions.

7) Mix the synthesized events and the background sound together to get the final result. The result is shown in Figure 3.7.

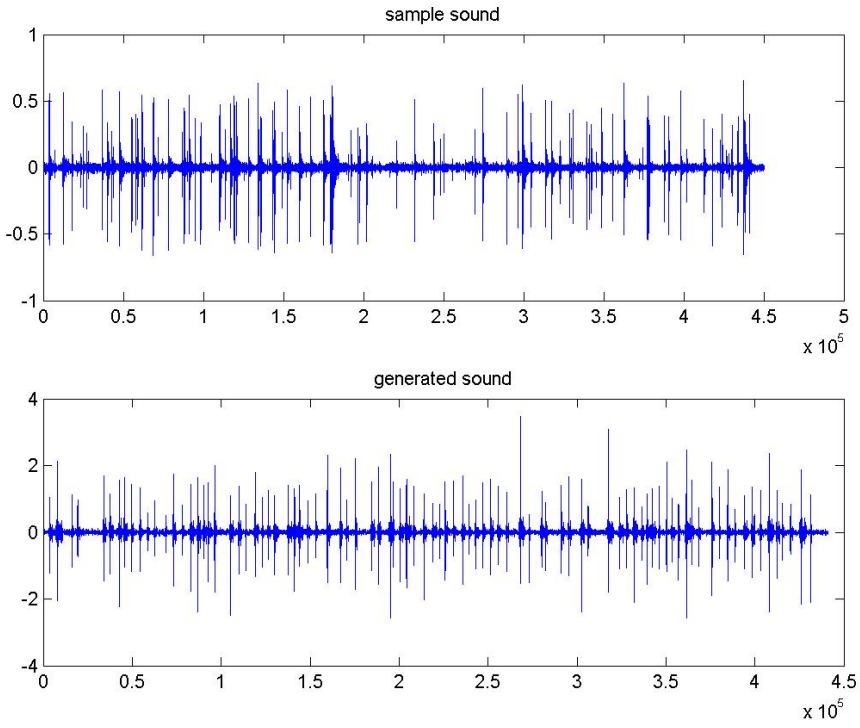

Figure 3.7: Sample sound (above) and synthesized sound (below).

3.4 Evaluation and Discussion

Informal listening tests show that the regenerated sounds capture some texture characters of the original audio clips. By using frame level contour extraction and TFLPC analysis, both the spectral and fine temporal characteristics of the sound are captured. To listen to and compare the original sound with the generated one, see http://www.zwhome.org/~lonce/Publications/dafx2004.html.

The error for each generated transient event comes from two sources: one is the error between the excitation signal and the original residual; another is the difference between the generated LPCC and the original one due to the clustering. It is not easy to quantitively measure the dissimilarity between the generated sound and the sample audio principally due to the statistical variation in the model.

3.4.1 Properties of Reflection Domain Clustering

In the clustering of the TFLPCC, we use the reflection domain coefficients instead of LPC domain coefficients. The reflection domain coefficients have several advantages compared to the LPC domain coefficients [Atal]. Some of the advantages are:

1) the all pole filter is stable under perturbation provided that the corresponding reflection coefficients all lie between -1 and +1,

2) interpolating between two of reflection coefficients yields a smooth change in the frequency response.

Figure 3.8 shows how the frequency response changes when we scale the reflection domain coefficients. The first plot is the frequency response of a time domain reflection coefficients. The second plot is the frequency response of the normalization of the coefficients whose norm is 1. The last plot is the frequency response of scale factor 0.01 multiplies the original coefficients. The figure shows that when the maximum component of reflect coefficient vector is much smaller

than 1, rescaling the coefficient vector does not change the frequency response of the LPCC much. In other words, such a change in the frequency response is acceptable and our clustering algorithm can be independent of the vector magnitude.

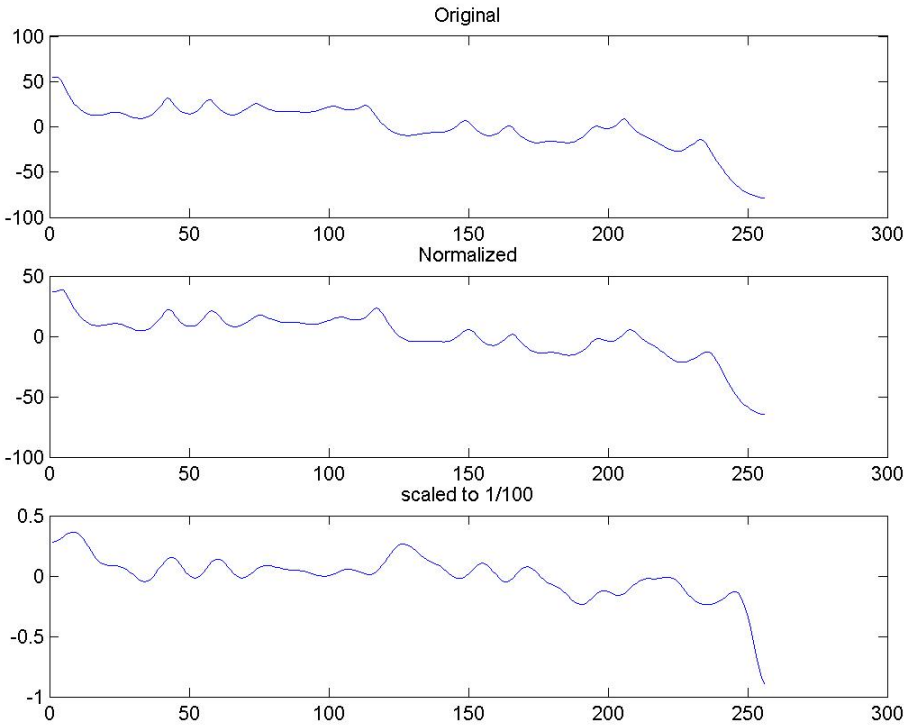

Figure 3.8: Scale effect of reflection LPC coefficients.

