Assignment 6: Beat Tracking

CS 4347: Sound and Music Computing

due Wednesday 18 March 2015, 11:59 pm

NOTE: For inquiries on this assignment, please contact ZHU Shenggao (a0107950@nus.edu.sg).

0. In this activity you will learn to design a basic beat tracking algorithm. You can download an excerpt of the song 'Moskau', by Deschingus Khan, of approximately 15 seconds in length from the following link:

   `http://www.comp.nus.edu.sg/~duanzy/media/Moskau.wav`

   Over the course of this assignment, you will take this file and calculate an accent signal, estimate its periodicity, and finally determine the locations of its beats.

1. You may begin by making a copy of your previous Python assignment. Modify this program to do the following:
   - Read the file 'Moskau.wav'.
   - Load the wav file and split the data into frames. Each frame should be 1024 samples long and have 50% overlap. Only include complete buffers (e.g., if the final buffer has 1020 samples, omit that buffer).

2. You will now implement the first major component of the beat tracker, the accent signal:
   - First, for each frame, apply a Hamming window first, then use a Fast Fourier Transform operation to calculate its spectrogram, $S[n]$.
     - Remember that this operation gives you "positive" and "negative" frequencies, whose values are mirrored around the Nyquist frequency. Discard the negative frequencies (array indices above N/2 for an FFT of length N).
   - Next, calculate the value of an accent signal A according to the following equation:

$$A[n] = \sum_{k=0}^{N/2} HWR(|S[n]| - |S[n-1]|), n = 1, 2, 3, ... \quad (1)$$

   - Where k is the index of each frequency bin, and HWR is an operation in which all negative values are set to 0. Its full name is 'half-wave rectification'.
   - In other words, you will first calculate the difference between the magnitude spectrograms in frame n and frame $n-1$. You will then set any negative values to 0, and then sum over all frequencies in the frame. The result will be a single value, which is the accent signal value for that frame.
   - Note: when calculating $A[n]$, start at $n = 1$, and ignore $n = 0$. I.e., the length of $A[n]$ array will be one less than the # of frames.

3. As the next step, estimate the periodicity of the accent signal.

- Do this by first autocorrelating the accent signal, and get the resultant *autocorrelation of the accent signal* (AAS).
  Hint: the autocorrelation function in Python is as follows:

```
def autocorr(x):
    result = numpy.correlate(x, x, mode='full')
    return result[result.size / 2:]
```

- Next, you will find the index of a local maximum of the autocorrelation of the accent signal (AAS). This index is related to the accent signal's periodicity.
  - The period of the accent signal means the same as the period of the audio signal. Given the audio period p in seconds, the audio tempo T in Beats Per Minute (BPM) can be calculated as $T = 60/p$.
  - The relationship between an autocorrelation index (an index in AAS) and the tempo T of the audio is

$$Index = \frac{60}{T * L} \tag{2}$$

    Where L is the time interval between successive frames in seconds, and the tempo is in Beats Per Minute (BPM). Each frame is shifted by 512 samples at 44,100 samples/second, or equivalently, by about .0116 seconds. Therefore, L = .0116 seconds (approximately).
  - Note that AAS has several maxima. You will therefore need to find a sensible range of AAS within which to search for a maximum.
  - You may assume the true tempo of this music is between 60 and 180 BPM. Convert 60 and 180 BPM to autocorrelation indices, and use these two indices to determine the sensible range of AAS.
  - Find the index of the local maximum value of the AAS within this range. This index is the approximate number of frames between beats in this audio. This index will be called the *tempo index*, or t for short.
    Hint: In order to find the index of the maximum of a numpy array x, you may use the argmax function:

```
index_max = x.argmax()
```

4. You will now create the algorithm to determine the locations of the beats in this piece of music.
   - Make your program search within the first t values of the original accent signal (not AAS) for a maximum value. For example, if your tempo index is 110, then search among the first 110 values. The index of this maximum value will be your first beat index.
   - Starting at the first beat index, your system should skip ahead by the t. In other words, if your first beat index is 50 and your tempo index is 110, your system should skip to index 160 of the accent signal.
   - Your system should then look for a maximum within 10 values on both sides of its new index. For instance, if your system's position is index 160, then the system should check for a maximum from index 150 to 170. The index of the maximum that you find will be your next beat index.
   - Repeat the previous two steps until you have stepped through the entire accent signal and have recorded a list of all of the beat indices.
   - When this has completed and you have all of your beat indices (in frames), you must convert them to times in seconds. This is a simple multiplication operation:

$$Time = Index * .0116 \quad (3)$$

- This equation works because the first time frame ranges from 0 seconds to .0223 seconds, with an average value of .0116 seconds, and each subsequent frame is shifted forwards in time by .0116 seconds.
- Write your beat times to file using the following line of code:

```
beat_time.tofile('beat_time.csv', sep=',')
```

5. Write a file called beattrack.txt and discuss the following:
   - In 1-2 sentences, describe the output. Did the system beat track the music correctly? (Hint: you can plot out the audio/accent signal in time and your calculated beats times. Then see if the beat times are reasonable or not?)
   - In 2-3 sentences, describe two assumptions that this system made about the music. For example: what would have happened if the tempo of the music had changed in the middle of the piece?
   - In 1-2 sentences, explain why you think the 'beat estimation' step searched for maxima within a narrow range instead of simply incrementing by the beat index. Hint: consider the resolution of .0116 seconds/frame. Is it likely that the period is exactly an integer number of frames? Is it likely that the musicians will play to a perfectly exact tempo?

6. Hint: you can compare your own outputs with the following results (but you should not copy and hard-code these results directly in your program):
   - Total # of frames: 1307
   - Total # of accent signal values: 1306
   - Tempo index: 39
   - Total # of beats: 33
   - First beat index: 33

7. Upload your result file beat_time.csv to

   `http://cs4347.smcnus.org`

   This will automatically grade the values you calculated. If any mistake is found, please check your program and resubmit. You are welcome to submit as many versions as you wish before the submission deadline.

   Submit a zip file containing your program's source code (as a .py), your discussion file, and an optional README.txt file to the same website. You may use anything in the python standard library, numpy (including pylab / matplotlib), and scipy libraries. No other libraries are permitted.

   Grading scheme:

   - **3/6 marks**: correct submission file (automatically graded by computer).
   - **2/6 marks**: readable source code (good variable names, clean functions, comments when needed).
   - **1/6 marks**: Discussion