A COMPRESSED DOMAIN BEAT DETECTOR USING MP3 AUDIO BITSTREAMS

Ye Wang, Miikka Vilermo
Speech and Audio Systems Laboratory
Nokia Research Center
P.O.Box 100
FIN-33721 Tampere, Finland
{ye.wang, miikka.vilermo}@nokia.com

ABSTRACT
This paper presents a novel beat detector that processes MPEG-1 Layer III (known as MP3) encoded audio bitstreams directly in the compressed domain. Most previous beat detection or tracking systems dealing with MIDI or PCM signals are not directly applicable to compressed audio bitstreams, such as MP3 bitstreams. We have developed the beat detector as a part of a beat-pattern based error concealment scheme for streaming music over error prone channels. Special effort was used to obtain a tailored trade-off between performance, complexity and memory consumption for this specific application. A comparison between the machine-detected results to the human annotation has shown that the proposed method correctly tracked beats in 4 out of 6 popular music test signals. The results were analyzed.

Keywords
Error concealment, Beat detection, Beat tracking, Compressed domain processing, Bitstream processing, MP3, MPEG audio.

1 INTRODUCTION

With rapid deployment of audio compression technologies, more and more audio content is stored and transmitted in compressed formats. The transmission of audio signals in compressed digital packet formats, such as MP3, has revolutionized the process of music distribution. Consequently, compressed bitstream processing is becoming a subject of study [1][2][3][4]. However, compressed domain bitstream processing is still in its infancy and many aspects such as beat detection remain unaddressed.

ACM Multimedia 2001 Sept. 30 - Oct. 5, 2001, Ottawa, Ontario, Canada

Beat detection or tracking is an important initial step in computer processing of music and is useful in various multimedia applications, such as automatic classification of music, content-based retrieval, audio track analysis in video, etc.

Beat detection or tracking systems can be classified according to the input data type. Most existing beat-tracking systems deal with musical score information (typically MIDI signals) [9][10][11], or PCM samples [12][13][14][15][16][17]. Some are designed for real-time applications.

None of the above-mentioned systems is directly applicable to a compressed domain bitstream such as MP3 bitstream, which has gained popularity not only in the Internet world, but also in consumer products. In addition, existing algorithms usually have such a high computational complexity that it is beyond the capability of a normal laptop computer (not to mention handheld devices) to perform a real-time application task – beat-pattern based error concealment for streaming music over error prone channels having burst packet losses [5]. Our objective here was not to develop a general-purpose beat detector, but to develop a beat tracking method as a building block of the error concealment system proposed in [5] with strict constraints on complexity and memory requirement. The proposed beat detector serves to segment music signals according to beats. The ultimate goal is to define a segment-similarity measure that relates closely to the subjective similarity, which will enable us to perform beat-pattern based error concealment and coding tasks better [5][7].

This paper is organized as follows. The concept of beat-pattern based error concealment is first outlined in section 2. It serves to clarify the necessity and requirements of such a beat tracking method. A window-type based beat detector is presented separately in section 3 due to its importance on error concealment. A Modified Discrete Cosine Transform (MDCT) domain beat detector is then detailed in section 4. Some preliminary evaluations of the new scheme are presented in section 5. Finally, section 6 concludes the

paper with some discussions and indicates some future work.

2 CONCEPT OF BEAT-PATTERN BASED ERROR CONCEALMENT

Error concealment usually serves as the last resort to mitigate the degradation of the audio quality when compressed audio packets are lost in error prone channels, such as mobile Internet and digital audio broadcasts.

Conventional error concealment methods include muting, interpolation or simply repeating a short segment immediately preceding the lost segment. They are useful if the lost segment is short (an usual assumption in the literature is around 20 ms) and the signal is fairly stationary. However, if these conditions do not hold, conventional methods will not produce satisfactory results.

To solve this problem, a new scheme was proposed to exploit the beat-pattern similarity of music signals to recover a possible burst packet loss in a best-effort based network such as the Internet. [5].

The beat-pattern based error concealment scheme results from the observations that a music signal typically exhibits rhythm and beat characteristics. And the beat-patterns of most music, particularly pop, march and dance music are fairly stable and repetitive.

The time signature of pop music is typically 4/4. The average inter-beat interval (IBI) is about 500 ms, thus the duration of a bar is about 2 s. Such long-term similarity of music has not been exploited in any existing audio coding technology. The concept is quite simple and straightforward. If the lost or distorted segment of the audio signal includes a beat, it would be better to replace it with a segment from a previous beat.

A conventional error concealment method and our new approach are illustrated in Figure 1 and 2 respectively. The small segments represent MP3 granules. An MP3 frame consists of two granules where each granule consists of 576 frequency components.