3.4.2 Comparison with an HMM Method

We introduce the basic knowledge of HMM in Chapter 2. In the framework in Section 3, we cluster the reflection domain TDLPCC and FDLPCC into clusters separately and record the TDLPCC and FDLPCC cluster index sequences for each event to preserve the original order of the frames. By such a clustering we get two "codebooks" of the TDLPCC and FDLPCC separately and greatly reduce the amount of information in reconstruction. However preserving the specific cluster

number sequence for each event also restricts the flexibility of modeling. To gain more flexibility, we train Gaussian Mixture Models to capture the order pattern.

After we get the reflection domain TDLPCC and FDLPCC sequence for each event as we do in Section 3.3, we use these TDLPCC sequences and FDLPCC sequences of all events to train two Gaussian mixture HMMs for TDLPC and FDLPC separately. In resynthesis, we use these two HMMs to generate the reflection domain TDLPCC sequence and FDLPCC sequence for regenerated events. However, result shows that such a system does not work well. The output of the HMMs sounds much more noisy than the given samples. In the Gaussian Mixture HMM, there are several possible Gaussian distributions for each state. When we generate coefficient vectors using the HMM, these distributions are chosen according to some probability. The randomness in the cluster sequence has a significant detrimental affect on the perceptual quality of the regenerated sound

3.4.3 Comparison with Event-Based Method

As another approach to reduce the amount of data, we implement a system using TFLPC analysis to entire events instead of overlapped frames. First the energy of each frame is calculated and then we extract events from the energy sequence of the whole sound as we do to the gain sequence in section 3.3. Next we apply TFLPC analysis to individual events instead of frames so that we have only one TDLPCC vector and one FDLPCC vector for each event compare with frame-based method's two vector sequences. The data amount is further reduced. However, there is a dramatic quality decrease when the event length is long. The

reasons are as follows:

1) When the event length increases, the modeling ability of LPC decreases. We can use a greater filter order, but the quality is still worse than the short window case,

2) The limited amount of data affects the parameter extraction for the Gaussian distribution of each cluster. We get only two LPC vectors for each event instead of two LPC coefficient vector sequences, so the data is not enough to estimate the proper Gaussian distribution parameters for each cluster.

Based on these reasons, among the several methods we implemented in our experiment, the frame-based TFLPC analysis method which is introduced in the system framework section worked the best.

Chapter 4 Application Scenario 2. Packet Loss Recovery

In this chapter, we present an application of sound modeling/synthesis in which we greatly reduce the redundant information required for recovery without obvious quality decrease. We achieve this goal by generating a model for the redundant information used in packet loss recovery and synthesizing the redundant information from model parameters and further regenerating the lost packets.

4.1 Problem Statement

Bandwidth efficiency and error robustness are two essential and conflicting requirements for streaming media content over error-prone channels, such as wireless channels. On such unreliable networks, packet loss can be common and arise in many different forms. For instance, packets can be dropped due to congestion at switches or arrive with too long a delay to be useful. On wireless networks, packet losses can be caused by channel characteristics, such as fading, or wireless network characteristics, such as handover in cellular network. However, it is important to guarantee user-perceived quality of service (QoS) for media streaming applications, especially music. For this purpose, we need a method to recover lost packets. The method should generate perceptually high

quality audio and use as little redundancy as possible to maintain bandwidth efficiency.

The objective of packet loss recovery in audio streaming is to reconstruct the lost packet with a perceptually indistinguishable replacer, or at least very similar. This is the requirement of the quality of the recovered audio packet.

Considering an established wireless standard, the maximum channel capacity is fixed and shared by many users. This fact adds the new requirement that the redundant information should be as few as possible, with the assumption that it would not affect the quality of recovered music.

In some systems, such as online music play or online gaming, there is also time requirement of recovery. Lost packet resubmission is generally not a good method in a system with critical time requirement due to its high latency. With consideration of such applications, there is the third requirement of the loss packet recovery method: it should be economical in computation so that the recovery process can be done with small latency. On the other hand we can consider the computational ability on the server side is unlimited, because the server generally has strong computation ability and there is no critical time requirement for server.

Based on the above consideration, we need to develop a packet lost recovery scheme with high recovered quality, small amount of redundant information and few amount of computation in receiver side.

4.2 Related Works

4.2.1 Packet Loss Recovery

There are many works related to packet loss recovery and these works can be categorized into two categories: sender-based and receiver-based. The sender-based methods can be further categorized into active retransmission and passive channel coding. The receiver-based methods usually are employed when the sender can not provide recovery information for the lost packets or the recovery schemes fail. In this section, we present a brief overview of these strategies.

Retransmission methods are closed-loop mechanisms that based on the retransmission of the packets that were not received at the destination. In some network protocols, such as Transfer Control Protocol (TCP), the retransmission is ensured: the protocol makes sure that all the frames in the original data arrives the destination. Many other protocols, such as User Datagram Protocols (UDP), do not ensure this point. Retransmission methods are not acceptable in many real time constrained applications such as live audio streaming, because it dramatically increases the end-to-end latency. Interactive audio applications have critical time requirements and the end-to-end delay need to be less than 250ms [Brady].

Most current sender-based methods belong to forward error correction (FEC), a kind of open-loop mechanisms based on the transmission of redundant information together with the original information so that the lost original data can be recovered from the redundant information. FEC is an attractive alternative for providing reliability without greatly increasing latency. This is particularly important for

applications with real time constraints over high speed networks [Shacham]. This kind of method usually achieves good performance with the cost of a large amount of redundant information [Wah]. The redundant information cost, that increases the bandwidth requirement of transmission, is the main drawback of FEC. FEC can be further categorized into media-independent FEC and media-dependent FEC, as discussed below.

