The Constant Q Transform

[Benjamin Blankertz]

1 Introduction

The constant Q transform as introduced in [Brown, 1991] is very close related to the Fourier transform. Like the Fourier transform a constant Q transform is a bank of filters, but in contrast to the former it has geometrically spaced center frequencies $f_k = f_0 \cdot 2^{\frac{k}{b}}$ ($k = 0, \dots$), where b dictates the number of filters per octave. To make the filter domains adjectant one chooses the bandwidth of the k-th filter as $\Delta_k^{\text{cq}} = f_{k+1} - f_k = f_k(2^{\frac{1}{b}} - 1)$. This yields a constant ratio of frequency to resolution $Q = \frac{f_k}{\Delta_k^{\text{cq}}} = (2^{\frac{1}{b}} - 1)^{-1}$. What makes the constant Q transform so useful is that by an appropriate choice for f_0 (minimal center frequency) and b the center frequencies directly correspond to musical notes. For instance choosing $b = 12$ and f_0 as the frequency of midinote 0 makes the k-th cq-bin correspond the midinote number k.

Another nice feature of the constant Q transform is its increasing time resolution towards higher frequencies. This resembles the situation in our auditory system. It is not only the digital computer that needs more time to perceive the frequency of a low tone but also our auditory sense. This is related to music usually being less agitated in the lower registers.

2 Deriving Filter Parameters for the Constant Q Transform

To derive the calculation of the constant Q transform of some sequence x we begin with an inspection of the familiar formula of a Fourier filter at z

$$(1) \qquad \sum_{n<N} x[n]\, e^{-2\pi i n z/N}$$

Each component of the constant Q transform, in the sequel called cq-bin, will be calculated as such a filter, but suitable values for z and window length N have to be found in order to match the properties discussed above. The bandwidth of the filter (1) is $\Delta_z^{\text{ft}} = f_s/N$ (f_s denotes the sampling rate) independently of z. Thus the desired bandwidth $\Delta_k^{\text{cq}} = f_k/Q$ can be realized by choosing a window of length $N_k = \frac{f_s}{\Delta_k^{\text{cq}}} = Q\frac{f_s}{f_k}$. The frequency to resolution ratio of the filter in (1) is $\frac{f_z}{\Delta_z^{\text{ft}}} = z$. To achieve a constant value Q

for the frequency to resolution ratio of each cq-bin one has to select $z := Q$. Thus for integer values Q the k-th cq-bin is the Q-th DFT-bin with window length $Q\frac{f_s}{f_k}$.

Summerizing we get the following recipe for the calculation of a constant Q transform: First choose minimal frequency f_0 and the number of bins per octave b according to the requirements of the application. The maximal frequency $f_{\max}$ only affects the number of cq-bins to be calculated[1]:

$$(2) \quad K := \lceil b \cdot \log_2(\frac{f_{\max}}{f_0}) \rceil$$

$$(3) \quad Q := (2^{\frac{1}{b}} - 1)^{-1} \qquad \text{and for } k < K \text{ set}$$

$$(4) \quad N_k := \lceil Q\frac{f_s}{f_k} \rceil$$

$$(5) \quad x^{\text{cq}}[k] := \frac{1}{N_k} \sum_{n<N_k} x[n]\ w_{N_k}[n]\ e^{-2\pi i n Q/N_k}$$

To reduce spectral leakage (cf. [Harris, 1978]), it is advisable to use the filter in conjunction with some window function: $\langle w_N[n] \ : \ n < N \rangle$ is some analysis window of length N. We followed Judith Brown by using Hamming windows.

3 Direct Implementation

For comparison first the usual FT-algorithm is given, not in its compact form `ft= (x .* win) * exp(-2*pi*i*(0:N-1)'*(0:N-1)/N)`, but with a loop.

```
function ft= slowFT(x, N)

for k=0:N-1;
   ft(k+1)= x(1:N) * (hamming(N) .* exp( -2*pi*i*k*(0:N-1)'/N)) / N;
end
```

And here is the direct implementation of the constant Q transform.

```
function cq= slowQ(x, minFreq, maxFreq, bins, fs)

Q= 1/(2^(1/bins)-1);
for k=1:ceil(bins*log2(maxFreq/minFreq));
   N= round(Q*fs/(minFreq*2^((k-1)/bins)));
   cq(k)= x(1:N) * (hamming(N) .* exp( -2*pi*i*Q*(0:N-1)'/N)) / N;
end
```

[1] $\lceil x \rceil$ denotes the least interger greater than or equal to x.