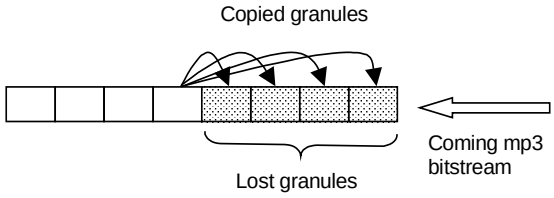

Figure 1. Illustration of a conventional error concealment method. Rectangles filled with dots represent corrupted MP3 granules. Blank rectangles represent error-free ones. The thin arrows indicate the repetitive copy of the immediately preceding granule to fill the erroneous audio segment.

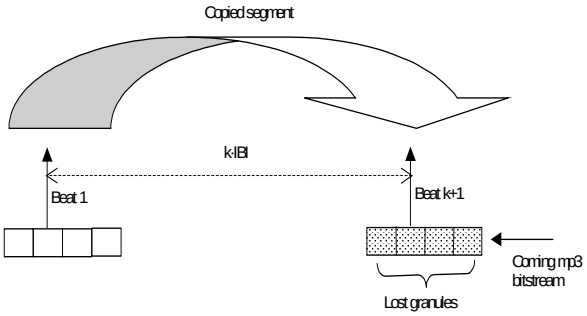

Figure 2. Concept of the beat-pattern based error concealment method. It replaces an erroneous audio segment around beat (k+1) with a corresponding segment from a previous beat as indicated by the thick arrow. k is a positive integer which is determined by the employed level of beat information (e.g. quarter-note or half-note level). IBI stands for inter-beat interval. Rectangles filled with dots indicate corrupted MP3 granules. Blank rectangles indicate error-free ones.

The assumption for this approach is that a segment around a beat, which often corresponds to a transient produced by a rhythmic instrument such as a drum, is subjectively more similar to a segment around a previous beat than its immediate neighboring segment. A possible psychological verification of this assumption is explained by the following example. If we observe typical pop music with a drum sound marking the beat in a 3-D time-frequency representation (see Figure 6), the drum sound usually appears as a ridge, short in the time domain and broad in the frequency domain, which masks all other sounds such as singing and other instruments well. It is usually so dominant in pop music that one perceives only the drum sound during the event. In spite of some variations in consecutive drum sounds, it is logical to propose that it would be subjectively more pleasant to replace a missing drum sound with a previous drum sound segment rather than with any other sound, such as singing. It becomes evident from this that a beat detector is a crucial element of the scheme. And it is reasonable to perform the beat detection directly in the compressed domains to avoid redundant operations.

The requirement of such a beat detector depends on the constraint on computational complexity and memory consumption. In our current implementation, the beat detector employs only the window types and the MDCT coefficients decoded from the MP3 bitstream to perform beat tracking. It outputs 3 parameters: beat position, IBI and confidence score. However, if the constraint on complexity and memory were relaxed, higher level structure (e.g. bar-level structure) would improve error

concealment performance.

3 WINDOW TYPE BASED BEAT DETECTION

MP3 uses 4 different window types: long, long-to-short, short and short-to-long which are indexed with 0, 1, 2, 3 respectively (see Figure 3(b)). The short window is introduced to tackle transient signals better. From our experiments with pop music, short windows often coincide with beats and offbeats since they are the most frequent events to trigger window-switching. We have observed that 99% of the window-switching patterns in all of our test signals appear in the following order: long => long-to-short => short => short => short-to-long => long. This pattern can be indexed as a sequence of 012230 (see Figure 3(b)).

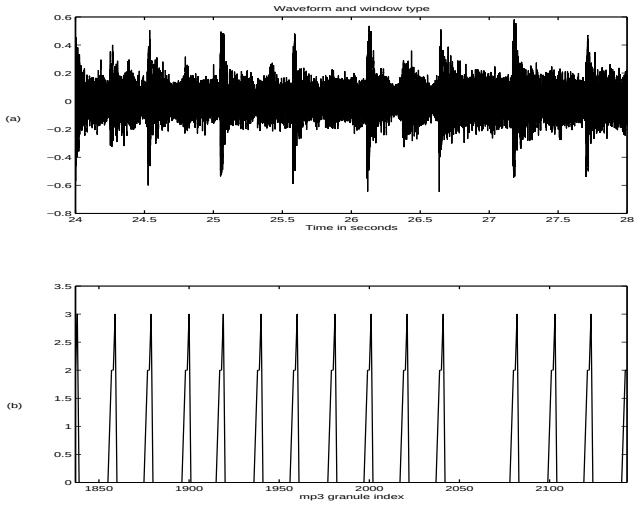

Figure 3. Comparison of music waveform and its corresponding mp3 window-switching pattern. (a) music waveform versus time in seconds, (b) window types (vertical axis) versus mp3 granule index (horizontal axis). The four window types (long, long-to-short, short and short-to-long) are indexed with 0, 1, 2, 3 respectively.

It should be noted that the window-switching pattern depends not only on the encoder implementation, but also on the applied bitrate. Therefore, window-switching alone is not a reliable cue for beat detection. For general purpose beat detection, we could even completely discard the window type information. A MDCT based method alone would be sufficient.

