AREEB: *A*utomatic *REfrain E*xtraction for Thum*B*nail

Alessio Mellina, Alexandro Sentinelli
Advanced System Technology, STMicroelectronics
Agrate Brianza (MI), Italy
alessiomellina@gmail.com , alexandro.sentinelli@st.com

Gustavo Marfia, Marco Roccetti
Computer Science Department, University of Bologna,
Mura A. Zamboni 7, 40127 Bologna, Italy
marfia@cs.unibo.it, roccetti@cs.unibo.it

***Abstract*—This paper investigates a given branch of the music information retrieval field, specifically that of audio thumbnail, through which a piece of music can be automatically summarized. We first provide a survey on the different techniques available today for the generation of music summaries, and then we propose a practical method that, based on best practices, is able to summarize songs with a great efficacy.**

Keywords- Music Information Retrieval and Management; Music Summarization; Audio Thumbnailing; End User Experience

I. INTRODUCTION

One of the targets of Multimedia Information Retrieval (MIR) is to enhance, improve and simplify the interaction with this kind of data, thanks to the automatic extraction of information. MIR, in fact, focuses on the extraction of information from a wide variety of musical data, possible examples ranging from optical recognition of notes out of a printed score, to automatic recognition of a piece of music in the form of an audio file, to the transcription of lyrics from a piece of sung music, to the identification of musical instruments based on the analysis of timbre, just to name a few.

A popular example of such kind of technologies is Shazam, a popular mobile application that, given a short recorded piece of music, can univocally identify that song out of a wide database. Thanks to this technology, if a person listens to a song on the radio and ignores the name of the author or the title of that song, he can record a part of that song with a mobile phone, submit the recording to Shazam and obtain the information he is seeking. Additionally, he can also immediately buy that song, thanks to online music stores that can offer digital download.

In this ample context, a promising and rather new area of research is audio thumbnail: that is the technique through which a piece of music can be automatically summarized. In fact, despite how easy it can be to gain access to musical data, the task of browsing and performing queries on this type of data may not be as easy. Most queries are carried out with a text-form interaction (even though some new approaches, like *query by humming*, have already been successfully implemented [1, 2]) that may be limited in efficacy in certain cases. Furthermore, it is not always easy to establish whether the results of a query are exactly what one is looking for.

An example is the following: one is looking for a specific song, which has to abide by certain criteria; after performing a query over a database of songs, a subset of proposals are received; he/she then has to decide whether the song he/she looking for is present in this subset. The only way to find the desired song is hence to have a "preview", or better a "trial", listening to all the proposed songs. This can be frustrating (amounting to going back and forth looking for the segment that will be helpful).

A less time-consuming approach would provide us with an automatically generated summary of each song, so that it wouldn't take more than a few seconds to establish whether the song we are listening to is relevant or not. This is where audio thumbnails come into play. Several techniques have already been developed, implemented and tested, either with original systems or with already existing approaches mutated from previously implemented applications.

In this article, we first survey on the most interesting techniques available today for the generation of music summaries, our aim being that of touching upon the similarities and the differences of those already existing techniques. Then, exploiting the insights of our study, we propose an method of practical relevance that, based on best practices, provide good results when employed to summarize popular music. Many experimental results confirm the efficacy of our approach. To conclude this Section, we deem that these technologies can have a notable impact of practical relevance on the music entertainment industry, well beyond the interest that they have gathered from a research viewpoint in the academic field. The remainder of this paper is structured as follows. In Section II, we survey on existing techniques. In Section III we propose our method, while we report on some few experimental results in Section IV. We finally conclude the paper with Section V.

II. METHODS FOR AUDIO THUMBNAILING

We now present an overview of audio thumb-nailing techniques; we will then propose our best *practical* solution, based on a comparison with those alternatives.