4 An Efficient Algorithm

Since the calculation of the constant Q transform according to the formula (5) is very time consuming, an efficient algorithm is highly desirable. Using matrix multiplication the constant Q transform of a row vector x of length N ($N \geq N_k$ for all $k < K$) is

$$(6) \quad x^{\mathrm{cq}} = x \cdot T^*$$

where T^* is the complex conjugate of the temporal kernel[2] $T = (T_{nk})_{n<N,k<K}$

$$(7) \quad T_{nk} := \begin{cases} \frac{1}{N_k}\, w_{N_k}[n]\; e^{2\pi i n Q/N_k} & \text{if } n < N_k \\ 0 & \text{otherwise} \end{cases}$$

Since the temporal kernel is independent of the input signal x one can speed up successive constant Q transforms by precalculating T^*. But this is very memory consuming and since there are many non vanishing elements in T the calculation of the matrix product $x \cdot T^*$ still takes quite a while.
Luckily Judith Brown and Miller Puckette came up with a very clever idea for improving the calculation [Brown and Puckette, 1992]. The idea is to carry out the matrix multiplication in the spectral domain. Since the windowed complex exponentials of the temporal kernel have a DFT that vanishes almost everywhere except for the immediate vicinity of the corresponding frequency the spectral kernel

$$(8) \quad S := \mathrm{DFT}(T) \quad \text{(one dimensional DFTs applied columnwise)}$$

is a sparse matrix (after eliminating components below some threshold value). This fact can be exploited for the calculation of x^{cq} owing to the identity

$$(9) \quad \sum_{n<N} x[n]\; y[n]^* \;=\; \frac{1}{N} \sum_{n<N} x^{\mathrm{ft}}[n]\; y^{\mathrm{ft}}[n]^*$$

where x and y are sequences of length N and x^{ft}, y^{ft} denote their unnormalized discrete Fourier transform. Applying this identity to the formula of constant Q transform (5) using definitions (7) and (8) yields

$$x^{\mathrm{cq}}[k] = \frac{1}{N} \sum_{n<N} x^{\mathrm{ft}}[n]\; S^*_{nk}$$

[2] In (7) we center the filter domains on the left for the ease of notation. Right-centering is more appropriate for real-time applications. Middle-centering has the advantage of making the spectral kernel (8) real.

or equivalently in matrix notation

$$(10) \quad x^{\mathrm{cq}} = \frac{1}{N}\, x^{\mathrm{ft}} \cdot S^*.$$

Due to the sparseness of S the calculation of the product $x^{\mathrm{ft}} \cdot S^*$ involves essentially less multiplications than $x \cdot T^*$.
The Fourier transforms that arise in the efficient algorithm should of course be calculated using FFTs. To this end N is chosen as the lowest power of 2 greater than or equal to N_0 (which is the maximum of all N_k). The calculation of the spectral kernel is quite expensive, but having done this once all succeeding constant Q transforms are performed much faster.

5 Implementation of the Efficient Algorithm

```
function sparKernel= sparseKernel(minFreq, maxFreq, bins, fs, thresh)

if nargin<5 thresh= 0.0054; end  % for Hamming window

Q= 1/(2^(1/bins)-1);                                              (3)
K= ceil( bins * log2(maxFreq/minFreq) );                          (2)
fftLen= 2^nextpow2( ceil(Q*fs/minFreq) );
tempKernel= zeros(fftLen, 1);
sparKernel= [];
for k= K:-1:1;
   len= ceil( Q * fs / (minFreq*2^((k-1)/bins)) );                (4)
   tempKernel(1:len)= ...
      hamming(len)/len .* exp(2*pi*i*Q*(0:len-1)'/len);           (7)
   specKernel= fft(tempKernel);                                   (8)
   specKernel(find(abs(specKernel)<=thresh))= 0;
   sparKernel= sparse([specKernel sparKernel]);
end
sparKernel= conj(sparKernel) / fftLen;                            (10).1

function cq= constQ(x, sparKernel)  % x must be a row vector
cq= fft(x,size(sparKernel,1)) * sparKernel;                       (10).2
```

A matlab implementation of the efficient algorithm for the constant Q transform [Brown and Puckette, 1992] coded by Benjamin Blankertz. The `sparseKernel` has only to be calculated once. This might take some seconds. After that, constant Q transforms of any row vector `x` can be done very

efficiently by calling `constQ`. The function `hamming` is included in the dsp-toolbox. It returns the Hamming window of the given length:

```
hamming(len)= 0.46-0.54*cos(2*pi*(0:len-1)'/len).
```

References

Judith C. Brown, Calculation of a constant Q spectral transform, *J. Acoust. Soc. Am.*, 89(1):425–434, 1991.

Judith C. Brown and Miller S. Puckette, An efficient algorithm for the calculation of a constant Q transform, *J. Acoust. Soc. Am.*, 92(5):2698–2701, 1992.

F. J. Harris, On the Use of Windows for Harmonic Analysis with Discrete Fourier Transform, in: *Proc. IEEE*, Bd. 66, pp. 51–83, 1978.