However, for error concealment purposes window type information plays an important role. Therefore, we take the following strategy to handle the beat information from the two separate sources. The MDCT based method serves as the baseline beat detector due to its reliability. Then the beat information (position and IBI) is checked with the window-switching pattern. If the window-switching also indicates a beat and its position departs from the MDCT based one less than 4 MP3 granules (ca. 13x4 = 52 ms), we take the beat information from the window-switching and adjust the beat information accordingly. That is, the window-switching method always has priority. The beat information from MDCT based method is used only in the absence of window-switching (see Figure 3, 5 and 6).

The rational of this strategy is that the window shapes in all MDCT based audio codecs including MPEG-2/4 advance audio coding (AAC) must satisfy certain conditions to achieve time domain alias cancellation (TDAC) [6]. If these conditions are violated due to the error concealment operation, the time domain alias will not be able to cancel each other during the overlap-add (OA) operation [6]. This will result in clearly audible distortion as a consequence.

For example, if the two consecutive short window granules indexed as 22 in a window-switching sequence of 012230 are lost in a transmission channel, it is easy to deduce their window types from their neighboring granules. And a previous short window granule pair should replace them to mitigate the subjective degradation. However, if we disregard the window-switching information available from the audio bitstream and replace the short window with any other neighboring window types, resulting in window-switching patterns such as 011130, the TDAC conditions will be violated. This will create annoying artifacts. We define the phenomenon as *window type mismatch phenomenon*.

To our best knowledge, this important issue has not been addressed in any publications to date.

Let's consider the same example of a MP3 granule sequence of 012230 as discussed above. In case a segment of four consecutive granules indexed as 1223 is *partially* corrupted in a communication channel, it is still possible to detect the transient, if we can correctly decode only the window type information (2 bits) of one *single* granule in the segment of four consecutive granules, even if their main data is totally corrupted.

The above analysis clearly suggests why *partially* damaged audio packets due to channel error should not be simply discarded because they can still be utilized to improve quality of service (QoS) in applications such as streaming music. This clarifies the significance of the window type information and the rational of our strategy to combine beat information from the two separate detection methods.

4 MDCT DOMAIN BEAT DETECTION

The MDCT coefficients based method has the following building blocks (see Figure 4 and 5):

- Feature Vector (FV) calculation: calculates the multi-band energy within each granule (ca. 13 ms) as a feature, and then forms a FV of each band within a search window. FV serves to separate beats and non-beats as much as possible. An element to mean ratio (EMR) can be used to improve the feature quality.

- Beat candidate selection: This process is performed in two stages. Beat candidates are first selected in individual bands based on a threshold method in a given search window. Within each search window the number of candidates in each band is either one or zero. If there are one or more valid candidates selected from individual bands, they are then clustered and converged to a single candidate according to certain criteria.
- Confidence score: A confidence score is calculated for each beat candidate from individual bands to score their reliability. Based on them, a final confidence score is calculated, which is used to determine whether a converged candidate is a beat.
- Statistical model: An inter-onset interval (IOI) histogram is usually employed to select the correct inter-beat interval (IBI) [13]. The idea is to use the IBI derived from the IOI histogram to predict the next beat. In our system a valid candidate in each individual band is defined as an onset. A set of previous IOIs in each band is stored in a FIFO for computing the candidate's confidence score of that band. Instead of a usual histogram approach, our statistical model employs a median in the FIFO buffer to predict the position of the next beat, which works quite well.
- Mark and output beat information: Before a beat candidate is finally marked and stored as a beat, it has to pass a confidence test. Only a candidate with sufficient confidence is selected as a beat (see Figure 9). Its position, IBI and confidence score are stored and also fed back to calculate the confidence score of future beat candidates. This beat information then is checked with the window-switching information, adjusted accordingly.

A high-level block diagram of the MDCT domain beat detector is illustrated in Figure 4. More detailed information about each block is given in the subsequent sub-sections.

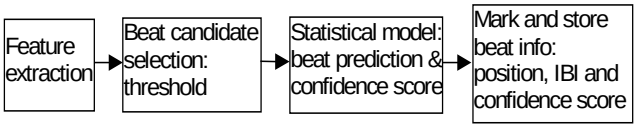

Figure 4. Block diagram of a MDCT based beat detector

Feature Extraction

We use subband energy or EMR of the subband energy in a search window as a feature vector (FV). The FV is directly calculated from decoded MDCT coefficients as illustrated in Figure 5. We chose an approach, which extracts FV from the full-band and individual subbands separately to avoid possible loss of information. The frequency boundaries of the new subbands are defined in table 1 and 2 for long and short windows respectively for a sampling frequency of 44.1 kHz. For other sampling frequencies the subbands can be defined in a similar manner.