Media-independent FEC usually uses block or algebraic codes to generate additional packets to recover lost original data packets. Each code takes a codeword of n data packets and generates k additional check packets so that the amount of transmission packets is $n+k$ for n original packets. An example using media-independent FEC is the exclusive-or (*XOR*) coding implemented in Rosenberg [Rosenberg96]. There are several advantages of media-independent methods. The first is that the operation of FEC does not depend on the contents of the packets, and the repair can be exactly done. Secondly the computation required to derive the error correction packets is small and simple to implement. The disadvantages are that it imposed additional delay, increase the bandwidth requirement of transmission.

Media-specific FEC extracts some characters from the content of the audio signal and uses these characters to recover lost packets. The transmitted original copy of the audio data is referred to as the primary encoding and the redundant transmissions are called secondary encodings. Usually the secondary encoding uses a lower-bandwidth and lower-quality encoding than the primary to save bandwidth. The choice of secondary encoding is usually depends on both the bandwidth and the computational complexity of the encodings and the application's requirement of the

quality of recovered audio. Erdol et al. [Erdol] use short-term energy and zero-crossing measurements as their secondary encoding. When a packet is lost, the receiver interpolates the audio signal about the crossings using the short time energy measurements. It is computationally cheap but can only recover short periods of loss because the measure is only a coarse feature of the original audio signal. Hardman et al. [Hardman] and Bolot et al. [Bolot] use the low-bit-rate analysis-by-synthesis codecs such as full rate GSM encoding. Media-specific FEC usually add a redundant overhead on each packet so that the size of each packet increases. A common used method is to add the redundant overhead of the packet on the next packet so that when the previous packet lost, it can be recovered from the next packet. The length of the overhead of media-specific FEC is variable, depending on the quality requirement of the repaired packets and without affecting the number of losses can be repaired.

Receiver-based methods usually recover the lost packet without any redundant information and is generally called error concealment. These kind of methods generally exploit correlations between the adjacent packets and is usually very simple and effective only when the packet loss rate is very low. Here we consider a primarily receiver-based method. With increasing computational resources and memory capacity, many receiver-based methods are becoming attractive. Based on the assumption that packet loss is infrequent, that packet size is small and that the signal is reasonably stationary for short enough segments, packet repetition can offer a good compromise between achieved quality and complexity [Perkins]. The assumption of stationarity, however, is not true for streaming music, particularly in the neighborhood of the musical “beat”. Furthermore, the simple packet repetition method produces a double-drumbeat effect [Wang2002] when

the missing packet immediately follows the beat, or fails to recover a beat when the missing packet is exactly on the beat. Listeners are much more sensitive to the errors due to packet repetition recovery when they occur around the beat than when they happen elsewhere.

In practice the error concealment methods usually do not work along. A sender-based scheme is used to repair most lost packets and the other gaps are left to receiver-based error concealment, which provides cheap and effective ways to recover the remain lost packet.

4.2.2 C-UEP Scheme

The failure of standard recovery techniques for this kind of signal led Wang *et. al.*[Wang2003] to a content-based method of error concealment which is called C-UEP, which means content-based unequal error protection. Recognizing the perceptual importance of the musical beat, they introduced a parametric vector quantization (PVQ) scheme as a secondary encoding of just the percussive sounds. This method, compared with the conventional techniques, provides a much higher quality of service (QoS), though there are still certain limitations.

The C-UEP scheme can be categorized as a forward error correction (FEC) method. It holds the high recovery quality advantage of the FEC and also partly overcomes the drawback of large amount of redundant information by using a codebook of drum beats instead of the original drum beats sounds. But there are still some limitations of the C-UEP system: First, the content-based codebook used for

recovery, which is sent to the receiver in a “header” segment prior to streaming the audio data, may be too large in application. Second, each codebook entry represents a whole class of transient events in a stream and the resulting approximations may simply not be good enough for some kinds of music. Furthermore, the drum beats used in clustering are directly extracted from the original music signals and may be contaminated with the singing voice and other noises.

