Audio Engineering Society

Convention e-Brief 567

Presented at the 148th Convention
2020 June 2–5, Online

Computational efficient real-time capable constant-Q spectrum analyzer

Felix Holzmüller[1,2], Paul Bereuter[1,2], Philipp Merz[1,2], Daniel Rudrich[1,3], and Alois Sontacchi[1,3]

[1] *University of Music and Performing Arts, Graz, Austria*
[2] *Graz University of Technology, Austria*
[3] *Institute of Electronic Music and Acoustics, Graz, Austria*

Correspondence should be addressed to Felix Holzmüller (`felix.holzmueller@posteo.net`)

ABSTRACT

The constant-Q transform (CQT) is a valuable tool for music information retrieval, e.g. for chroma calculation and harmonic analysis. In this e-Brief, we propose a block based, real-time capable, efficient analysis algorithm resting upon a subsampling technique performed with fast Fourier transform. In addition, advanced features such as time resolution enhancement towards lower frequencies and a robust CQT-based tuner are presented. Finally a reference-implementation in C++ in form of a VST3-Plugin is introduced. The plugin's source code will be available openly for further development.

1 Introduction

VST-Plugins that show a visual representation of the short-time Fourier transform (STFT) are quite a common tool in modern DAWs. In a musical context, the linear frequency bin spacing is not optimal, since the scale of western music is geometrically spaced. Linear spacing in respect of musical analysis leads to an insufficient resolution in lower frequency regions, while the higher frequencies are presented with an expendable amount of accuracy. In signal processing a general solution to this problem has been found: the constant-Q transform (CQT). So far it has not been implemented for efficient, accurate real-time applications.

Since the CQT was proposed by J.C. Brown in 1991 [1] for musical analysis, numerous authors have contributed to making its calculation more efficient. With the latest additions by G.A. Velasco et al. [2], N. Holighaus et al. [3] and an implementation by C. Schörkhuber et al. [4], it is possible to create a computationally inexpensive, fast Fourier transform (FFT) based CQT-analyzer. Based on this approach, it is possible to develop a real-time capable algorithm and a corresponding implementation as a VST-Plugin using the JUCE framework [5]. Section 2 explains the analyzer's algorithm in detail, by going from its original direct implementation through the various optimization steps. Section 3 features a manual describing the analyzer's user interface. A short summary in section 4 concludes this publication. The program, its source code and a documentation featuring a detailed overview of the framework as well as the plugin's dataflow can be found here: `https://git.iem.at/audioplugins/cqt-analyzer`

2 Algorithm

The CQT can be seen as a filter-bank with logarithmically spaced center frequencies f_k and a constant quality factor Q for all filters. The modulated localization functions (the so called *atoms*) $a_k[n]$ use a window function $g_k[n]$ and a sampling frequency f_s. The samples of a discrete input-signal block $x[m]$ are weighted with the atoms and summed. This defines the CQT as [4, p. 2]

$$X[k,n] = \sum_{m=0}^{N_k-1} x[m]\ a_k^*[m-n], \quad n,k \in \mathbb{N}, \text{ with} \tag{1}$$

$$a_k[m] = g_k[m]\ e^{j2\pi m \frac{f_k}{f_s}}, \qquad m \in \mathbb{Z}. \tag{2}$$

In this case n and k denote time and frequency indices, respectively in the CQT domain. In this notation one can see the CQT's relation to the Discrete Fourier Transformation (DFT): the same kernel function is used, but with logarithmically spaced analysis frequencies f_k, bandwidths B_k[1] and the analysis window length in time domain N_k.

$$f_k = f_0\ 2^{\frac{k}{b}} \tag{3}$$

$$B_k = \frac{f_k}{Q} + \gamma \tag{4}$$

$$Q_{\text{new},k} = \frac{f_k}{B_k} \tag{5}$$

$$b_{\text{new},k} = \frac{\ln(2)}{\operatorname{arsinh}\left(\frac{1}{2\ Q_{\text{new},k}}\right)} \tag{6}$$

$$N_k = Q_{\text{new},k}\ \frac{f_s}{f_k} \tag{7}$$

f_0 is the lowest frequency to be analyzed and b is the resolution in bins per octave. In case of constant-Q, B_k is directly proportional to f_k. With $\gamma > 0$, a frequency-dependent recalculation of $Q_{\text{new},k}$ and $b_{\text{new},k}$ is necessary. The constant-Q case can be achieved by $\gamma = 0$. However, the purpose of γ will be discussed in chapter 2.3.

