Application of a Content-Based Percussive Sound Synthesizer to Packet Loss Recovery in Music Streaming

Lonce Wyse
Institute for Incomm Research (I2R)
Heng Mui Keng Terrace
Singapore 119613
lonce@zwhome.org

Ye Wang
National University of Singapore (NUS)
3 Science Drive 2
Singapore 117543
wangye@comp.nus.edu.sg

Xinglei Zhu
I2R &NUS
Heng Mui Keng Terrace
Singapore 119613
xzhu@i2r.a-star.edu.sg

ABSTRACT
This paper presents a novel method to recover lost packets in music streaming using a synthesizer to generate percussive sounds. As an improvement of the state-of-the-art system that uses a content-based audio codebook, the new method can greatly reduce the redundant information needed to recover perceptually critical lost packets.

Categories and Subject Descriptors
H5.5 [**Information Interfaces and Presentation**]: Sound and Music Computing – *Signal analysis, synthesis, and processing*.

General Terms
Algorithms, Performance, Reliability, Human Factors.

Keywords
Music streaming, sound synthesis, packet error recovery.

1. INTRODUCTION
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. 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.

There has been a great deal of work on packet loss recovery, and 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 [1]. 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 [2] 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.

The failure of standard recovery techniques for this kind of signal led Wang *et. al.* [3] to a content-based method of error concealment. 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 QoS, though there are still certain limitations. First, the content-based codebook used for recovery, which is sent to receiver in a "header" segment prior to streaming the audio data, may be too large in application. In addition, 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.

In this paper, we present a novel method that uses a synthesizer to generate percussive sounds to reconstruct the lost packets. Instead of a PVQ codebook, which provides audio segments to recover lost packets, a codebook of parameters r is used to control the resynthesis of the percussive used for error recovery. The synthesized transients, together with a standard recovery method for non-transient sounds, reconstruct all lost packets. As we will see later, the method can maintain the perceptual audio quality improvements of other transient recovery methods, while greatly reducing the redundancy information needed to reconstruct perceptually critical music segments. Furthermore, the model-based approach also scales well with transmission and device capabilities. It could be used, for example, in real time regeneration without significant changes to the system design.

2. SYNTHESIS AND MODELING
For our purposes, a synthesizer is a library of general-purpose code with methods for generating and transforming sound algorithmically without necessarily using any recorded

MM'03, November 2-8, 2003, Berkeley, California, USA.

information. A sound model is an algorithm for generating a specific class of sounds (using a synthesizer) under the control of a sound-specific set of parameters.

2.1 The Nature of the Modeling Task

In this paper, our focus is on perceived quality of a musical stream, not on objective signal modeling or even perceptual signal matching. We focus on percussive sounds in a single frame, whose length is very short. Suppose the duration of the frame is 2048 PCM samples and the sampling frequency is 44.1KHz, the length of one frame is about 46ms. We need to generate sounds that are perceptually similar to these short percussive sounds when placed in the original musical context.

Using as few as four representative percussive audio vectors to reconstruct all lost transient packets in a piece of popular music, Wang et al. [3] found that listeners showed an overwhelming preference for their reconstructions over reconstructions based on packet repetition. This suggests that a very approximate sound model matching only a few critical features of the target sounds can be effective at significantly enhancing the perceived quality. The implication is that the sound models need not be indistinguishable from the target audio vectors derived using Wang's method to match their ability to improve perceived quality of service. This allows us to make our sound models simple with light computational demands and controlled by just a few parameters.

There is a long history of musical instrument, and in particular, percussion sound modeling (e.g. Karplus and Strong, [4]). Our modeling task is somewhat different since, while our target sounds are percussive, they seldom consist of a single instrument. The musical signal is generally made up of many simultaneously sounding instruments, some noisy, others pitched.

On the other hand, the target sounds are characterizable to such an extent that a general purpose audio coding scheme is not required – a single synthesis model "committed" to the characterizable class of sounds with only a small number of controllable dimensions is feasible. This is exactly what allows us to achieve the significant reduction in bandwidth required by this method.

2.2 Transmission Strategies

Suppose we need to transfer a piece of music containing percussion sound from server to client, there are several issues we need to consider:

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 more 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*. [3] 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.

Different considerations of the above issues generate different transfer strategies.

3. SYSTEM FRAMEWORK

The minimum assumption we need to make for our method to work is that there is a synthesizer on the client side. Given this synthesizer, the specific model can be very small and either exist on the client, or be sent on a per-song basis without being a significant factor in the bandwidth requirement.

There are five basic steps in the framework: detect percussive sounds; select codebook vectors; extract resynthesis parameters; synthesize percussion sounds; reconstruct the lost packet. Percussive detection, codebook selection and feature extraction are done on sever side; synthesis and reconstruction are done on the receiver device.