4.3 Analysis/Synthesis Solution

4.3.1 System Framework

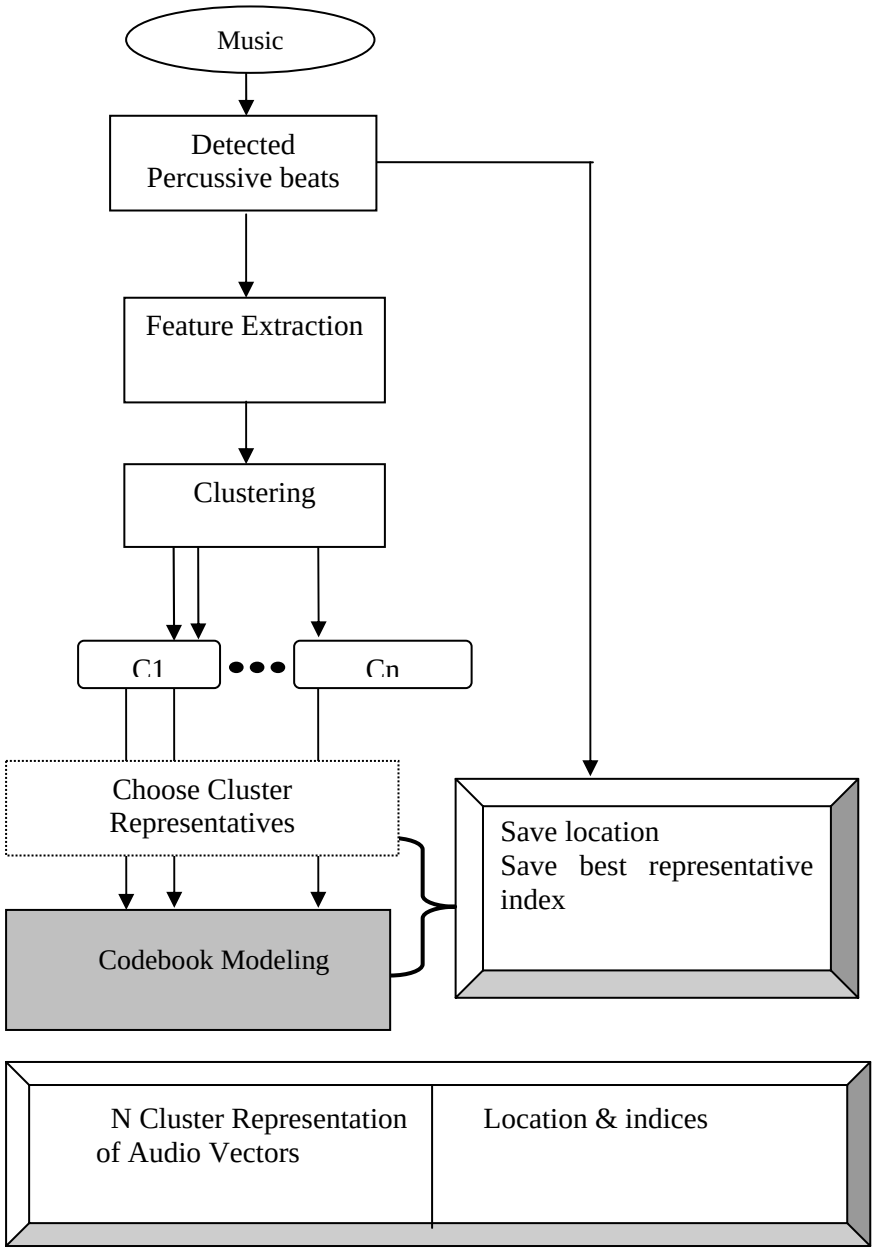

Figure 4.1 System Framework on server side.
The shaded parts are new contributions of this thesis.

The block diagram of the proposed packet loss recovery system framework on server side is shown in Figure 4.1. The shaded parts are contributions of this thesis.

Out system is an improvement of the previous C-UEP system in [Wang2002]. It is an unequal error protection scheme, with protection emphasis on the drum beats. Our work further reduces the amount of the redundant information by building a LPC model of the codebook items.

Our system aims to achieve low additional bandwidth. The current structure is based on the below consideration of network transmission:

1. The analysis of the music, including detecting and encoding percussive sound, will be done on the server side prior to transmission. Real-time analysis and resynthesis on mobile devices is currently impractical given computational resources.

2. To recreate percussive sound, we need a synthesis model on the client side. One alternative would be to create a percussive model for each piece of music and transfer the model to client side before music streaming. Another possibility would be to assume a single general model on the client side that can generate percussive sounds for any piece. The former way may generate better quality sound but would require more bandwidth. It is also more difficult to generate a model automatically than it is to parameterize one.

3. What is the optimal degree of data reduction via vector quantization to perform to create the transients codebook? Wang *et al*.[Wang2003] found four vectors to be adequate for a substantial increase in perceived quality. If the codebook entries are small enough, there would be less pressure to sacrifice quality with such a drastic reduction in the number of entries.

4. The client side resynthesis of the audio codebook could be done either prior to streaming, or in real-time on an as-needed basis. If the computation ability on the client side cannot support real time calculation, a pre-stored replacement vector buffer is necessary for recovering lost packets.

Based on the above consideration, we use the current scheme in codebook modeling progress.

There are six basic components in the framework: percussive sounds detection; codebook selection; codebook modeling; transmitting the codebook; synthesizing percussion sounds; reconstructing the lost packet. Percussive sounds detection, codebook selection and codebook modeling are done on sever side, which is shown in Figure 4.1; synthesis and reconstruction are done on the receiver device. In the following sections, we will give a detailed description of each component.

4.3.2 Percussive Sounds Detection

Our beat detection process first detects the onsets in the music streaming using sub-band processing [Wang2003]. Percussive events are detected by looking for sudden increases in intensity across several sub-bands.

4.3.3 Codebook Vector Quantization

After the transient segments are extracted, they are clustered according to a set of perceptual features, and a single vector from the center of each cluster is chosen as a representative for the codebook. In [Wang2003], the codebook and indices make

up a “header” segment to the audio file that is sent prior to streaming audio. Since the audio vectors in the codebook dominate the size of the header segment, our focus in this paper is on reducing the size of the codebook.

4.3.4 Codebook Modeling

To reduce the size of the audio vector codebook, we use a generative model of the audio vector with a small number of parameters used to control the model in resynthesizing the vector on the client.

Currently we use a single percussive sound synthesis model for all audio vectors. The task of the analysis/synthesis system is to minimize the perceptual difference between the original and the resynthesized audio (Figure 4.2).