The first step towards music summarization is answering the question "*What does music summarization mean*?". When trying to summarize a piece of music, a good solution is to present the most significant part of that piece of music. Since "significant" is quite arbitrary and it depends a lot on the listener, an assumption must be made: usually, the most relevant part of a piece of music is that part which is most *repeated*, since that is the part that is normally used by listeners to identify the whole piece. The *most repeated* part

can be the main theme in the case of classical music, or the chorus/refrain in the case of popular music and its subgenres [2] (i.e., the portion that is repeated the most throughout the whole song). This assumption brings to an obvious consequence: since we are looking for repetitive parts throughout a song, the songs we will be able to summarize will have to be repetitive somehow (the presence of recurring themes, phrases, or sections), otherwise we we will not be able to provide the listener with an accurate thumbnail. Therefore, this algorithm does not work well with all kinds of music: experimental or electro-acoustic music compositions, for example, cannot be easily summarized with this procedure. Other genres, such as popular music (and all of its subgenres) and some works from classical music, have instead a high percentage of success.

There are obviously notable advantages that such systems can bring to on-line music store, such as, for example, a better song identification, which means facilitating the choice of the product and thus, potentially, increasing sales. To witness this potential, it is enough here to cite a few software products, which are inspired by similar concepts, like Midomi, Soundhound, Shazam, RefraiD, and especially SmartMusicKiosk [3]; even if only a few of these exploit with efficacy thumbnails.

In the following Subsections, we provide insights on the main technical mechanisms used to get thumbnails, namely: Frame segmentation, Feature Extraction, Similarity Computation and Thumbnail Extraction.

A. Frame Segmentation

When segmenting a piece of audio data into frames (Figure 1), there are a few factors that must be taken into account: sampling rate, frame length, number of overlapping samples, hopsize. All these factors have a direct influence on the outcome of the algorithm. Some algorithms, like those in [2, 3], require that the audio file be resampled for a better processing, but usually the first step amounts to splitting the stream of samples into temporal frames (the size of this frame may vary from 10 ms to approximately half a second, depending on the exploited algorithms).

A windowing function is then applied as a second step 2, 3, 4] while the subsequent entails the conversion of the signal from the time domain to the frequency domain. A transform function (such as the FFT/STFT) is therefore used. In some cases [3, 5], a pass band filter is applied to the resulting spectrum (e.g. 20-2000 Hz, 130-8000 Hz); this helps selecting the frequencies that are likely to carry the most relevant spectral information of a temporal frame.

When using smoothing window functions such as the ones mentioned above, overlapping must be considered, according to the length of the frames. Usually, the size of such overlap is between 25% and 50% of the frame size. Hopsize length, too, is correlated to frame size and overlap size. A final refinement to the frame extraction phase is the introduction of a beat tracking algorithm as a preliminary operation [5]: with this algorithm one can obtain the tempo of the song, and thus adjusting the temporal dimension of the frame so that the frame length is a fraction of the song's tempo.

B. Feature Extraction

After dividing the stream of audio into single frames, it is necessary to choose a valid method for representing its content. Two main representations can be applied: the first implies the use of the Mel-Frequency Cepstral Coefficients (MFCC) [2, 6, 9, 11], while the second implies the use of the chroma vectors [3, 5, 7, 8].

Cepstrum is the power spectrum of the logarithm of the power spectrum of a function and represents a smoother version of the power spectrum of a signal. By calculating the cepstrum, one can extract from each frame a set of coefficients, called Mel-Cepstral features, that can be used to describe the frame. Chroma vectors, instead, are a spectral representation based on the chromagram, where frequency is a bidimensional representation of the pitch, formed by chroma (pitch class) and tone height (octave number). The chromagram (Figure 2) is a 12-dimensional vector (one dimension for each pitch class) where all the energy corresponding to a single pitch class at the different tone heights is summed to represent each of the 12 coefficients. Therefore, the chromagram amounts to a spectral analysis based on the musical scale: for this reason, such analysis is best suited for signals with strong periodic and harmonic behavior (i.e., music) [12].

C. Similarity Computation and Song Structure

After extracting features for each vector, similarity of features throughout the song must be computed. The most common approach implies that the similarity between each couple of frames is calculated. By computing this, one can later discover whether there are extended regions with relatively long series of one-to-one similarity. Various distance measures can be used to verify similarity, such as: Kullback-Leibler, Euclidean or cosine [2]. The results of this computation must be embedded in a Similarity Matrix (SM); this way, it is easier to identify long sections of similarity within a song. These sections are then usually clustered using a moving average filter. Another approach proposes the use of Hidden Markov Models (HMM) [9]: feature vectors are used as observations of potential states. A refinement on the number of states can be achieved using a K-means algorithm before applying the HMM [8]. Similarity, in this case, can be inferred as the repetition of sequences of observations, generated by hidden states. In both cases, we can obtain the structure of a song via clustering of similar sequences of observations (for the HMM) or computing the length of the matching diagonal segments (for the SM).

