Description of audio tempo extraction final project
Simon de Leon
McGill University, MUMT611 2/16/06

1.0 Introduction

The main goal of this project was to implement the tempo extraction algorithm described by Alonso et al. (2004). Its implementation in MATLAB was made to comply with the instructions for the MIREX 2005 audio tempo extraction competition. Applying the algorithm to the competition's training data returns results roughly on par with the results achieved by the similar algorithm in MIREX 2005 on the larger testing data set.

This document provides a brief overview of the algorithm of interest, along with a discussion of this project's organization, software, results, and future work.

2.0 Algorithm

The algorithm can be decomposed into three sub-blocks: onset extraction, periodicity estimation, and temporal estimation of beat locations. Onset extraction is the process of locating the hypothetical beats of the music. The periodicity estimation extracts the tempo from these hypothetical beats. Finally, the temporal estimation of beat locations aligns a pulse train at the extracted tempo to the music.

2.1 Onset extraction

Onset extraction is concerned with the location of the most salient features of the music. Salient features typically correspond to note changes, percussive events, and harmonic shifts. To locate these features, the time derivative of the frequency component magnitudes is taken and summed at each time index. This is known as the spectral flux, and typically returns a detection function with semi-periodic spikes corresponding to the beat onsets of the music.

To reduce false beat onsets, the frequency component magnitudes are low-pass filtered according to a half-Hanning window magnitude curve and then logarithmically compressed (Klapuri 1999). These settings are informed by psychoacoustics. The derivatives are obtained using an eighth-order FIR filter differentiator as described by Proakis and Manolakis (1995). Finally, there is a dynamic threshold applied to the spectral flux that consists of a dynamic threshold set to the median of 25 local samples.

As an alternative to the low-pass filtering, it might prove useful if the frequency component magnitudes and derivatives were weighted according to frequency masking and the critical bands. This would help eliminate the contributions of perceptually irrelevant information to the spectral flux.

2.2 Periodicity detection

The periodicity detection scheme is applied to the detection function and extracts a tempo. Two methods were studied and described in the presentation of the algorithm: autocorrelation and spectral product. The autocorrelation is the classical periodicity estimation tool that works by cross-correlating the detection function with itself. Alternatively, the spectral product takes the FFT of the detection function and determines the frequency with the largest product sum of integer multiples. This frequency is assumed to be

the frequency of the beats. In the study presented by Alonso et al. (2004), the auto-correlation outperformed the spectral product consistently across all genres.

2.3 Temporal estimation of beat locations

Using the extracted tempo and the detection function, a pulse train is generated and mixed with the original signal to generate a metronome track. The pulse train uses the period of the locally captured tempo, and is cross-correlated with the detection function to ensure phase accuracy. Note that the correlation was performed directly in the time domain.

When beats are missing or not found within a given tolerance of their expected locations, the algorithm works to re-align the phase of the generated pulse train with the detection function. This can be a problem with music that has short silent breaks, since the algorithm will stutter and fail to produce a pulse in the empty space.

2.4 Tracking optimum tempo tracks

Although not described by Alonso et al. (2004), this section concerns the dynamic programming algorithm briefly described by Alonso et al. (2005) for the MIREX 2005 competition. In order to provide two weighted tempo choices as required by the MIREX protocol, a majority rule decision maker was applied to the periodicity results of the autocorrelation, spectral sum, and spectral product methods over sections of the detection function.

This feature was not completed due to time constraints but is considered key for a practical application of this algorithm when only one tempo can be applied. The method implemented in this project extracts tempo from an analysis on the beginning of the detection function, and performed quite well due to the fact that the training data was composed of relatively short clips with steady tempos.

3.0 Organization

The software project is divided into four folders at the top layer: *doc*, *results*, *train*, and *src*. The *train* folder was obtained from the MIREX 2005 competition website and contains 20 monophonic audio clips at 44.1 kHz. Additionally, groundtruth data is provided in text files following the format “[T1] \t [T2} \t [ST1] \t [P1] \t [P2] \t [M]”. T1 and T2 are the extracted tempos with their normalized relative strength indicated by ST1. The M value provides a multiplicity factor that is used to determine acceptable integer multiples or fractions of the groundtruth tempos. Finally, P1 and P2 provide the phase offset in seconds. Note that ST1 is fixed to 0.5 since the dynamic programming algorithm used to weigh tempos was not implemented

The *results* folder contains the output text and audio files from the MATLAB simulation. The format for the text data is similar to the groundtruth data format except without the multiplicity factor. Audio files are the results of printing a metronome track at the extracted tempo onto the audio clip with dynamic phase alignment. Their filenames are appended with “_1” and “_2” to indicate which extracted tempo was used to generate the added metronome track. The files *missedphase.txt* and *missedtempo.txt* provide details on the failed attempts as “[NUM] \t [R1] \t [R2] \t [T1] [T2]”, where NUM is the training data number, R1 and R2 are the results, and T1 and T2 are the groundtruth values. Also included in this directory is *summary.txt*, which contains four numbers in the format “[CT1] \t [CT2] \t [CP1] \t [CP2]”. CT1 is the percentage of results where at least one of the tempos was correct, and CT2 is the percentage of results where both tempos are correct. Similarly, CP1 and CP2 provide percentages for correctness although the phase results are suspicious for reasons discussed later.