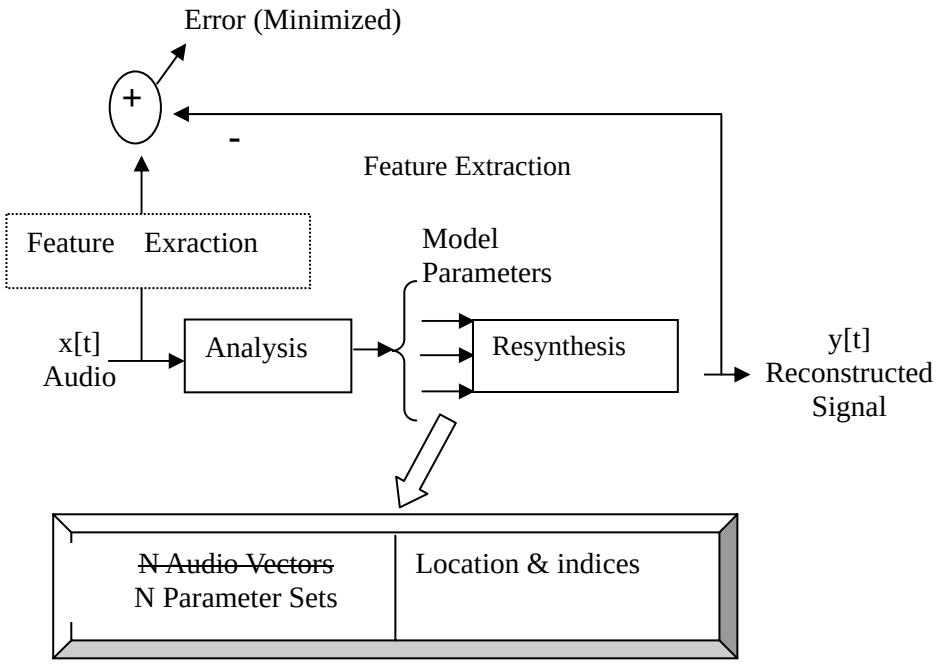

Figure 4.2. Codebook modeling and synthesis.
This unit represents the codebook modeling part of Figure 4.1 and the synthesis progress on receiver side.

We model the transient audio vectors as a signal containing a mix of noise and periodic information with a single broad spectral shape. The only time-varying component of the model is the amplitude attack and decay. The analysis is done in the following steps:

1. Extract the contour of the percussive event (Figure 4.3). We find the maximum point of the signal and use this point as the vertex of the contour triangle. The duration of the codebook vectors is fixed and currently 2304 PCM samples. We also keep the total energy of the percussive sound as a parameter for resynthesis.

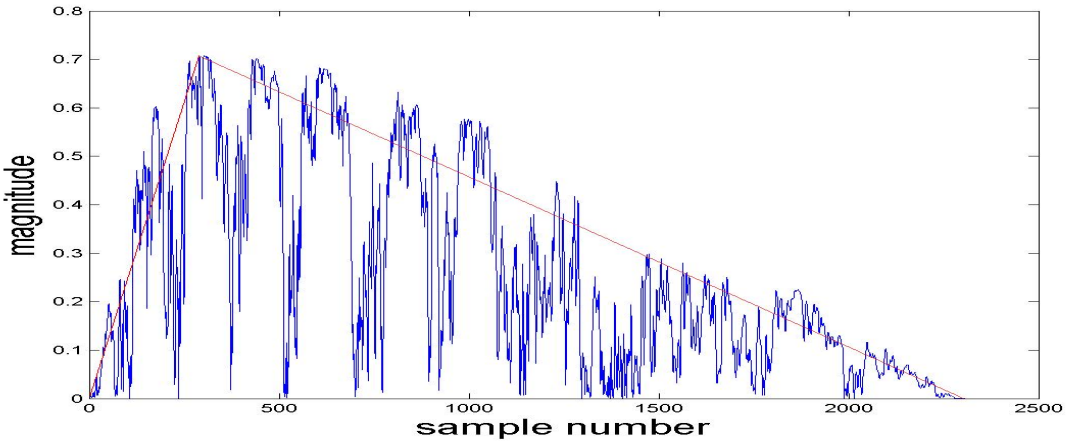

Figure 4.3 Event Contour

2. Next we model the overall spectral shape of the vector using standard Linear Predictive (LPC) analysis. Here we set the number of coefficients to 12 to capture only the coarse spectral structure. With the residual error signal from the LPC analysis, it would be possible to exactly regenerate the original signal.

3. Next we model the residual (Figure 4.4) as a pitched signal plus white noise. We derive a pitch estimate by taking an autocorrelation of the FFT-derived power spectrum. We take the pitch to be that of the maximal peak of the autocorrelation in the range of 100-500 Hz. We use the ratio of the peak to total power in the spectrum as a measure “pitch salience”, similar to Slaney [Slaney].

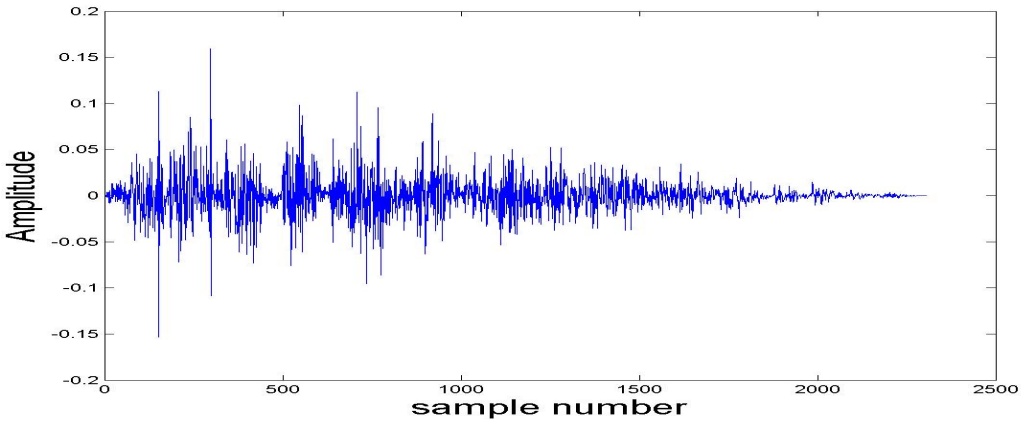

Figure 4.4 Residual of the LPC process

By this procedure, we have converted the audio vector codebook to a set of parameters, one set for each original audio vector. The number of parameters used in this method is 16: 12 LPC coefficients, amplitude peak time and total energy of the percussive sound, pitch and pitch salience.