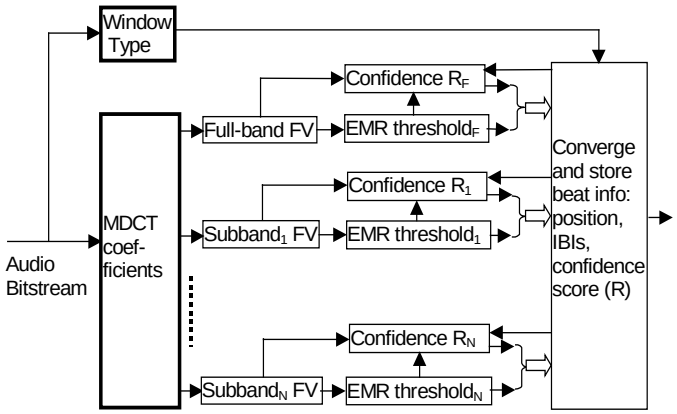

Figure 5. Block diagram of a compressed domain beat detector using MP3 bitstream. FV stands for feature vector.

Sub-band	Frequency interval (Hz)	Index of MDCT coefficients	Scale factor band index
1	0-459	0-11	0-2
2	460-918	12-23	3-5
3	919-1337	24-35	6-7
4	1338-3404	36-89	8-12
5	3405-7462	90-195	13-16
6	7463-22050	196-575	17-21

Table 1. Subband division for long windows

Sub-band	Frequency interval (Hz)	Index of MDCT coefficients	Scale factor band index
1	0-459	0-3	0
2	460-918	4-7	1
3	919-1337	8-11	2
4	1338-3404	12-29	3-5
5	3405-7465	30-65	6-8
6	7463-22050	66-191	9-12

Table 2. Subband division for short windows

MP3 employs a hybrid filterbank. In principle, the feature extraction can also be performed after an Inverse Modified Discrete Cosine Transform (IMDCT) step [8]. We chose the decoded MDCT coefficients for feature extraction, in order to make the algorithm more general and applicable to

other codecs such as MPEG2/4 AAC, which uses only a MDCT.

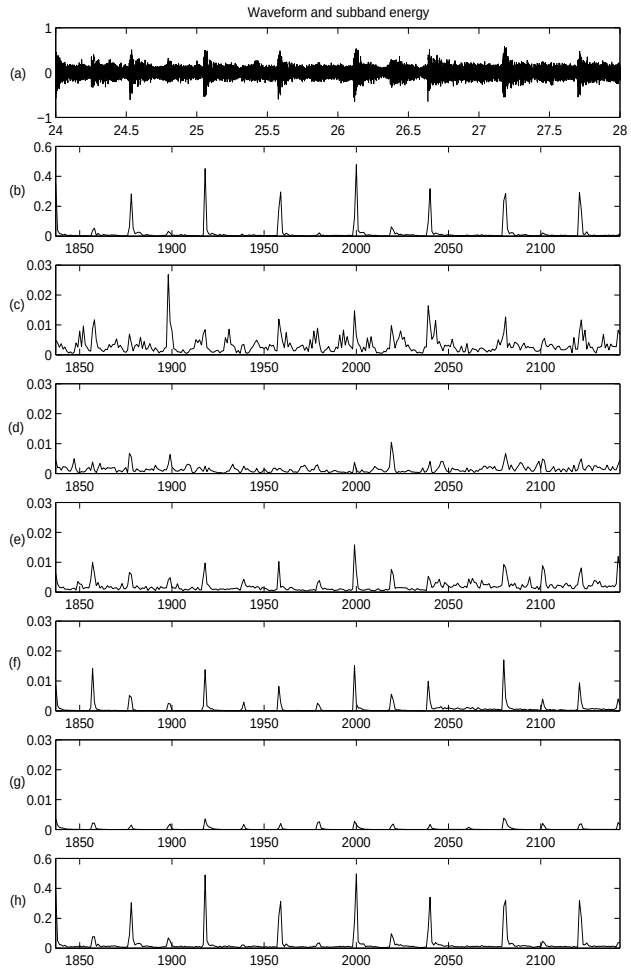

Figure 6. Music waveform of a 4 seconds segment and its corresponding subband energy employing the same pop music sample as in Figure 3. (a) music waveform versus time in seconds, (b)-(h) energy in subbands 1-6 and full-band versus mp3 granule index.

MP3 has an option to use long or short windows. The window length is 36 subband samples in the case of long windows and 12 subband samples in the case of short window. 50% window overlap is used in the MDCT. In order to have a consistent frequency resolution for both long and short windows we grouped the MDCT coefficients of each granule into 6 newly defined subbands (see tables 1 and 2) for feature extraction. For other codecs or configurations, similar frequency divisions can be performed. This frequency division is different in comparison to most previous beat detectors due to the constraint of the MPEG standard and system complexity.

In Figure 5, each band gives only one value by summation of the energy within a granule [8]. Thus the time resolution of our beat detector is one MP3 granule (ca. 13 ms) as opposed to a theoretical beat event, which has no duration.