2.1 Calculating the CQT with a fast convolution

As a consequence of these differences the optimized algorithm used to calculate the DFT (the FFT) is in this case not applicable, making a direct implementation of eq. (1) computationally too expensive or inaccurate for a real-time applications.

[1] In this case B_k is defined as the absolute bandwidth, not as the -3 dB bandwidth.

The FFT is still of use, since the transformation in eq. (1) can be rewritten into a convolution, assuming the atoms $a_k[n]$ are symmetric:

$$\begin{aligned} X[k,n] &= \sum_{n=0}^{N_k-1} x[m]\ a_k^*[m-n] \\ &= \sum_{n=0}^{N_k-1} x[m]\ a_k[n-m] \\ &= (x * a_k)[n] \end{aligned} \tag{8}$$

The convolution can be efficiently computed by a *fast convolution*, that is, using the FFT to compute the convolution by means of a multiplication in the frequency domain and going back to the time domain via an inverse-DFT (IDFT):

$$\begin{aligned} X[k,n] &= (x * a_k)[n] \\ &= \mathscr{F}^{-1}_{i \mapsto n}\{[\mathscr{F}_{n \mapsto i}\{x\} \cdot \mathscr{F}_{n \mapsto i}\{a_k\}](i)\}[n] \end{aligned} \tag{9}$$

where i shall denote a DFT-bin and k a CQT-bin. This requires block processing of the signal. The block lengths correspond to the DFT length N_{DFT}. The length of the DFT N_{DFT} depends on the maximum window length $N_{\max} = \max(N_k)$, occurring at the lowest analysis frequency f_0. The signal is blocked with 50% overlap. Each block is windowed with a *Hann*-window before its transformation into the frequency domain. For a computationally more efficient implementation meaning less overlap, the usage of *Tukey*-windows as proposed in [3] would be also possible. Using oversampling with a factor $os \in \mathbb{N}^*$ its length is defined as

$$\begin{aligned} N_{\text{DFT}} &= os \cdot \text{nextPower}_2\left(N_{max}\right) \\ &= os \cdot \text{nextPower}_2\left(Q_{\text{new}}\ \frac{f_s}{f_0}\right) \\ &= os \cdot 2^{\left\lceil \log_2\left(Q_{\text{new}} \frac{f_s}{f_0}\right)\right\rceil}. \end{aligned} \tag{10}$$

As the algorithm only deals with real valued input signals, we can optimize by only calculating the DFT for positive frequencies.

There are two immediate optimizations for this procedure. Firstly, the input signal needs to be transformed to the frequency domain only once for the calculation of all frequency bins. Secondly, there is no need to repeatedly transform the localization functions, they can be designed and stored in the frequency domain prior to the transformation itself, where they constitute window functions A_k. This gives the advantage of a window with compact support, so that applying the window is computationally easy. As a drawback slight ripples can be observed, due to the window's infinite support in time doamin, after transforming back into the CQT domain. We propose using a *Hann* window due to its good sidelobe suppression and narrow mainlobe as well as perfect overlapping.

$$Y(i) = \mathscr{F}_{n\mapsto i}\{x[n]\}(i) \tag{11}$$

$$X[k,n] = \mathscr{F}^{-1}_{i\mapsto n}\{Y(i)A_k(i)\}[n] \tag{12}$$

2.2 Subsampling in the frequency domain

In this state the algorithm's output, namely one coefficient for each bin in each time step, contains a large amount of redundancy. The Shannon-Nyquist sampling theorem (in its extension to non-baseband signals) states that this redundancy can be removed by subsampling the output of each bin: as long as the sampling rate after subsampling f_s^k is at least the size of the absolute bandwidth B_k, no information will be lost.

$$f_s^k \geq B_k \tag{13}$$

To further reduce the computational effort it is also possible to perform the subsampling in the frequency domain. This is done by applying an IDFT with $N_{\text{IDFT}} < N_{\text{DFT}}$ only along the range where the respective frequency domain window is non-zero, or in other words, by shifting the windowed spectrum to the baseband before transforming back to the time domain with a lower resolution IDFT.

$$i_{u,k} = \left\lfloor \left(f_k \cdot 2^{\frac{1}{b_{\text{new},k}}}\right) \frac{N_{\text{DFT}}}{f_s} \right\rfloor \tag{14}$$

$$i_{l,k} = \left\lceil \left(f_k \cdot 2^{\frac{-1}{b_{\text{new},k}}}\right) \frac{N_{\text{DFT}}}{f_s} \right\rceil \tag{15}$$