4.3.5 Transmission of Parameter Codebook

The parameter codebook, together with the indices of transient packets, is sent in a header segment before the streaming of audio packets begins. The header is sent using a reliable transmission method.

4.3.6 Synthesize the Percussive Sounds

When the client receives the parameter sets in the header segment of the song data, it regenerates the percussive audio vector codebook.

The synthesis process has following steps: First, synthesize the residual. We use white noise as the source and apply a comb filter with a delay corresponding to the pitch. The pitch salience parameter is used to determine the filter weights – the relative balance between the delay tap and white noise. The pitch parameter is then used to amplitude modulate the noisy signal with a sharp attack and exponential decay at the pitch period, while the pitch salience parameter is used to control the decay rate - a longer decay rate makes the amplitude modulation less pronounced and the signal less pitched. Next, we recover the course spectral shape using the LPC-derived filter. Finally, we generate the temporal amplitude contour from the peak time and level and apply the contour to the regenerated signal. Then we normalize the regenerated signal so that it has the same energy as the original percussive sound.

The exact methods of analysis and synthesis are not important as long as the key perceptual characteristics (pitch, noisiness, spectral shape, signal energy and amplitude envelope) are similar to the original.

4.3.7 Reconstruct the Lost Packets

With the reconstructed audio vector codebook, the packet loss recovery process can proceed exactly as in [Wang2002]. When a packet is detected as lost, if it comes from a segment labeled as transient, it is replaced using one of the codebook entries (Figure 4.5).

If there is no transient in the lost packet, standard methods using neighboring frames are used to do the recovery work.

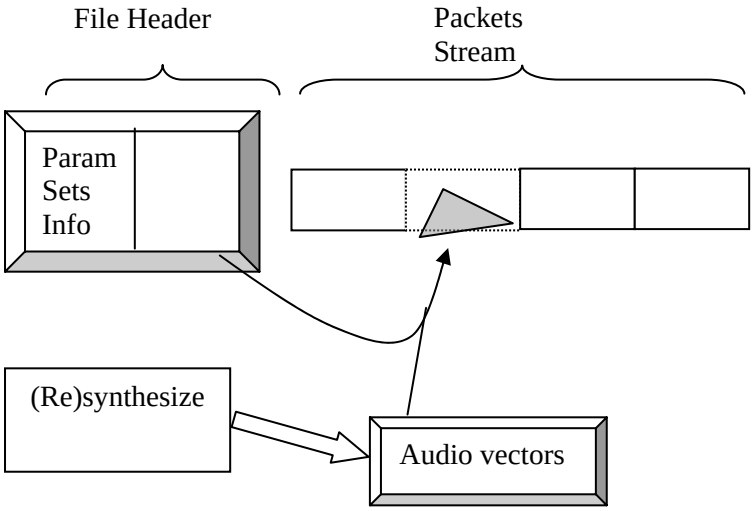

Figure 4.5 Reconstruction of a lost packet containing a percussive transient, which is denoted as a grey triangle.

4.4 Evaluation and Discussion

The analysis/synthesis method can greatly reduce the codebook data needed to recover the lost transients. For example, consider a 16 item codebook where the duration of each entry represents 2048 PCM samples. Using audio vectors as in [Wang2003], we need 64K bytes of redundancy data. Using a codebook of 16 synthesis parameters for each entry as described herein, the total codebook size in the head packet is only 2*16*16 = 512 bytes, a reduction of two orders of magnitude.

Heard in isolation, the resynthesized codebook vectors sound similar, but do not sound identical to the original codebook methods, and since the synthesis parameters are derived from the original codebook vectors, we can’t expect the

synthetic method to be a perceptual improvement. However, because the original method uses only a relatively small number of audio vectors to represent all transients in the music anyway, the difference between the synthetic and original vectors does not generally lead to a significant difference in the perceived quality in the context of the music. Both methods address the perceptual sensitivity to beats that has not been addressed by other recovery methods.

Sound examples for comparison can be found at www.zwhome.org/~lonce/Publications/ACM2003.html. The examples include the original song excerpt with missing packets, and their recreations using the simple repetition method, the PVQ method and the parametric method. To get better results, we can increase the vector number of the codebook, or even build one parameter vector for each transient in the music, without being a burden of transmission bandwidth.

Chapter 5 Conclusions and Future Work

In this thesis we present applications of sound modeling/synthesis in sound texture modeling and packet loss recovery. In both applications we highlight the benefits of building sound model for specific class of sounds to gain data reduction and variety.

In sound texture modeling we had demonstrated a method for modeling certain classes of sound textures. The method involves analysis at different time scales to preserve perceptually relevant information for synthesis. Future work will focus on improvement of quality and generalization of this method to a wider class of sounds. Currently we use a frame-based TFLPC analysis. If we could capture the order pattern of the frames inside events, we could build pattern models to gain more flexibility. In the current system we assume all the events are of the same kind and use a single Poisson distribution to simulate the occurrence of the events. This assumption may be violated for some sounds, such as the sound from tennis game containing the players' footstep sound and the ball-hitting sound. By classifying the events into different classes and using different statistical distributions for sequencing them, we can build a better model for the sounds containing more than one kind of event.

Some sounds with both broadband noise and densely-packed micro-transients are very difficult to segment into individual transient events. It is difficult to get global statistical features such as event density to control the resynthesis.

