BEAT DETECTION PROJECT

COMPUTER MUSIC - REPRESENTATIONS AND MODELS
Music and Acoustic Engineering

Giovanni Agosti 928366
Antonio Orrù 926795

Goal

Detecting the bpm and the meter of a drum groove contained in an audio track.

Approximations and limitations

- The algorithm works only with perfectly timed drums groove (i.e. electronic drum grooves in which each note is perfectly aligned with the time grid)
- The algorithm detects bpm and meter of the first bar and propagate them on the whole track, so it's not able to detect bpm oscillations or metric modulation within the track.

We believe that both limitations could be overcome with a more sophisticated algorithm and we'll point out hypothetical solutions for this problems within the paper.

Meter formalization

Because of the huge variety of possible grooves and of the ambiguity of the musical notation (for instance it would be difficult to formalize the difference between a $\frac{2}{2}$ and a $\frac{4}{4}$ or between a $\frac{3}{2}$ and a $\frac{3}{4}$) [1] we decided to adopt the following formalization:

- BARS can be composed by 3, 4, 5 or 7 BEATS
- BEATS can be divided in 2 (straight $\frac{1}{8}$ notes) or in 3 ($\frac{1}{8}$ notes triplet)

In this way we can cover most of the time signatures of western music, for example:

- a bar of 3 BEATS with a straight subdivision is a $\frac{3}{4}$ or a $\frac{6}{8}$
- a bar of 3 BEATS with a triplet subdivision is a $\frac{9}{8}$
- a bar of 4 BEATS with triplet subdivision is a $\frac{12}{8}$
- etc...

Of course, there are still some ambiguities, like between $\frac{12}{8}$ and $\frac{6}{8}$ and so on.

We are anyway assuming the grooves to be clear grooves with a clear metric subdivision and with a backbeat kind of interpretation. This algorithm doesn't work with, for instance, a Latin groove or an Afro groove, where mostly nothing is played on beat one.

[1] We also find ambiguous the notation $\frac{3}{4}$ itself because in fact it would rather be a $\frac{3}{3}$, meaning a bar composed of 3 beats.

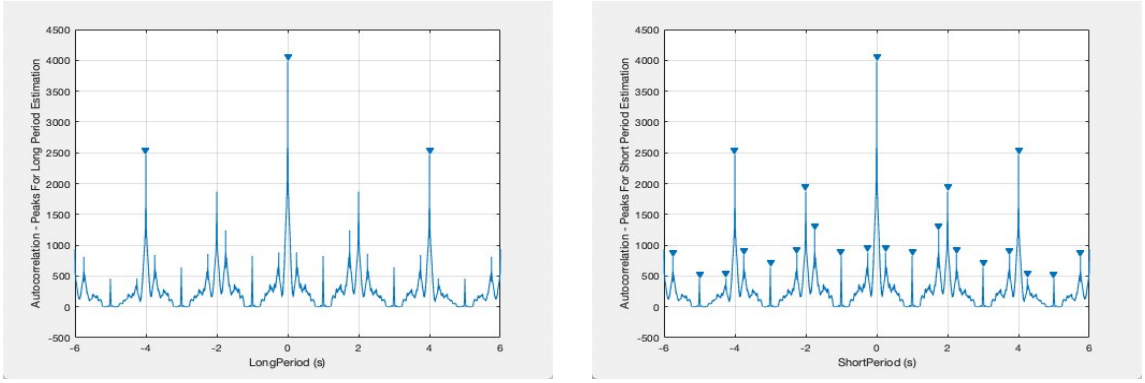
The bpm value is limited in the range $75 \leq \text{bpm} < 150$ and whenever the value is not within this range is simply multiplied or divided by 2 until it gets in the range.

Procedure

The algorithm is based on observing the peaks of the auto-correlation of the signal:
the maximum peak of the function will be in correspondence of the time lag after which the groove repeats, so it will correspond to the time duration of one bar (or 2).

Because of how musical meters are constructed (they are based on an integer subdivision of the bar and of the beats), the auto-correlation will also exhibit minor peaks corresponding to time lags which have an integer relation with the time duration of the bar. By knowing the duration of the bar and how it is subdivided we can determine the meter and the bpm.

For example in "Groove1", which is a very simple $\frac{4}{4}$ groove at 120 bpm, we can see that the maximum peak of the auto-correlation is at 4 seconds, meaning the groove has a period of 4 seconds, which in this case corresponds to 2 bars. We can also observe smaller peaks at integer dividend of 4 seconds such as at 2 seconds, 1 second and also at smaller values, which are still dividend of 4.



(a) Bar length (b) Subdivision

Figure 1: Peaks of auto-correlation

By calling *LongLength* the hypothetical duration of the bar, and *ShortLength* the first (closest to zero) small peak detected (other peaks could be tested in this stage for a more sophisticated algorithm), we can simply compute the subdivision of the bar as:

$$subdivision = round(LongLength/ShortLength)$$

If the subdivision is a multiple of 3 we can either have a bar of a 3 beats or a triplet subdivision of another meter. We can test this either by checking other small peaks of auto-correlation or by assuming (like we did for simplicity) that the bpm of the track has an integer value.

If subdivision is not a multiple of 3 than it will correspond to the meter (or a multiple of the meter) and we just need to verify that it falls in the range $3 \leq \text{meter} < 8$ and if it doesn't, we simply multiply or divide by 2 until it does.

We can finally obtain the value of the bpm as:

$$bpm\_hypothesis = round(60/(LongLenght/subdivision))$$

In this stage we are often rounding values in order to avoid numerical errors.

Finally, a metronome sound at the correct bpm is generated and added to the original drum track.

Algorithm

1. Onset Detector (in this case is simply done via gating but could be more sophisticated)
2. Extracting peaks of auto-correlation
3. Bpm and Meter estimation
4. Synthesis of metronome sound

Possible upgrades

The first upgrade I would implement is to calculate the auto-correlation with a sliding time window in order to be able to extract informations while the tune evolves.
Up to now the auto-correlation is evaluated on a a rectangular time window of duration:

$$lag = round(60/lowestBPM * longestMETER,0)$$

which ensure to include the longest bar possible (with longest meter and lowest bpm).

This would need to be the first auto-correlation lag even with a sliding window because at the beginning we of course don't know informations on the signal. After the first calculation we could instead evaluate the auto-correlation on time lags corresponding to the previously evaluated duration of the bar (which is a much smaller time interval) and slide the time window in correspondence of every quarter note.
In this way (and including some tolerance) we could follow slight oscillation of the bpm or detect metric modulation which could occur during the song while keeping the amount of computational cost very low (much lower than a pitch detection based on auto-correlation for instance).

The second upgrade of course would be to make the algorithm functioning "real-time". This would be definitely possible in terms of computational cost but would be problematic because the computations are always at least one bar late (we can search for auto-correlation only with period not smaller than one bar).
This could be overcome with some kind of prediction strategy based on the informations of the last bar available.
Probably some time domain analysis of the transient could be helpful for more robust results.