D. Thumbnail Extraction

When extracting the final thumbnail, one must make some assumptions in order to select the most relevant section of a song. First, the selected segment has to be the one that has the highest repetition rate, or it can be the longest section containing the most frequent segment. As an alternative, one

can impose that the section is repeated a minimum of 3 times [4] and a maximum of 4 times [5]. One can then determine the length of such segment: some propose a minimum of 5 s[4] and/or a maximum of 40 s[3], or one tenth of the song's total duration [5]. This length can be adjusted according to the database on which the algorithm operates and according to the final use of the thumbnail. An additional refinement assumes that the selected section lies in the first half of the song [9] : the second half of the song may in fact typically contain long instrumental parts that might be repetitive and might lead to errors.

III. AREEB: A NEW STATE-OF-THE-ART SOLUTION

We have just presented a quick overview of all the steps one must follow to obtain valid audio thumbnails for a given song. We are now going to propose a best *practical* approach as the sum of *best practices* for each single step.

A. Frame Segmentation with AREEB

Our first step is to resample the audio data at 16 kHz. This choice is correlated with the use of a band pass filter while extracting features from each frame. We then decided to use a beat tracking algorithm as an additional operation. Using a beat tracking algorithm gives us the possibility to match the frame length with a sub multiple of the musical bar (i.e., the length corresponding to a quarter note or an octave note): this way, we can obtain a more precise feature vector. This can be easily seen since melodic and harmonic variations throughout a song do not occur at random time instants: rather, they are correlated to the tempo and rhythm of a song. Furthermore, synchronizing the segmentation of a song with its tempo helps us to evaluate the frame-to-frame similarity between same parts of repeated sections: these frames will have exactly the same relative time location within their section. Among the alternative beat tracking algorithms, we selected the one presented in [10].

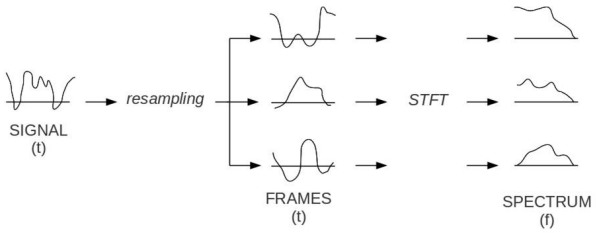

Figure 1. Frame Segmentation.

To refine the length of the frames, we impose that such length exceeds the duration of a quarter (or of an octave) of the calculated tempo by a ratio of 25%. This measure is used for overlapping, and accordingly the overlapping ratio will be 25% of the segment's length, with the aim of maintaining synchronization with the tempo of the song. For windowing, we used a Hanning window function. In summary, we resorted to a dynamic segmentation of a musical piece as it enhances the extraction of features, thus adapting segmentation to the musical context, based on the consideration that musical signal has a stationary behavior correlated to the rhythmic changes of a song.

B. Feature Extraction with AREEB

For each frame, we extracted a vector of features corresponding to the chromagram (as in Figure 2).

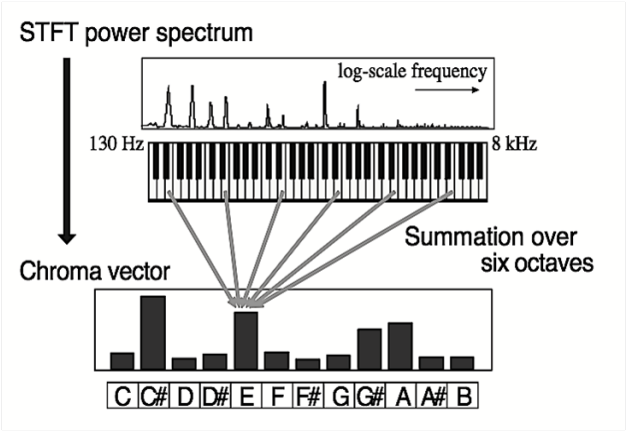

Figure 2. Chroma vector's computation.