The energy $E_b(n)$ of band b in granule n is calculated directly by summing the squares of the decoded MDCT coefficients to give:

$$E_b(n) = \sum_{j=N1}^{N2} [X_j(n)]^2 \qquad (1)$$

where $X_j(n)$ is the j[th] normalized MDCT coefficient decoded at granule n, $N1$ is the lower bound index and $N2$ is the higher bound index of MDCT coefficients defined in Table 1 and 2. Since the feature extraction is performed in granule level, the energy in three short windows (equal to one long window in duration) is combined into one so that we have comparable energy for both long and short windows.

Based on the observations of the extracted features from different pop music, we have concluded that subbands 1, 5, 6 and the full-band features are generally reliable for pop music beat tracking. The features extracted from a pop music extract are illustrated in Figure 6.

For simplicity our current system only uses these 4 bands to extract the feature vector. The reason that subbands 2, 3 and 4 usually give rather poor features is that singing and instruments other than drums are mostly concentrated in these bands. Consequently, the beat and non-beat separation is usually rather difficult in these bands. As illustrated in Figure 6, feature vectors are extracted in multiple bands and then processed separately.

Search window

The search window size determines the FV size, which is used for selecting beat candidates in individual bands. The search window size can be fixed or adaptive. Based on our experiments both methods are feasible. In the case of the fixed window size, the minimal possible IBI (~325 ms) is chosen as the search window size so that the maximal number of possible beats within the search window is one. The current system uses an adaptive window size because of its slightly better performance. It is calculated as the closest odd integer to the median of the stored IOIs, so that we have a symmetric window around a valid sample:

$$window\_size\_new = 2\lfloor median(\overline{IOI})/2 \rfloor + 1 \qquad (2)$$

The hop size is selected to be half of the new search window size.

$$hop\_size\_new = round(window\_size\_new/2) \qquad (3)$$

Beat candidate selection

The basic principle of beat candidate selection is setting a proper threshold for the extracted FV. The local maxima within a search window, which fulfils certain conditions, are selected to be beat candidates. This process is performed in each band separately. There are two threshold-based approaches for selecting beat candidates. The first approach uses the primitive FV (multi-band energy) directly and the second approach uses an improved FV (EMR).

The first method is based on the absolute value of the multi-band energy of beats and non-beats. A threshold is set based on the distribution of beat and non-beat for selecting beat candidates within the search window. This approach is computationally simple but needs some knowledge of the feature in order to set a proper threshold. It has three possible outputs in the search window: no candidate, one candidate or multiple candidates. In the case of one or multiple candidates, it is desirable to have a subsequent statistical model to determine the reliability of each candidate as a beat. The beat detector in [5] was based on this method.

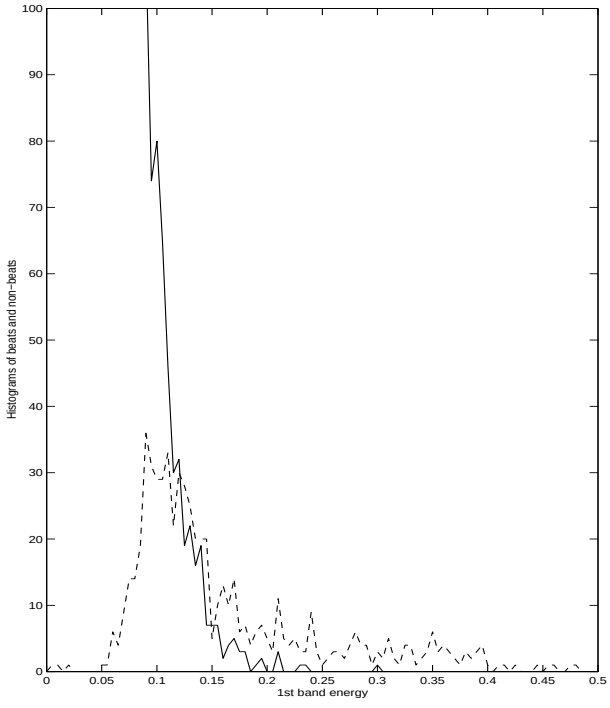

Figure 7. Histogram of beats (dashed line) and non-beats (solid line) versus their first-band energy (feature vector extracted with the first method) employing a pop music sample (6 minutes in duration).

The second method uses the primitive FV to calculate an EMR within the search window to form a new FV. That is, we calculate the ratio of each element (energy in each granule) to the mean value (average energy in the search window). And then the maximum EMR is compared with a given threshold. If the EMR is greater than the threshold, this local maximum is selected as a beat candidate. The beat candidate is sent to the next stage for further processing as illustrated in Figure 5.

The second approach seems to be superior to the first approach in most cases since it measures the relative distance between the individual element and the mean, not their absolute values. Therefore, the EMR threshold can be set as a constant value, while the threshold in the first method should be adaptive to cope with the wide dynamic range in music signals. EMR is used in our current implementation. Comparison of the two methods with an identical sample (6 minutes in duration) is shown in Figures 7 and 8. The beats were picked up manually by a human subject. Although the EMR method has slightly better separation between beats and non-beats, none of the two FVs is good enough to separate beats and non-beats reliably without a subsequent statistical model. The wide signal dynamic range and relatively strong offbeats mainly cause the bad separation.

