PARAMETRIC VECTOR QUANTIZATION FOR CODING PERCUSSIVE SOUNDS IN MUSIC

Ye Wang[1,], Jian Tang[1], Ali Ahmaniemi[1] and Markus Vaalgamaa[2]*

[1]Nokia Research Center, Tampere, Finland
[2]Nokia Mobile Phones, Helsinki, Finland
[*]School of Computing, National University of Singapore, Singapore

ABSTRACT

This paper presents a novel parametric vector quantization (PVQ) scheme as the secondary encoding to code perceptually salient percussive sounds such as drums in the time domain. It is deployed to improve the quality of service in the case of packet losses during percussive events. As a generalization and an improvement of our earlier system, the new scheme can achieve a better balance between bandwidth efficiency and error robustness. The proposed coding technique has been implemented with the MPEG-2 AAC (Advanced Audio Coding) frame structure. Experimental results with music samples have shown the effectiveness of the proposed scheme.

1. INTRODUCTION

Error concealment is usually a receiver-based error recovery method, which serves as an important part in mitigating the degradation of audio quality when data packets are lost in audio streaming over error prone channels such as mobile Internet. A fundamental limitation of all conventional methods is the assumption of short-term similarity of audio signals. This assumption is not always valid, especially in the case of transients in music.

To overcome the above-mentioned limitation, we have developed a drumbeat pattern based active error concealment method [1] for streaming music, which frequently has percussive objects such as drums to maintain the beat.

The main idea of our previous work was to recover musical beat structures in the case of packet losses, a concept analogous to pitch prediction (also referred to as long term prediction) in speech coding, since beat structures are essential to the perception of most music. It performed quite nicely when a music signal has a strongly metered beat structure.

However, further research revealed that our previous method still has some serious limitations. First, although it is fairly common that music, especially pop music exhibits a beat structure with strong regular accents, this assumption cannot be generalized. Our previous method fails if the drum pattern does not obey the above assumption or it changes abruptly, which is often the case in real-life music. Replacing a bass drum with a snare drum or other percussive sound is generally not acceptable in high quality audio. Second, musical beat information *per se* cannot guarantee the perceptual similarity of the two audio segments on the beats. Thus, the computations on the musical beat detection are largely irrelevant for the purpose of error concealment. The main concern in this type of error concealment should be drumbeats (the physical sound), not musical beats (a perceptual entity that can exist without any sound). Third, the time resolution of an AAC frame is not sufficient for detecting transients according to psychoacoustics [2]. This was one reason behind the failure of our early system to eliminate so-called *double-drumbeat effect* [1]. Fourth, in order to solve the *window shape mismatch* problem [3], we were forced to increase memory consumption in the decoder to save the same class of frequency data by a factor of four. To overcome these limitations, we propose a more general scheme for coding the percussive sounds, which may or may not correspond to the musical beats. This scheme can be considered as an extension to our earlier system [1].

The rationale to add secondary encoding for better protection of percussive sounds in music can be justified from two perspectives. From musicological viewpoint, percussions are often used to maintain the musical beats in a piece of music, which is one of the clearest features of the music to both musicians and non-musicians [4]. From an information theory viewpoint, a percussive signal produced by a beat instrument generally has greater perceptual entropy (PE) [5]. This phenomenon was vividly verified with the Huffman bits fluctuation in a piece of pop music [6].

2. SYSTEM DESCRIPTION

In this section, we focus on the three major steps in the proposed scheme: detection of percussive sounds, clustering of percussive sounds and generation of a codebook for percussive sound clusters.