We choose the chromagram over cepstral coefficient for one main reason: cepstrum is a generic analysis that aims at deconvoluting the main signal from its echoes, or reverbs. The chromagram is a spectral analysis specifically designed for musical signals, and can better represent the frames we have previously computed. Before calculating the chromagram, a bandpass filter (from 130 Hz to 8 kHz) is applied. This restriction helps us isolate the relevant portion of the spectrum, since this is the band where we can find most of the melodic and harmonic contents. For the chromagram vector, each of the 12 dimensions is calculated as follows:

$$vc_c(t) = \sum_{h=Oct_L}^{Oct_H} \int_{-\infty}^{+\infty} BPF_{c,h}(f)\Psi_p(f,t)df \qquad (1)$$

where the $BPF_{c,h}(f)$ is a bandpass filter that passes the signal at a log-scale frequency $F_{c,h}$ (in cents) of pitch class c in octave number h,

$$F_{c,h} = 1200h + 100(c-1) \qquad (2)$$

This filter is defined using the following Hanning window:

$$BPF_{c,h}(f) = \frac{1}{2}\left(1 - \cos\frac{2\pi\left(f - \left(F_{c,h} - 100\right)\right)}{200}\right) \qquad (3)$$

The result is a 12-dimensional vector, where each element of the vector *v(t)* corresponds to the summation of energy of a pitch class *c* over the considered octaves *L* to *H*. Since we have previously filtered the signal with a bandpass filter, the octaves we consider now are 6 (from octave 3 to 8).

C. Similarity Computation with AREEB

We must now compute the similarity (i.e., the distance) between each pair of vectors, in order to embed the results in

a self-SM. For this calculation, we adopt the simple scalar product:

$$d_c(v_i, v_j) = \frac{\langle v_i, v_j \rangle}{\| v_i \| * \| v_j \|} \quad (4)$$

where d_c is the cosine of the angle between the feature vectors v_i and v_j. The results of this operation are included in a self-SM (as shown in Figure 3, for example). This matrix will have maximum values on the diagonal. Additional diagonal lines that will appear in the matrix will represent regions that bear a high similarity.

It is possible to detect the similarity patterns by summing the columns (SbC) or performing moving averages (MA) along the diagonals of the matrix.

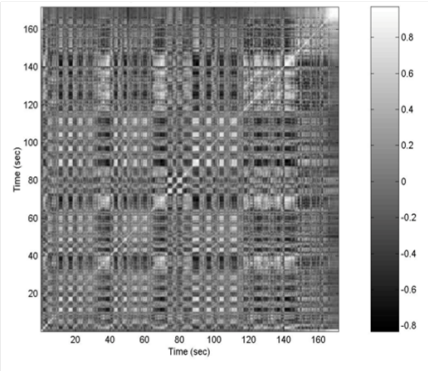

Figure 3. SM (Magical Mystery, Beatles).

We actually tried either approaches; however, empirical results have shown an improvement of 15% in refrain detection while using the MA technique. In general the SbC has positive results for those songs that have internal minor repetition in the same refrain, but also some false positive for long non relevant sequences.

The MA needs to detect similarity between extended regions of the song: the results are once again embedded in a time-lag SM. To reduce the computational complexity of this operation (some millions of operations for 3 minutes length song, and many not sequential access to the memory) we exploited the property moving average as follows:

$$MA_{new} = MA_{old} - \left(\frac{x_{i-n}}{n} - \frac{x_i}{n}\right) \quad (5)$$

Then we made some realistic assumptions about the "location" of the refrain in popular songs to make the algorithm both more robust and less expensive: the refrain has to be not too long, not too short, thus in an time interval between 10 and 45 seconds. We decided the length was L_{MA}=30 seconds. Then, we applied a computation mask over the SM to reduce the amount of MA computed (Figure 4) by considering that, in general, the majority of popular songs has the refrain repeated a couple of times in the first part of the songs.