$i_{u,k}$ and $i_{l,k}$ are the DFT-bins that mark the upper and lower bounds of A_k. The values of the windows itself are obtained from a large, precalculated Hann window A_{lookup} of length M. For every sampling point of A_{lookup} a frequency $f_{A_{\text{lookup}}}[k,m]$ is assigned for each bin k, based on the CQT's logarithmical frequency spacing. These are calculated as

$$f_{A_{\text{lookup}}}[k,m] = f_k \cdot 2^{\frac{-\lfloor M/2\rfloor + m}{\lfloor M/2\rfloor \cdot b_{\text{new},k}}} \quad \text{for } m = 0,1,\ldots,M-1. \tag{16}$$

A length of $M = 8 \cdot N_{\text{IDFT}} + 1$ (see eq. 19) is more than sufficient for usage without further interpolation. The values of A_{lookup} whose corresponding frequencies $f_{A_{\text{lookup}}}[k,m]$ are closest to the frequencies of the DFT-bins $f_i = i \cdot \frac{N_{\text{DFT}}}{f_s}$ for $i_{l,k} \leq i \leq i_{u,k}$ are chosen and stored as the window function A_k for the kth CQT-bin.

The shift to the baseband is computed with:

$$Y'(i,k) = \begin{cases} Y(i_{l,k}+i)\cdot A_k(i_{l,k}+i), & \text{for} \\ & 0 \leq i \leq i_{u,k} - i_{l,k} \\ 0, & \text{else} \end{cases} \tag{17}$$

The spectrum is then transformed back to obtain X'.

$$X'[k,n_k] = \mathscr{F}^{-1}_{i\mapsto n_k}\{Y'(i,k)\} \tag{18}$$

X' is the subsampled CQT of x, with the time variables n_k of the individual channels progressing with f_s^k, if the IDFT size is chosen to be the minimum, namely

$$N_{\text{IDFT}} = \text{nextPower}_2\left(N_{\text{DFT}}\,\frac{B_{k,\max}}{f_s}\right). \tag{19}$$

In this implementation the maximum required IDFT length (for the lowest frequency bin) is used for all bins, applying zero-padding when necessary. Although slightly less efficient, this has the advantage of all channels running at the same rate. Finally, the time-CQT representation is obtained using an overlap-and-add algorithm on the absolute values $|X'[k,n_k]|$ using

$$hs = 2\,\frac{N_{\text{IDFT}}}{os}. \tag{20}$$

as the hopsize for reconstruction of the blocks.

2.3 Low frequency time resolution enhancement

The in eq. (4) introduced parameter γ enables the modification of a CQT-bin's bandwidth B_k [4, p. 5]. This is done to enhance the low frequency time resolution at the expense of the frequency resolution. Note that the center frequencies of the CQT-bins f_k remain the same.

In the context of human perception this makes sense, since the filterbank of the human auditory system only resembles a constant-Q system above approximately 500 Hz [4, p. 5]. These properties can be explained using the theory of the equivalent rectangular bandwidths (ERB) [6]. Strictly speaking the resulting transformation is not a *constant*-Q transform anymore, when $\gamma \neq 0$.

2.4 Tuner algorithm

The implemented tuner algorithm is only sufficient for a resolution b of at least 36 bins per octave (therefore 3 bins per semitone). The algorithm is based upon parabolic interpolation, where the location of the maximum of the parabola through three samples is determined. Initially, the index k_c of the nearest CQT-bin to the given tuning frequency f_{ref} is found. For parabolic interpolation, three sampling points are needed. Therefore the bins are summed semitone-wise at the relative position of the tuning bin, as well as one bin above and below[2], resulting in an average amplitude distribution over all semitones. This can be calculated as

$$\begin{aligned}
&X_{\text{sum},c}[n_k] = \textstyle\sum_k X'[k,n_k],\\
&\text{for } \left\{k \,\middle|\, (k) \bmod \left(\tfrac{b}{12}\right) = (k_c) \bmod \left(\tfrac{b}{12}\right)\right\}\\
&X_{\text{sum},l}[n_k] = \textstyle\sum_k X'[k,n_k],\\
&\text{for } \left\{k \,\middle|\, (k) \bmod \left(\tfrac{b}{12}\right) = (k_c - 1) \bmod \left(\tfrac{b}{12}\right)\right\}\\
&X_{\text{sum},u}[n_k] = \textstyle\sum_k X'[k,n_k],\\
&\text{for } \left\{k \,\middle|\, (k) \bmod \left(\tfrac{b}{12}\right) = (k_c + 1) \bmod \left(\tfrac{b}{12}\right)\right\}.
\end{aligned} \tag{21}$$