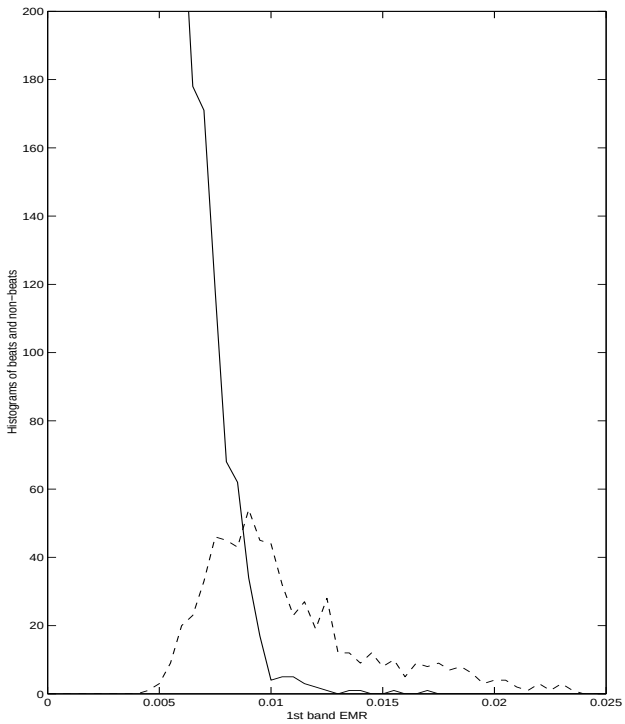

Figure 8. Histogram of beats (dashed line) and non-beats

(solid line) versus their EMR measure (feature vector extracted with the second method) employing the same pop music sample as in Figure 7.

In order not to miss a possible beat, we were forced to set a threshold towards the lower end of the beat population, which is about 0.005 in Figure 8. Thus the probability of selecting non-beats as beat candidates is rather high. Subsequent statistical models will eventually remove false selections.

Statistical model

We define a valid candidate in each band as an onset and store a number of previous IOI values in a FIFO buffer for beat prediction in each band. Then we use the median of the IOI vector to calculate the confidence scores of all beat candidates in individual bands. This simple statistical model has proven to be quite effective.

The IOI vector size is a tunable parameter for adjusting the responsiveness of the beat detector. If the IOI vector size is kept small, the beat detector is quick to adapt to a changed tempo at the cost of instability. If the IOI vector size is large, it becomes slow to adapt to a changed tempo, but it can tackle more difficult situations better. In the current implementation, the FIFO buffer size is 9. Since we store the IOI as opposed to the final IBI in the buffer, the tempo change is registered in the FIFO. However, the search window size is only updated to follow the new tempo after 4 IBIs, which is about 2~3 seconds in duration.

Confidence score

The confidence score for an individual beat candidate is calculated to measure its reliability:

$$R_i = \max_{k=1,2,3} \left\{ \frac{median\left(\overline{IOI}\right)}{median\left(\overline{IOI}\right) + \left| median\left(\overline{IOI}\right) - \frac{\left(I_i - I_{last\_beat}\right)}{k} \right|} \right\} \cdot f\left(E_i\right) \quad (4)$$

where k =1, 2, 3. k is introduced to cope with the situation that the current *IOI* is 2 or 3 times longer than the predicted value due to a decreased tempo or a missed candidate. $\overline{IOI}$ is a vector of previous inter-onset intervals. The size of $\overline{IOI}$ is an odd number. $median\left(\overline{IOI}\right)$ is used as a prediction of the current beat. i is the current beat candidate index. I_i is the MP3 granule index of the current beat candidate. I_{last_beat} is the MP3 granule index of the previous beat.

$$f\left(E_i\right) = \begin{cases} 0, & E_i < threshold_i \\ 1, & E_i \geq threshold_i \end{cases} \quad (5)$$

where E_i is energy of each candidate. $f\left(E_i\right)$ is introduced to discard candidates having too low energy.

The confidence score of the converged beat stream R is calculated by

$$R = \max\left\{R_F, R_1, \cdots, R_N\right\} \quad (6)$$

where N indicates the number of subbands and F indicates the full-band. The dashed line in Figure 9 shows the converged confidence score of a pop music, which is used to reject non-beats.