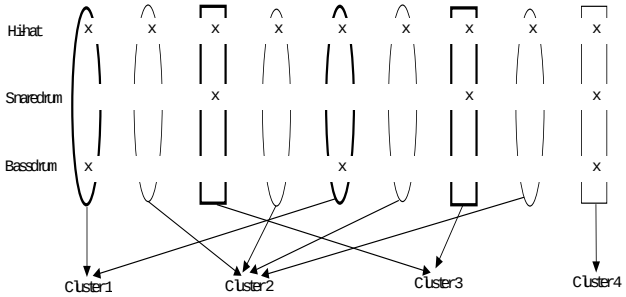

Figure 1. Percussion clustering conceptualization

The general conceptual framework of the proposed method is illustrated in Figure 1. The parametric vector quantization (PVQ) is implemented by employing an MPEG-2 AAC frame structure. In the encoder side, we first detect all perceptually salient percussive sounds in a piece of music off-line, and then employ the PVQ to cluster them into a few classes, such as bass drum,

snare drum, hi-hat, or their combinations based on their perceptual similarity. The parameters of the secondary encoding (i.e., VQ indices) are subsequently embedded into the AAC bitstream as ancillary data, similar to the system in [1]. The flowchart is illustrated in Figure 2.

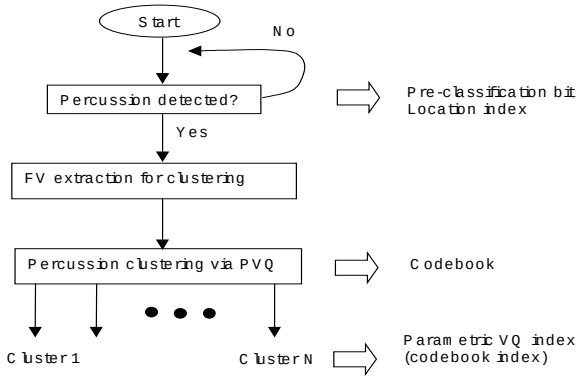

Figure 2. Flowchart of percussion detection and clustering

The codebook (the representatives of all the clusters) is transmitted in advance to fill the receiver's percussion buffers before the actual streaming begins. The PVQ bitstream (codebook and codebook index) is used to reconstruct the percussive sound, and the neighboring frames are used to reconstruct the stationary part of the lost frame, as illustrated in Figure 3.

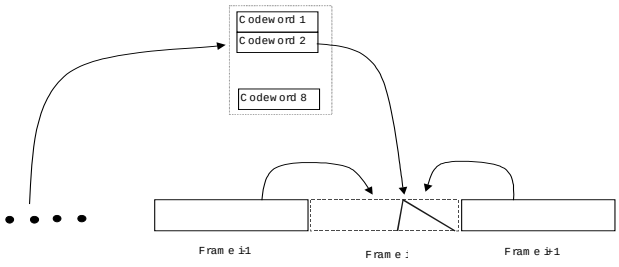

Figure 3. Reconstruction of percussive sound in the decoder. The dashed rectangle represents the missing AAC frame. The triangle represents the percussion from the codebook.

2.1. Percussive sound detection

We use a subband approach, similar to [3], to perform onset detection with a time resolution defined by the short window (256 PCM samples) within an AAC frame (2048 PCM samples) [7]. In general, a percept of an onset is caused by a noticeable change in intensity, pitch and timbre of the sound [8]. Our onset detector is based on the subband intensity alone, since a perceptually salient percussive event is usually accompanied by an intensity surge at least in one subband.

The onset detection part is illustrated in Figure 4. Although the general structure of our onset detector is similar to that in [9], there are some fundamental differences in them.

The subband energy slope (first order difference function) is calculated first as the preliminary feature, followed by a halfwave rectifier. To prevent excessive fluctuation of the preliminary feature due to the increased time resolution, a smoothing function is introduced by simply summing previous feature values over a fixed time window, which is similar to the temporal energy integration of the human auditory system. Unlike a fixed threshold proposed in [9], we apply a novel adaptive threshold in all subbands to select possible candidates. In each subband we pick up only the maximum of all local maxima above the threshold within a pre-defined time window.