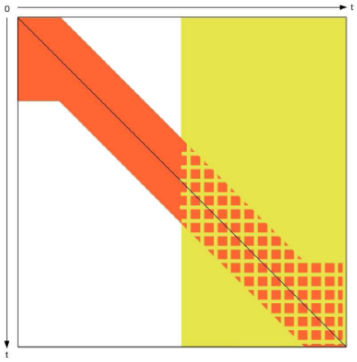

Figure 4. Computational Mask for the MA.

Then we ignored the similarity among too close segments by setting a sort of shadow region around the main diagonal that avoids for example to compare the versus from the refrain.

D. *Thumbnail Extraction with AREEB*

With the help of the SM, we can now identify the section of the song that will become the thumbnail. This thumbnail must include the most repeated section of the song (to select the section among the various repetition, we can decide to pick its first appearance). There are some additional refinements that we adopted to get a better thumbnail: first, the considered repeated section must have a length between 10 and 40 seconds, so that we can avoid considering shorter repeated sections with minor relevance. We can also decide to check for the repeated sections in the first half of the song only, in order to avoid instrumental parts that could lead to errors. In the case of popular music, each chrous is likely to be formed by two almost identical halfs: these results in the SM as diagonal segments composed by two similar subsegments. At this point, it is sufficient to locate and store the start and end points of the selected segment: these will be the bounds of our audio thumbnail. With these labels we can gain access to the thumbnail of each song whenever needed, and it is not necessary to recalculate the audio summary each time.

IV. END USER EXPERIMENTS

The addressed question here is: "*Would you use this summary to represent the song?*" We made experiments to confirm the efficacy of our approach with a set of real *music listeners* (end users). We benchmarked our application through a qualitative test on a both a generic database and another personally customized with respect to the listener.

Table I. Summary scores.

Feedback	# songs	% success
(NA) Not Acceptable	5	0
(PA) Partially Acceptable	5	5
(A) Acceptable	40	80

A. Test on a Generic Database

We set up an archive of ~50 songs of different kinds of music and recording style (live vs studio) and technology (digital vs analogical), all of them rather popular. By scoring with 3 possible alternatives (*Acceptable*, *Partially Accept*able, *Non Acceptable*), we collected the results as expressed in Table I. Of 50 songs we got 40 successful summary extractions, 5 partials, 5 negatives. It is interesting to remark a particular negative result ("Teardrop", Massive Attack) that brings out indeed the weakest point of AREEB. Songs with popular and clear refrains, fail miserably if they have also long drum sessions, or pure rhythmical music sequences, that are long enough to cover a sufficient portion of the song (in particular enough to cover L_{MA}), as our method gives very high recognition pattern scores to those rhythmical parts.

B. Qualitative Customized Test

The user was asked to give a score to AREEB's summaries obtained from two different sets of 10 well-known songs usually played in his own music archive. We used a score similar to the previous, having simply remapped [(NA), (PA), (A)] into [0, 0.5, 1]: the score was averaged on the ten songs. Results are shown Table .

Table II. Individual test summary.

user	%success part I	%success part II
1	85	80
2	75	70
3	85	80
4	90	90
5	75	85
6	70	80
7	65	90
8	80	80
Avg	**78.125**	**81.875**
StDev	**8.43**	**6.51**

Although the statistics is still small to get a clear understanding of AREEB's behavior, the results of both sets of tests can be considered as rather promising.

V. CONCLUSION

Based on *best practices*, we have devised and experimented with an an efficient tool for extracting audio summaries from popular songs [13, 14]. A demo of our software prototype can be downloaded from [15]. We believe that the relevance of such techniques will become key for many different application scenarios, as witnessed also by the consistent amount of research that is being devoted to multimedia home entertainment [16]-[20].

ACKNOWLEDGEMENTS

Our acknowledgements go to the Italian Projects DAMASCO (FIRB) and ALTER-NET (PRIN) projects.

REFERENCES

[1] B. Pardo, J. Shifrin, W. Birmingham, "Name that tune: a pilot study in finding a melody from a sung query", *Journal of the American Society for Information Science and Technology*, vol. 55, n. 4, February 2004, pp. 283-300.