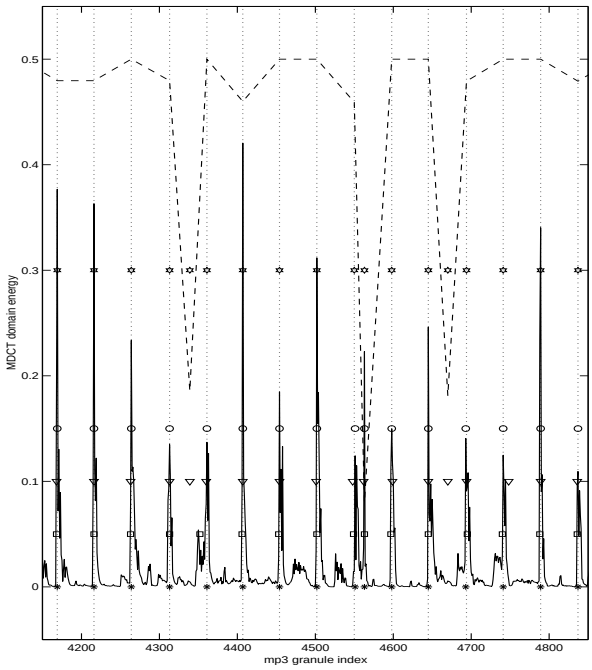

Figure 9. Multi-band features of a pop music sample: full-band energy (solid line), candidates from subband 1 (star), subband 5 (squares), subband 6 (triangles), and full-band (circles), converged beat candidates (hexagram), detected beats (dotted lines). The dashed line indicates the confidence score of the converged beat candidates, which is used to discard non-beats at this stage. For illustration purposes, the confidence score is shifted downwards by 0.5.

Converge and store beat information

Beat candidates together with their confidence scores from all the bands are converged. The candidate that has the greatest confidence score within a search window is selected as a center point. If candidates from other bands are close to the selected center point (less than 4 MP3 granules, for example), they are clustered. The confidence of a cluster is the maximum confidence of its members and the location of the cluster is the rounded mean of all locations of its members. All other candidates are ignored. As the final step, the candidate is accepted as a beat if its final confidence score is above a constant threshold. Beat position, IBI, and overall confidence score are sent to the application module after checking with the window

switching pattern.

5 PRELIMINARY EXPERIMENTAL RESULTS

We tested the proposed beat detector on 6 popular songs with durations from 1 to 6 minutes. The input was monaural audio signals sampled from a few commercial compact discs. The signals are then compressed using a MP3 encoder at bitrate of 64 ~ 96 kbps.

The system utilizes some basic musical knowledge to track beats at the quarter-note level. It assumes that beats generally have more energy than offbeats and the tempo is constrained to be between 50 and 180 M.M. (Mälzel's Metronome: the number of quarter notes per minute) and is roughly constant. Since the sampling frequency of all test sounds is 44.1 kHz, the MP3 granule length is 576, the time resolution of our system is ~ 13 ms (=576/44.1).

The beat annotation of the 6 test samples were performed by the second author, who is a M.Sc. student at Tampere University of Technology and a violinist at Tampere Conservatoire. The reason to choose a musician for beat annotation was that we wanted a more precise and consistent result.

The machine-detected results were then compared to human annotation. The criteria to count a failure were: (1) if the detected beat position departs from the annotation by or more than 4 MP3 granules that is ~ 52 ms; (2) if the algorithm simply fails to detect a beat; (3) if the algorithm picks up a non-beat as a beat.

The proposed method correctly tracked beats in 4 out of 6 popular music test signals.

We found that the algorithm worked almost without error if there was a simple strong bass drum pattern marking the beat such as songs from the band ABBA. However, the algorithm failed completely, if there was no clear drum beat or the beat pattern was rather complex. The algorithm often made some mistakes at the beginning of each music sample due to irregularity of the intro and at the end of each sample where the signal was fading away. If we disregard the beginning and the end for a few IBIs, the algorithm made only one mistake (missed one beat) during a 6-minutes test song, for example.

We discuss the reason why the beat tracker fails completely in two of the test samples. The algorithm relies on the assumption that the actual beats are in general stronger than the offbeats for example. If this assumption does not hold, the algorithm fails since it does not use any advanced musical knowledge. In particular, one failed sample has no drums and uses only a synthesized shaker sound marking beats. Another failed sample has rather complex beat-pattern. The bass drum beat varies a lot mixed with snare drum and hi-hats.

These preliminary results show that the proposed algorithm can deal with realistic music signals. However, some improvements are still necessary.

6 DISCUSSION

A beat tracking system is developed as a building block of an error concealment scheme. It should be noted that the error concealment and beat detection concept could be easily applied to cope with other audio bitstreams with minor modifications.

The beat detector was implemented partly in C and partly in Matlab. Its memory consumption and computational complexity are modest.

Essentially, the beat detection and error concealment are still two separate tasks. In order to reduce the complexity of the decoder, it might be a better alternative to implement the beat detector in the encoder side and to embed the current beat information in its preceding beat as ancillary data in the MP3 bitstream. The decoder could then directly use the beat information for error concealment. In this way we not only reduce the complexity of the decoder but also know with certainty whether the missing segment has a beat or not from the embedded beat information from its previous beat. Otherwise the decoder would have difficulties to guess a beat with damaged packets.

The algorithm does not work with signals such as speech and classic music. It is just intended for pop music with quite regular beat structure, which is an important class of music in streaming applications.