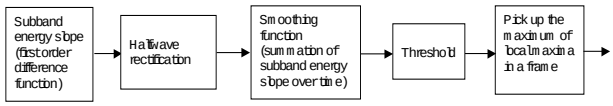

Figure 4. Block diagram of subband processing for onset detection

Another fundamental difference between our onset detector and that of [9] is that we simply use the smoothed subband energy slope instead of the so-called *first order relative differential function* proposed in [9]. The reason for us to dismiss the logarithm operation in the calculation of the *first order relative differential function* is its poor performance in reliable onset detection.

In order to detect an onset component, we need a feature, which can separate an onset and others as much as possible. The smoothed first order difference function (feature) combined with the proposed adaptive threshold can detect onsets reliably. However, if a logarithm operation is applied to the feature, its dynamic range will be compressed, thus making the onset detection much more difficult.

Our new adaptive threshold in each subband is calculated based on the smoothed first order difference function (feature):

$$F_{thr} = K \cdot m + C ,$$

where K is derived from the variance of the feature and K is set as a constant in our current implementation, m is the local mean of the feature over a duration of 301 short windows (ca. 900 ms) excluding the middle 5 short windows, C is a constant, which is based on large training data statistics. C indicates the minimum detectable changes in each subband.

It is very common that the onset positions detected from different subbands are not consistent. The final position of the onset is a weighted mean of onset candidates from different subbands.

We have introduced a confidence score to indicate how pure (without mixing with other sounds such as singing-voice) is the detected percussion.

$$R_s = \frac{F_s - F_{thr}}{F_s} ,$$

where R_s is the confidence score of the percussion in an individual subband s, F_s is the feature value of the percussion in the subband.

$$R_i = \frac{1}{N}\sum_{s=1}^{N} R_s \cdot w_s ,$$

where R_i is the overall confidence score of the percussion in the frame *i*, *N* is the number of subbands. w_s is the weighting factor and $\sum_{s=1}^{N} w_s = 1$. The selection principle of w_s is the same as that in [3]. That is, we put more weight to the low and high frequency range, where the main energy of the percussion is presented.

2.2. Clustering of percussive sounds

After the pre-processing, all percussive sounds are detected and their positions indexed. For the purpose of percussion clustering, it can be advantageous to employ a new set of FV based on short window spectral data with uniform window shape, either sine or Kaiser-Bessel derived (KBD) window as defined in MPEG AAC standard. The frequency resolution of our method is then limited by the short window length of AAC.

Feature Vector (FV)

The duration of percussive sounds is estimated using the energy contour of the detected percussions.

The input to the clustering process is a set of multi-dimensional feature vectors (FVs). For a compact description of percussions, a 12-dimensional FV is used in our current implementation, which is different from [10]. The 12-dimensional FV includes 3 fullband features (*i.e.* the fullband peak energy, confidence score calculated in the onset detection module, bandwidth calculated the same way as in [11]) and 3x3 subband features. For the feature extraction, we employed 3 subbands that are in the frequency ranges of 0-172 Hz, 172-344 Hz and 11025 - 22050 Hz, respectively. Two features are dedicated to the low subband energy and the third feature is dedicated to the high subband energy. This is to describe the frequency domain characteristics of the percussion.

The fullband peak energy and confidence score roughly describe the onset characteristics of a percussive event, and the bandwidth describes the bandwidth characteristics of the event. The subband features describe a signal of 15 short windows in duration starting from the onset. The 15-short-windows signal is converted into 3 sets of subband features; each set represents 5 consecutive short windows. This is used to describe the decay characteristics of the percussion.

This 12-dimentional vector of features worked quite well with our test signals. However, it is possible to further optimize the features. Possible improvements include introducing a weighting factor for each feature according to its perceptual significance, including more features such as spectral flatness.

LBG-VQ Algorithm

This module is used to train the preliminary codebook. A well-known method of obtaining the VQ codebook is the generalized Lloyd algorithm (GLA) [12].