4.0 Software

The provided MATLAB scripts implement onset detection, dynamic threshold filtering, periodicity detection, phase extraction, metronome overlay and all the evaluation routines. Of these scripts, the calculation of the spectral flux for onset detection was provided courtesy of Miguel Alonso (*SEF_true.m*).

The main calling script *evaluate.m* loops through all the training data and writes individual results into the *results* folder. It also compares the extracted tempos and phases with the groundtruth data and writes the necessary summary files. Most importantly, there are user modifiable constants at the top of the file that allow the user to specify the training clips used ('clips'), the metronome sound ('tickfile'), and whether or not the simulation will output new audio files with metronome tracks ('writeaudio').

Of all the supporting functions, the most important one for tempo extraction is *periodicity.m*. This function uses the autocorrelation method to determine the period of the detection function. It starts by finding the three largest correlation peaks within the minimum and maximum ranges. For the training data, these limits correspond to 210 bpm and 30 bpm, respectively. In Alonso et al. (2004), it is simply stated that a multiplicity relationship between the three largest lags is searched. However, some deviation was taken for the purposes of providing two tempos in following with the MIREX 2005 protocol. Essentially, the relative strength of the three peaks decide how the periodicity will be determined from them, be it their actual lag value or a difference from a neighbouring lag peak. This method was determined empirically from the detection functions of challenging audio clips, and is the primary reason why the strength factor ST1 in the results output is fixed.

5.0 Discussion of results

Preliminary results indicate that at least one correct tempo was found 95% of the time across the training data. Both tempos extracted were correct 40% of the time. The phase results indicate 100% accuracy but it is believed that this number may off due to the difficulty observed from the output audio results. Regardless, the tempo results closely resemble the MIREX 2005 results and indicate a high degree of correlation between the two implementations. Although the training data set is a small sampling of the competition's testing data set, it is considered to be indicative of the wide variety of genres and tempos tested.

One file from the training data failed to provide a satisfactory tempo (*train18.wav*). This file is a solo vocal with lightly strummed guitar that caused spurious detection function peaks from sung syllables. This illustrates a general weakness of the spectral flux method when applied to quieter, non-percussive music. Percussive events provide the greatest rate of change of spectral energy, and their periodicity in relation to the perceptual tempo is crucial for success. For music such as samba and broken beat, the percussive events are related to the perceptual tempo by odd fractions and also result in false tempos.

6.0 Future work

The algorithm needs further work to allow strength weighting between extracted tempos in order to be fully compliant with the MIREX 2005 evaluation protocol. The phase detection area also needs to be further investigated, since most of the algorithms tested in the MIREX 2005 competition performed poorly in

phase detection. Furthermore, the current implementation extracts the tempo from the beginning of the audio input and then dynamically aligns the following metronome locations to the detection function. It would be more accurate if the entire audio file was taken into account. This would smooth over difficult spots such as silence and drum fills. Finally, it would be very interesting to develop an automatic beat-matching application that mixes and blends songs together like a DJ.

7.0 Conclusion

The goals as originally stated for this project have been completed. However, further goals were realized during the tuning stage of the development, including the need to take the entire audio file into account when extracting tempo and the difficulties in conforming to the MIREX 2005 specifications. In particular, there are remaining issues concerning phase and its evaluation that need to be addressed.

Tuning a single low-level algorithm to be accurate across a wide range of genres is difficult and is most important for music cataloguing and data mining. However, the algorithm as it stands would work very well with electronic DJ music and could possibly be applied in an effective automatic DJ program.

References

Alonso, M., B. David, and G. Richard. 2004. Tempo and beat estimation of musical signals. *Proceedings of the 5th International Conference on Music Information Retrieval*.

Alonso, M., B. David, and G. Richard. 2005. Tempo extraction for audio recordings. *The Music Information Retrieval and Music Digital Library Evaluation Project.*

Klapuri, A. 1999. Sound onset detection by applying psychoacoustic knowledge. *Proceedings of the IEEE International Conference of Acoustics, Speech and Signal Processing*: 3089–92.

Proakis, J., and D. Manolakis. 1995. *Digital signal processing: Principles, algorithms and applications*. 3rd Ed. New York: Prentice Hall.

Laroche, J. 2001. Estimating tempo, swing and beat locations in audio recordings. *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*: 135–8.

Paulus, J., and A. Klapuri. 2002. Measuring the similarity of rhythmic patterns. *Proceedings of the International Symposium on Musical Information Retrieval.*