AES-53 Tutorial exercise: Complete the worked example

Note: The code completed so far can be found in a public BitBucket repository with URL

http://bitbucket.org/cannam/aes53-tutorial-bits

Review: What we already did

We have been starting to make a simple music tempo estimator.

We completed the following steps:

- Cloned a repository with some test data and some basic Python modules to use;
- In an IPython Notebook, wrote "framer" functions to split an audio sample array up into non-overlapping frames of a certain fixed number of samples each;
- Wrote and ran unit-tests for these functions;
- Used the functions to obtain frames from a test audio file;
- Wrote an "onset detection function" to turn a series of audio frames into a series of values, per-frame, estimating of the "novelty" of the frame compared to the previous one;
- Applied this detection function to the frames from the test file, and plotted the result;
- Calculated the autocorrelation (using a supplied function) of the detection function, and plotted the result;
- Pulled out the framer functions into a new module (as "supporting code" rather than "experimental code") and re-ran the experiment using this new module.

We have seen that the autocorrelation function has several peaks. The first and biggest is at the zero'th autocorrelation lag – that is, the correlation of the detection function with itself, unshifted. The next peak is likely to correspond to some repeating structure in the music itself, and its lag can be converted to a tempo estimate.

There are functions provided in the signal_processing module in the test repository to convert autocorrelation lag to a beats-per-minute value (lag_to_bpm) and the reverse (bpm_to_lag).

Our task

Add a function that:

- Accepts a beats-per-minute range (that is, high and low values, for example 50 to 170) and converts those to autocorrelation lag distances using bpm_to_lag;
- Finds the peak autocorrelation value inside that range of lag distances;
- Converts the peak lag to bpm, obtaining a tempo estimate for the audio recording.

Call this function in the notebook using the autocorrelation already calculated. Try this process with the 120bpm.wav test file and check that it returns 120bpm.

At each significant step, save the notebook and commit it to version control in the repository.

At the end, push the repository to a public hosting service such as BitBucket. (Note that BitBucket supports private projects if you don't want to make a public one.)