3.1 Transient Detection, Codebook Selection

The first two steps, detecting percussive sounds and selecting codebook vectors, are done exactly the same as in Wang *et al.* [3]. The process is illustrated in Figure 1. Percussive events are

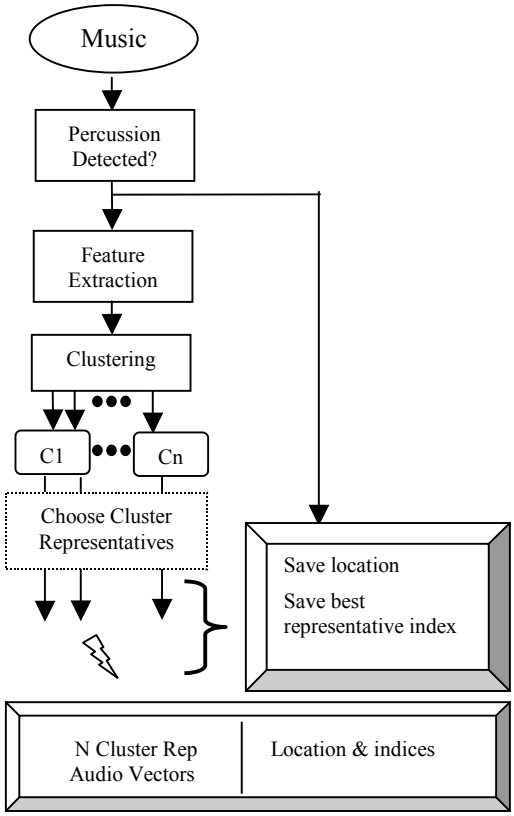

Figure 1. Percussion detection and codebook selection to create audio file header segment.

detected by looking for sudden increases in intensity across several subbands. 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 Wang *et al*. [3], 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.

3.2 Parametric Representation

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 2).

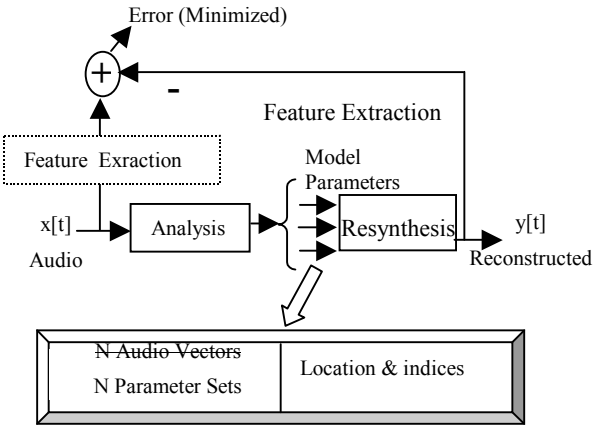

Figure 2. Analysis of audio to replace the codebook of audio vectors with a codebook of model synthesis parameters. This unit replaces the in Figure 1.

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 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.

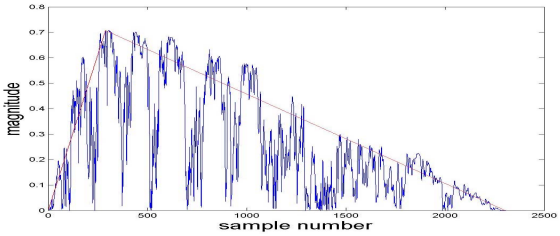

Figure 3. Event Contour

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) 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[5].

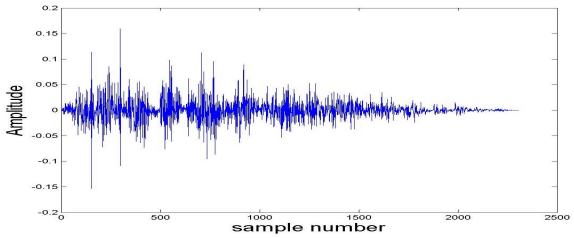

Figure 4. Residual of the LPC process

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, total energy of the percussive sound, pitch and pitch salience.

3.3 Transmission of Parameter Codebook

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

3.4 Resynthesis

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.

3.5 Lost Packet Recovery

With the reconstructed audio vector codebook, the packet loss recovery process can proceed exactly as in [3]. 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 5).

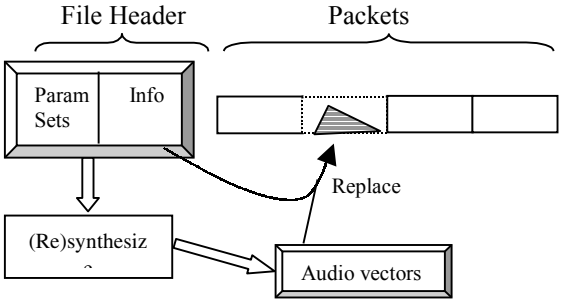

Figure 5. Reconstruction of a lost packet containing a percussive transient.

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

4. EVALUATION

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 [3], 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.

5. FUTURE WORKS

We have shown 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 QoS. Future work will focus on quality enhancements.

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.

One class of error in the current method occurs when 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, many more codebook entries would be necessary, or vector quantization could be done away with all together and a parameter set for each transient could be sent in the header. This would solve the pitch mismatches 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.

If the client computational resources were adequate, synthesis parameters for each transient segment could be sent in the packet previous to the transient and used to resynthesize a lost packet in real time. The header segment would need only consist of transient packet identifiers (no codebook data would be necessary at all). Some current phones, assuming computational equivalence to a 40 MHz PC, can already run our simple synthesis algorithms in approximately real time.

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.

6. REFERENCES

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

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

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

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

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