Segmentation of such complex sounds should also be explored to generalize this method for flexible resynthesis.

In the application of packet loss recovery system, we showed how the current state-of-the-art content based audio codebook method of packet loss recovery can be vastly improved in bandwidth requirements using synthetic modeling and synthesis without sacrificing perceived quality of service. The modeling and resynthesis approach scales up nicely. Given the existence of a synthesizer on the client, models (code that calls synthesizer library functions) are small. Two kilobytes is typical, smaller than the size of a single 46 ms audio packet. This means that several very different models for classes of sounds (different algorithms, different parameterizations) could be used for a wider variety of sounds than just percussive transients.

Future work will mainly focus on quality enhancements. A common situation in music streaming is that the percussive sounds do not occur alone but are mixed with other sounds concurrently, especially singing voice. This “contamination” affects the codebook quantization process and degrades recovered sound quality. For example, it is possible to recover a percussive packet containing male singing voice with another packet containing a female singing voice. Another possible situation is that the lost packet has a clear pitch, and there are no codebook entries with matching pitch due to quantization step. To provide a good match across a range of pitches, we could increase the number of codebook entries and still use less header bandwidth compared to the audio vector codebook with only a couple of entries. The analysis/resynthesis system affords good flexibility for addressing both quality and bandwidth issues.

Another possible way to enhance the quality without increasing redundant information is that we separate the percussive sounds out from the mixture before quantization. Statistically the percussive sound and singing voice can be considered as independent and we can apply the independent component analysis (ICA) technologies to separate them. Although the independent components from ICA do not directly correspond to the sources one-by-one, we can group components to build such a corresponding relationship and generate the sources from components groups. We can generate better codebook by eliminating singing voice source. The separated percussive sources are helpful in the quantization process if we can classify them from individual packets into specific classes.

The use of synthetic sound offers a combination of extremely low bandwidth requirements and real-time flexibility. It provides many options for managing computational and bandwidth/memory constraints and we expect it to be useful in a growing number of device and application contexts.

Bibliography

[Arfib] D. Arfib, F. Keiler, and U. Z¨olzer. Source-filter processing. In U. Z¨olzer, editor, Digital Audio Effects, pages 299--372. John Wiley and Sons, Ltd., Chichester Sussex, UK, 2002

[Arnaud] N.St. Arnaud, K. Popat, “Analysis and synthesis of sound textures,” in *AJCAI workshop on Computational Auditory Scene Analysis*, 1995.

[Atal]B.S. Atal, P.V. Cox, P. Kroon, “Spectral quantization and interpolation for CELP coders”, *International Conference on Acoustics, Speech, and Signal Processing*, May 1989.

[Athineos]M. Athineos, D.P.W. Ellis, “Sound texture modeling with linear prediction in both time and frequency domains,” in *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 2003.

[Baker] J.K.Baker, “The dragon system-An Overview”, IEEE Trans. Acoust. Speech Signal Processing, vol. ASSP-23, no.1, pp.24-29, Feb.1975.

[Baum] L.E.Baum, T.Petrie, “ Statiscial inference for probabilistic functions of finite state Markov chains”, Ann. Math. Stat., vol.37, pp.1554-1563, 1966.

[Brady] P.T.Brady, “Effects of transmission delay on conversational behavior on echo-free telephone circuits”, Bell Sys. Tech. J., Vol. 50, pp.115-134, Jan., 1971.

[Chen] Y.L.Chen and B.S.Chen, "Model based multirate representation of speech signals and its application to recovery of missing speech packets", IEEE Trans. Speech and Audio Processing, Vol.15, No.3, pp.220-231, May 1997.

[Comon] P.Comon, "Independent component analysis - a new concept?", Signal Processing, Vol.36, pp. 287-314, 1994.

[Deering] S.Deering, "Multicast Routing in a Datagram Internetwork", Ph.D thesis, Standford University, Palo Alto, CA, Dec.,1991.

[Dubnov] S. Dubnov, Z.B. Joseph, R. E. Yaniv, D. Lischinski, M. Werman, "Synthesizing sound textures through wavelet tree learning," *IEEE CGA*, vol. 22, no. 4, pp. 38– 48, Jul/Aug 2002.

[Erdol] N.Erdol, C. Castelluccia and A. Zilouchian, "Recovery of missing speech packets using the short-time energy and zero-crossing measurements", Trans. Speech and Audio Processing, vol. 1, No.3, pp. 295-303, Jul., 1993.

[Goodman] D.J.Goodman, "Waveform substitution techniques for recovering missing speech segments in packet voice communications", IEEE Trans. Acoustics, Speech, and Signal Processing, Vol.ASSP-34, No.6, pp. 1440-1448, Dec. 1986.

[Gruber] J.G.Gruber and L.Strawczynski, "Subjective effects of variable delay and clipping in dynamically managed voice systems", IEEE Trans. Commun., Vol. COM-33, No.8, pp.801-808, Aug., 1985.

[Halkidi] M. Halkidi, Y. Batistakis, M. Vazirgiannis, "On Clustering Validation Techniques", Journal of Intelligent Information Systems, 2001.

[Haykin] S. Haykin, “Neural Networks: A Comprehensive Foundation”, Prentice Hall, 1999.

[Herman] Hermann von Helmholtz, “On the Sensations of Tones”, translated by Alexander J.Ellis, New York 1954, pages 124-127.

[Herre] J. Herre and J.D. Johnston, “Enhancing the Performance of Perceptual Audio Coders by Using Temporal Noise Shaping (TNS),” in *Proeedings of. 101st AES Conference*., Nov 1996.

[Hyvärinen97_1] A.Hyvärinen and E. Oja., “A fast fixed-point algorithm for independent component analysis”, Neural Computation, Vol 9, No.7, pp1483-1492, 1997.

