

Creative Computing II

Audio Features for Music Information Retrieval

23rd February 2010

The fundamental building block for many audio features for content-based Music Information Retrieval is the constant- Q transform. It is possible to approximate a constant- Q transform by combining appropriately combining bins from the linear Fourier Transform. The details of implementing this transform are beyond the scope of the course, but applying it (the subject of this lab) is not.

1. This part is about the direct application of the constant- Q transformation to synthetic, simple signals made up of sinusoids at known frequencies.

- (a) Obtain the provided `constq.m` from the course website, and read the code and comments. Place the code in a location that *Octave* can find it.
- (b) Use the `constq` to construct a matrix appropriate for performing a constant- Q transform for 44.1kHz audio windowed over 0.1s, with 12 bins per octave. You may leave the `loEdge` and `hiEdge` parameters at their default values.

From the source code, the first parameter to `constq` is the window size in samples; the second is the sample rate, and the third is the number of frequency bins per octave. Thus, we generate the constant- Q transform matrix with $Q = \text{constq}(4410, 44100, 12)$;

- (c) Construct a vector representing a sinusoidal signal at 440Hz lasting 0.1s, sampled at a rate of 44.1kHz.

```
x = sin(2*pi*440*[0:4409]/44100);
```

- (d) Using the Fourier Transform, compute the power spectrum of the vector you constructed in part 1c.

```
p = abs(fft(x)).^2
```

- (e) Apply the matrix you constructed in part 1b to the power spectrum to generate the constant- Q spectrum.

```
s = Q*p'
```

- (f) Plot the logarithms of the values of the constant- Q spectrum you have obtained. Check that you understand the location of any particular features in your plot.

With `plot(log(s))`, you should get a plot with a couple of distinguishing features: firstly, a strong peak at an x position of 37; secondly, a couple of ‘missing’ entries at low values on the x -axis. The peak’s location comes from understanding what the constant- Q bins are: the first entry corresponds to a frequency of 55Hz (low A), and there twelve bins per octave, so going up by 12 entries doubles the frequency. The thirteenth entry is then the bin for a frequency 110Hz, the 25th 220Hz and the 37th 440Hz – which is the frequency of our input signal. So the constant- Q transform has successfully captured the frequency content of the input signal.

The ‘missing’ values are easier to explain: in this case, the linear FFT has bins every 10Hz. Near the bottom of the constant- Q spectrum, the bandwidth for each

bin is much smaller than 10Hz, so there will be bins which do not have any associated signal content at all, so the power in them will be 0 (and so the log power will be minus infinity).

- (g) Repeat steps 1c to 1f using different constructed signals. You may wish to try signals at different fundamental frequencies (*e.g.* 415Hz, 220Hz, 110Hz) and signals made by adding together sinusoids at different frequencies.

2. This part is about the application of the constant- Q spectrum to musical signals.

- (a) Obtain a digital music file in `wav` format, ideally with a sample rate of 44.1kHz. You may if you wish use the provided (MIDI-synthesized) `toccatav.wav` audio file, or transcode a Creative-Commons licenced file from <http://www.jamendo.com/>.
- (b) Use `wavread` to load the music file into `Octave`.
- (c) As in part 1 above, apply the constant- Q transformation to the power spectrum of windows of audio data. You should attempt to visualise the output vectors, and verify that the audio feature corresponds to some concept of the musical pitch in the audio. (You may want to refer back to the work with Sonic Visualiser in lab 10.)

*The first thing to do is to convert the audio data read in using `wavread` into windows. One way of doing that is first to convert the audio into a single channel stream of data (say by taking just one channel), then using the `reshape` operator to convert a long vector into a matrix where the columns are successive windows: `windows = reshape(y(:,1)(1:4410*floor(size(y,1)/4410)),4410, '')` The extra complexity in the use of `floor` is to make sure that the data being reshaped fits exactly in an integral number of windows.*

Once that's done, the `fft` operator will do what is required, so to visualize the windowed constant- Q transform over the entirety of the audio:

*`imagesc(log(Q*abs(fft(windows).^2)))`*

In that picture, the x-axis is time measured in windows (so each unit is 0.1s) and the y-axis is constant- Q bin. The first note in the music is an A, so you should be able to see peaks at bins 37, 49 and 61.

3. This part is about the transformation of the constant- Q spectrum into the chromagram and cepstrum features.

- (a) the constant- Q transform can be transformed into a chromagram by adding the values in corresponding bins in each octave together. Implement this for a 12-bin-per-octave constant- Q transform by defining a 12×86 dimensional matrix, which when it multiplies the constant- Q spectrum produces a chromagram, and apply this transformation to the features you computed in part 2 above.

The easy way to do this is with an explicit `for` loop:

`C = zeros(12,86); for j = (1:86) C(1+rem(j,12),j) = 1; end`

Then the chroma view of the same audio from the previous part can be produced with

*`imagesc(log(C*Q*abs(fft(windows).^2)))`.*

- (b) a cepstrum can be produced by taking the Fourier Transform of the constant- Q power spectrum. Again, compute cepstrum values from the constant- Q features you computed in part 2 above, and visualise the magnitude of those feature values. Can you relate high values of the cepstrum feature to any acoustic content?

Other Resources:

- Sonic Annotator: <http://www.omras2.org/SonicAnnotator> and Vamp Plugins <http://vamp-plugins.org/>.
- Brown, J. C. and M. S. Puckette, *An efficient algorithm for the calculation of a constant Q transform*, J. Acoust. Soc. Am., 1992.
- Rhodes, C., *Music Information Retrieval*. In *2910346: Topics in Sound and Music*. Draft available at <http://www.doc.gold.ac.uk/~mas01cr/teaching/cc346/chapter.pdf>.