The GLA algorithm is sometimes referred to as the Linde-Buzo-Gray (LBG) algorithm. Given a codebook, the VQ encoding process is the minimum-distortion quantization of an input vector, under a certain distance measure. This means finding the nearest neighborhood in the codebook, which requires vector distance computations using the exhaustive search of the codebook.

Intuitively, the VQ codebook size should be the number of musical instruments and their combinations used in a piece of music. It is rather difficult to determine the right size of codebook since the number of percussive instruments in a piece of music is unknown. However, the intention of PVQ is not to separate percussions produced by different music instruments but to cluster them into a number of artificial classes based on perceptual similarity measurement. Using an EM (Expectation maximization)-Based algorithm, the total average-distortion of VQ will decrease monotonically when the size of codebook increases. That is, the larger the codebook size, the more accurate representation the codebook for the FV space.

However, using an unnecessarily large codebook requires time-consuming codebook training and more bits for coding the codebook and codeword index. Our experiments have shown that 8 clusters are sufficient for most of our test music signals. Even 4 clusters can be satisfactory for a majority of music samples. The percussive events within each class are perceptually similar.

2.3. Percussive Codebook selection

The proposed PVQ can be considered as a particular implementation of the concept proposed in [13] and an improved version of the scheme proposed in [6].

Since the percussive codebook contains the representations of all clusters, it has to be chosen carefully. Our codebook is not constructed simply based on the centroid of each class, but based on the following criterion:

$$k = \arg\max_{D_k \le D_{thr}} (R_k),$$

where D_{thr} is the threshold distance for each class. A member with confidence R_k in class *k,* whose distance (D_k) to its centroid is beyond D_{thr}, cannot be selected for the codebook. The member within D_{thr} that has the maximum confidence score, is chosen for the codebook to represent class *k.*

The rationale for the above criterion is that members that are too far from the centroid should not be included in the codebook, and those heavily contaminated with other sustaining sounds such as singing-voice (*i.e.* the confidence is low) should also be excluded from the percussive codebook.

2.4. Discussion

In the networked world, users will soon be able to search through vast databases at the song level [14]. Based on this assumption, the pre-processing and PVQ of our system is also performed at an individual song level.

There are two major reasons for us to use the actual data for training of the codebook of the PVQ.

- It is desirable to eliminate the mismatch between training data and actual data to yield a very compact codebook. In

the proposed method, the overhead information for the percussive sounds is extremely small, e.g. only a few bits per AAC frame.

- There are many different percussive instruments for different types of music. From a VQ point of view, the vector space is a fairly large set. However, the percussive sounds in one individual song will occupy just a very small subset of the large set. If a large set is desirable, the corresponding codebook has to be either pre-stored in the receiver or transmitted before the streaming of music. For terminals with strict memory constraints, this could pose a problem.

3. EVALUATION OF PERFORMANCE

To evaluate the performance of the proposed scheme, we have employed an unsymmetrical window function, which approximates the drum contour, to isolate all detected percussions from the music signal and muted all remaining sounds. The duration of our percussive window is 2048 PCM samples (ca. 15 short windows), which is ca. 46 ms if the sampling frequency is 44.1 kHz. The percussive window function is preliminary and can be improved in future. Our percussive codebook is selected from the isolated percussive events. All the isolated percussions are labeled with their cluster indices so that we can playback one particular cluster or all clusters at a time for human judgment.

Six pop music samples with a duration of around 30 seconds from commercial CDs were used in our tests. The number of clusters was four for all test samples in this test evaluation. The 3rd author, who is also a musician, performed the annotation. One cluster of the detected percussions was listened at a time. The ones, which were clearly different from others in that cluster, were counted as incorrect. The process continued until all four clusters were gone through. The annotated results are summarized in Table 1.