[2] M. Cooper, J. Foote, "Automatic music summarization via similarity analysis", in *Proc. ISMIR*, Paris, 2002.

[3] M. Goto, "SmartMusicKIOSK: music listening station with chorus-search function", in *Proc. the ACM 16th Annual Symposium on User Interface Software and Technology* (UIST'03), Vancouver, Canada, pp. 31-40.

[4] J. Wellhausen, M. Höynck, "Audio thumbnailing using MPEG-7 low level audio descriptors", in *Proc. the SPIE ITCOM*, Orlando, 2003.

[5] M. Bartsch, G. Wakefield, "To catch a chorus: using chroma-based representations for audio thumbnailing", in *Proc. of the IEEE 2001 Workshop on the Applications of Signal Processing to Audio and Acoustics* (WASPAA'01), New Paltz, NY, pp. 15-18.

[6] D. Childers, D. Skinner, R. Kemerait, "The cepstrum: a guide to processing", in *Proc. of the IEEE*, IEEE, vol. 65, n. 10, October 1977, pp. 1428-1443.

[7] G. Wakefield, "Mathematical representation of joint time-chroma distributions", in *Proc. SPIE*, Denver, 1999.

[8] G. Peeters, A. La Burthe, X. Rodet, "Toward automatic music audio summary generation from music signal analysis", in *Proc. ISMIR*, Paris, 2002.

[9] B. Logan, S. Chu, "Music summarization using key phrases", in *Proc. of the 2000 IEEE Conference on Acoustics, Speech and Signal Processing* (ICASSP'00), Istanbul, Turkey, pp. 749-752.

[10] A. Pikrakis, I. Antonopoulos, S. Theodoridis, "Music meter and tempo tracking from raw polyphonic audio", POLYMNIA-EPAN project, Athens, 2004

[11] M. McKinney, J. Breebaart, "Features for audio and music classification", in *Proc. ISMIR*, Eindhoven, 2003

[12] J. Martinez, R. Koenen, F. Pereira, "MPEG-7: the generic multimedia content description standard", *IEEE Multimedia*, IEEE, vol. 9, n. 2, April-June 2002, pp. 78-87.

[13] R. Dannenberg, N. Hu, "Pattern discovery techniques for music audio", in *Proc. ISMIR*, Paris, 2002

[14] C. Burges, D. Plastina, J. Platt et al., "Duplicate detection and audo thumbnails with audio fingerprinting", Microsoft Research Technical Report, Redmond, 2004

[15] "AREEBdemo.avi", http://dl.dropbox.com/u/2931988/AREEBdemo.avi, accessed July 11, 2011

[16] G. Marfia and M. Roccetti, "TCP At Last: Reconsidering TCP's Role for Wireless Entertainment Centers at Home", *IEEE Transactions on Consumer Electronics*, IEEE Consumer Electronics Society, vol. 56, n. 4, November 2010, pp. 2233-2240.

[17] G. Marfia and M. Roccetti, "Dealing with Wireless Links in the Era of Bandwidth Demanding Wireless Home Entertainment", in *Proc. of the 2010 IEEE International Conference on Multimedia and Expo (ICME'10)*, IEEE, Singapore, July 2010, pp. 1376-1381.

[18] C.E. Palazzi, M. Gerla, G. Pau, G. Marfia, M. Roccetti and M.Y. Sanadidi, "Balancing Video on Demand Flows over Links with Heterogeneous Delays," in *Proc. of the 3rd ACM International Mobile Multimedia Communications Conference* (Mobimedia'07), ACM, August 2007, Nafpaktos, Greece, pp. 1-6.

[19] C. E. Palazzi, S. Ferretti, S. Cacciaguerra and M. Roccetti, "A RIO-like Technique for Interactivity Loss Avoidance in Fast-Paced Multiplayer Online Games," *ACM Computers in Entertainment*, ACM, vol. 3, n. 2, April 2005, pp. 1-11.

[20] A. Cattaneo, G. Marfia, A. Sentinelli, A. Vitali, L. Celetto, M. Roccetti and M. Gerla, "Using Digital Fountains in Future IPTV Streaming Platforms: a Future Perspective," *IEEE Communications Magazine*, IEEE Communications Society, 2011, to appear.