We believe that it may be a better option to use the 32-subband signal for beat detection instead of the 576-MDCT coefficients of a MP3 granule. This will not only improve the time resolution but also avoid the alias introduced by MDCT [6]. After all a beat is more a temporal phenomenon.

The current implementation is clearly an application oriented work in nature. We intend to port a good PCM domain beat tracker into the compressed domain to examine its performance shift.

The implemented system is still in its early version. There are many avenues open for further work. Because of its ad hoc nature, all major building blocks (e.g. the subband division, the statistical model and the confidence score) can be further optimized.

Another possible extension is to include more high-level musical knowledge into the system for better performance at the expense of complexity.

For applications other than error concealment, some modifications and optimizations may be necessary to satisfy the specific requirements.

ACKNOWLEDGEMENTS

The Academy of Finland and Nokia Foundation are

acknowledged for providing the first author scholarships which enabled him to conduct a major part of this research at Department of Experimental Psychology, University of Cambridge, UK under supervision of Prof. Brian C.J. Moore. We thank Mr. Juha Ojanpera for assistance with programming, and Dr. Jilei Tian, Mr. Jarno Seppänen, Prof. Anibal Ferreira, Prof. Brian C.J. Moore and three anonymous reviewers for helpful comments on an earlier version of this paper.

REFERENCES

1. Patel, N.V., Sethi, I.K. "Audio Characterization for Video Indexing", Proc. SPIE Vol. 2670, Storage and Retrieval for Image and Video Databases IV, Jan/Feb 1996, San Jose, CA, USA, pp. 373-384.
2. Nakajima, Y., Lu, Y., Sugano, M., Yoneyama, A., Yanagihara, H., Kurematsu, A. "A Fast Classification from MPEG Coded Data", Proc. of International Conference on Acoustic, Speech and Signal Processing (ICASSP), 1999, Phoenix, Arizona, USA, pp.3005-3008.
3. Boccignone, G., DeSanto, M., Percannella, G. "Joint Audio-Video Processing of MPEG Encoded Sequences", Proc. IEEE Intl Conf. On Multimedia Computing and Systems (ICMCS99), 1999, pp.225-229.
4. Pfeiffer, S., Robert-Ribes, J.; Kim, D. "Audio Content Extraction from MPEG-encoded sequences", First International Workshop on Intelligent Multimedia Computing and Networking (IMMCN2000), Feb./March 2000, Atlantic City, New Jersey, pp. 513-516.
5. Wang, Y., "A Beat-Pattern based Error Concealment Scheme for Music Delivery with Burst Packet Loss", accepted by IEEE International Conference on Multimedia and Expo (ICME2001), August, 2001, Tokyo, Japan.
6. Wang, Y., Vilermo, M., Isherwood, D. "The Impact of the Relationship Between MDCT and DFT on Audio Compression: A Step Towards Solving the Mismatch", The First IEEE Pacific-Rim Conference on Multimedia (IEEE-PCM2000), December 13-15, 2000, Sydney, Australia, pp. 130-138.
7. Wang, Y., Ojanpera, J., Vilermo, M., Vaananen, M. "Schemes for Re-compressing MP3 Audio Bitstreams", accepted by the Audio Engineering Society (AES) 111th International Convention, September 21-24, 2001, New York, USA.
8. ISO/IEC 11172-3, "Information Technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s", 1993.
9. Dannenberg, R.B., Mont-Reynaud, B., "Following an improvisation in real time," Proc. Int. Comp. Music Conf., 1987, pp.241-248.
10. Desain, P., Honing, H., "Advanced issues in beat induction modeling: syncopation, tempo and timing," Proc. Int. Comp. Music Conf., 1994, pp. 92-94.
11. Rosenthal, D., "Machine Rhythm: Computer Emulation of Human Rhythm Perception," PhD thesis, MIT, 1992.
12. Scheirer, E.D., "Tempo and beat analysis of acoustic musical signals," J. Acousti. Soc. Am., 1998, vol. 103, no.1, pp. 588-601.
13. Goto, M., Muraoka, Y. "Music understanding at the beat level: Real-time beat tracking for audio signals", in "Computational Auditory Scene Analysis", edited by Rosenthal D. and Okuno H., 1998, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA, pp. 157-176.
14. Todd, N.P.M., "The auditory 'primal sketch': A multi-scale model of rhythmic grouping," J. New Music Research, 1994, vol. 23, no. 1, pp.25-70.
15. Smith, L.M., "A multi-resolution time-frequency analysis and interpretation of musical rhythm," PhD thesis, University of Western Australia, 1999.
16. Dixon, S.E., "A beat tracking system for audio signal," Proc. Conf. Computat. And Mathemat. Methods in Music, Vienna, Austria, 1999, pp.101-110.
17. Klapuri, A., "Sound onset detection by applying psychoacoustic knowledge," Proc. IEEE Int. Conf. Acoust., Speech, and Signal Proc., 1999, vol. 6, pp. 3089-3092.