Music excerpt	Percussions in total	Incorrect	Correct (%)
Toto	55	6	89
Abba	43	3	93
Aerosmith	72	19	71
Roxette	56	8	86
Sabrina	76	20	74
Michael Jackson	78	4	95

Table 1. Summary of percussion clustering results judged by a human subject compared to that by the proposed system

The results show that the proposed system works well with typical music material. The small percentage of error in the evaluation is due to the following reasons:

- If there are more different kinds of drums used in a song, it becomes a reality that two different drums are clustered into the same cluster, thus producing incorrect results. For some songs four clusters are not sufficient.
- Drumbeats mixed with other sustaining sounds such as singing voice pose a problem in percussion clustering. There are still rooms for improvement in this aspect.
- The percussive window function has changed the time-frequency characteristics of the percussion somewhat.

4. CONCLUSION

An efficient audio coding technique is proposed for coding percussive sounds in music with promising results. The major advantages of the proposed scheme are the negligible overhead information, lower system latency than the network-based re-transmission and efficient memory management. The next step is to refine the parameters and to integrate the proposed scheme into our audio streaming system for packet loss recovery.

5. ACKNOWLEDGEMENT

Ye Wang wishes to thank Dr. Jilei Tian, Dr. Simon Dixon and Mr. David Isherwood for helpful discussions and suggestions.

6. REFERENCES

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

[2] Moore, B.C.J, "An Introduction to the Psychology of Hearing," Academic Press, 1997

[3] Wang, Y., Vilermo, M., "A Compressed Domain Beat Detector Using MP3 Audio Bitstreams," The 9th ACM International Conference on Multimedia, Ottawa, Canada, pp. 194-202, September 30-October 5, 2001

[4] Martin, K.D., Scheirer, E.D., Vercoe, B.L., "Music Content Analysis through Models of Audition," ACM Multimedia'98 Workshop on Content Processing of Music for Multimedia Applications, Bristol, UK, 1998

[5] Johnston, J.D., "Estimation of Perceptual Entropy Using Noise Masking Criteria," IEEE International Conference on Acoustics, Speech and Signal Processing, 1988

[6] Wang, Y., Ojanpera, J., Vilermo, M., Vaananen, M., "Schemes for Re-compressing MP3 Audio Bitstreams", Audio Engineering Society (AES) 111th Convention, Nov. 30 – Dec. 3, 2001, New York, USA

[7] Wang, Y., Vilermo, M., "The Modified Discrete Cosine Transform: its Implications for Audio Coding and Error Concealment," to appear in the January issue of the Journal of Audio Engineering Society, 2003

[8] Moelants, D., Rampazzo, C., "A Computer System for the Automatic Detection of Perceptual Onsets in a Musical Signal," In Camurri, Antonio (Ed.) " KANSEI, The Technology of Emotion," pp. 140-146, Genova, 1997

[9] Klapuri, A., "Sound Onset Detection by Applying Psychoacoustic Knowledge," ICASSP2000, Istanbul, Turkey, June 5-9, 2000

[10] Wold, E., Blum, T., Keislar, D., Wheaton, J., "Content-Based Classification, Search, and Retrieval of Audio," IEEE Multimedia Vol.3, No. 3, pp.27-36, Fall 1996

[11] Peltonen, V., Tuomi, J., Klapuri, A., Huopaniemi, J., Sorsa, T., "Computational Auditory Scene Recognition," ICASSP2002, Orlando, Florida, USA, May 13-17, 2002

[12] Huang, X. D., Acerd, A., Hon, H.W., "Spoken Language Processing, *A Guide to Theory, Algorithm and System Development*," Upper Saddle River, NJ, Prentice Hall, 2001

[13] Scheirer, E. D., "Structured Audio, Kolmogorov Complexity, and Generalized Audio Coding," IEEE Transactions on Speech and Audio Processing, Vol.9, No. 8, November 2001

[14] Logan, B., Salomon, A., "A Music Similarity Function based on Signal Analysis," IEEE International Conference on Multimedia and Expo, Tokyo, Japan, 2001