[Hyvärinen97_2]A. Hyvärinen, “A family of fixed-point algorithms for independent component analysis”, ICASSP,1997.

[Hyvärinen99]A. Hyvärinen. “Fast and robust fixed-point algorithms for independent component analysis”, IEEE Trans. on Neural Network*s*, 1999.

[Jayant] N.S.Jayant and S.W.Christenssen, “Effects of packet losses in waveform coded speech and improvements due to an odd-even sample-interpolation procedure”, IEEE Trans. Commun., Vol.COM-29, No.2, pp.101-109, Feb.,1981.

[Jolliffe] I.T.Jolliffe. “Principal Component Analysis”, Springer-Verlag, 1986.

[Karplus] K. Karplus, A. Strong, “Digital Synthesis of Plucked-String and Drum Timbres ”, Computer Music Journal, vol.7, no.2, 1983.

[Kay] S.M. Kay, “Modern Spectral Estimation, Englewood Cliffs”, NJ, Prentice-Hall, 1988.

[Kavcic] A. Kavcic, B. Yang, “A new efficient subspace tracking algorithm based on singular value decomposition”, ICASSP, vol.4, pp.485-488, 1994.

[Kendall] M.Kendall, “Multivariate Analysis”,Charles Griffin&Co., 1975.

[Makhoul] J. Makhoul. “Linear prediction: A tutorial review”, Proceedings of the IEEE, vol. 63, no.4, pp. 561--580, 1975.

[Manning] Manning, Peter, “Electronic and Computer Music”, Clarendon Press, 1985.

[Papoulis] A. Papoulis, “Probability, Random Variables, and Stochastic Processes”, McGraw-Hill, 3rd edition, 1991.

[Perkins] C. Perkins, O. Hodson, V. Hardman, “A Survey of Packet Loss Recovery Techniques for Streaming Audio”, IEEE Network, vol.12, no.5, pp40-48, 1998.

[Ramsey] J.L.Ramsey, “Realization of optimum interleavers”, IEEE Trans. Info. Theory, vol. IT-16, pp 338-345, May, 1970.

[Rosenberg96] J.Rosenberg, “Reliablity enhancements to NeVoT”, Dec, 1996.

[Rosenberg98] J.Rosenberg and H.Schulzrinne, “An RTP payload format for generic forward error correction”, IETF Audio/Video Transport WG, work in progress(internet-draft), Jul., 1998.

[Sanneck] H.Sanneck, “A new technique for audio packet loss concealment”, IEEE Global Internet 1996, IEEE, pp.48-52, Dec. 1996.

[Scheirer] E. Scheirer, B. Vercoe, “SAOL: The MPEG-4 Structured Audio Orchestra Language”, *Computer Music Journal 23:2*, pp 31-51, 1999.

[Schroeder] M. R. Schroeder. “Computer Speech: Recognition, Compression, and Syn- theis”, Springer Verlag, Berlin, Germany, 1999.

[Shacham] N. Shacham, P. McKenney, “Packet recovery in high-speed networks using coding and buffer management”, *Proc. IEEE Infocom '90*, San Fransisco, CA, pp. 124-131, May 1990.

[Slaney] M. Slaney, “Auditory Toolbox”, Technical Report #1998-010, Interval Research Corporation, http://rvl4.ecn.purdue.edu/~malcolm/interval/1998-010/ .

[Tzanetakis]G. Tzanetakis, P. Cook, “Musical Genre Classification of Audio Signals”. *IEEE Transactions on Speech and Audio Processing*, 10(5), July 2002.

[Viswanathan] V.R. Viswanathan, “Variable frame rate transmittion: A review of methodology and application to narrow-band LPC speech coding”, IEEE Trans. Commun., vol. COM-30, No.4, pp.674-687, Apr., 1982.

[Wah] B.W. Wah, X. Su, D. Lin, “A Survey of Error Concealment Schemes for Real-time Audio and Video Transmissions Over the Internet”, IEEE International symposium on Multimedia Software Engineering, Taipei, pp.17-24, Dec. 2000.

[Wang2002] Y. Wang, S. Streich, “A Drumbeat-Pattern Based Error Concealment Method for Music Streaming Applications”, IEEE ICASSP2002, Orlando, Florida, USA, May 13-17, 2002.

[Wang2003] Y. Wang, J. Tang, A. Ahmaniemi, M. Vaalgamaa, “Parametric Vector Quantization for Coding Percussive Sound in Music”. IEEE ICASSP 2003, Hong Kong.

[Wang2003] Y. Wang, J. Tang, A. Ahmaniemi, M. Vaalgamaa. “Parametric vector quantization for coding percussive sounds in music”, ICASSP, 2003.

[Warren] R.M.Warren, “Auditory Percption”, Pergamon Press, 1982.

[Warren1988]W.H. Warren, R.R. Verbrugge, "Auditory Perception of Breaking and Bouncing Events: Psychophysics, ", *Natural Computation*, W. Richards, Ed., pp. 364--375. MIT Press, 1988.

[Wyse] L. Wyse, Y. Wang, X. Zhu, “Application of a Content-based Percussive Sound Synthesizer to Packet Loss Recovery in Music Streaming”. *Proceeding of the 11th ACM International Conference on Multimedia (Berkeley, CA*), 335-339,2003.

Appendix. Publications

- L. Wyse, Y. Wang, X. Zhu, “Application of a Content-based Percussive Sound Synthesizer to Packet Loss Recovery in Music Streaming”. *Proceeding of the 11th ACM International Conference on Multimedia (Berkeley, CA*), 335-339,2003.

- X. Zhu, L. Wyse, “Sound Texture Modeling Using TFLPC”. DAFx, 2004.