The subscripts l and u describe the bin below and above the center-bin k_c. If the maximum value does not rest upon the reference-bin (e.g. in case of bigger detuning), the center-bin index k_c is appropriately shifted up or down. The parabolic interpolation for unevenly spaced sampling points according to [7] is given as

$$\begin{aligned}
&x_{\max} =\\
&= b + \frac{(f(a)-f(b))(c-b)^2-(f(c)-f(b))(b-a)^2}{2[(f(a)-f(b))(c-b)+(f(c)-f(b))(b-a)]},
\end{aligned} \tag{22}$$

where

$$\begin{aligned}
a &= f_{k_l}, & f(a) &= X_{\text{sum},l}[n_k],\\
b &= f_{k_c}, & f(b) &= X_{\text{sum},c}[n_k],\\
c &= f_{k_u}, & f(c) &= X_{\text{sum},u}[n_k],
\end{aligned} \tag{23}$$

$$f_{\text{tuning}}[n_k] = x_{\max}.$$

[2] e.g. for a resolution of 3 bins per semitone all bins at the center of the semitone, all bins below and all bins above are summed up.

The interpolation is visualized in figure 1.

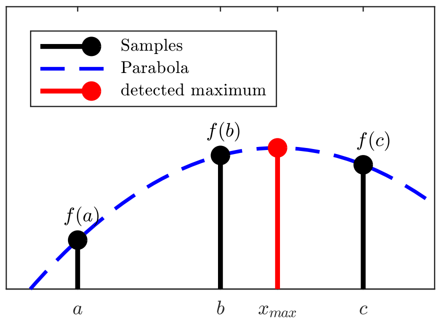

Fig. 1: Parabolic interpolation.

An integration or averaging can smoothen fluctuations of the calculated value. With

$$det[n_k] = 1200 \log_2 \left(\frac{f_{\text{tuning}}[n_k]}{f_{\text{ref}}}\right) \tag{24}$$

the detuning in cent is calculated. If the value exceeds the range $[-50;50]$, 100 cents are either added or subtracted.
Note that the accuracy reduces for increasing γ due to the impaired frequency resolution.

3 VST3-Implementation

The Interface of the VST3-implementation in C++ using the JUCE framework [5] is shown in figure 2. There are in total six controls for varying the CQT parameters as shown in table 1.

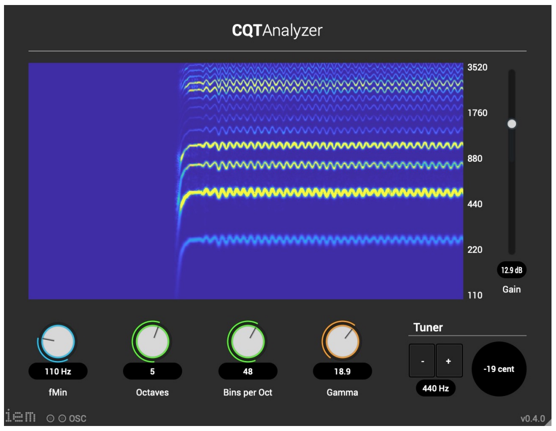

Fig. 2: The RT-CQT VST User Interface

Control Element	Action
fMin	Center Frequency of the lowest bin (cf. f_0)
Octaves	Number of octaves to analyze, starting from fMin, defines maximum frequency
Bins per Octave	Number of CQT-bins per octave, e.g. set to 24 for a quartertone resolution (cf. b)
Gamma	Low frequency time resolution enhancement (cf. B_k and γ)
Tuner	Tuner reference frequency
Gain	Gain for visualization

Table 1: Control parameters of RT-CQT VST

4 Summary

This efficient real-time CQT-Analyzer is based on the algorithm proposed in [3] and [4]. The algorithm can be summarized by the following steps. The time domain signal is transformed into the frequency domain by means of a FFT. Then for each CQT-bin, the spectral components of the DFT between the CQT-bin's lower bound and its upper bound get extracted by means of a frequency-domain window. The extract is then shifted to the baseband, to perform subsampling in the frequency domain by applying an IDFT of a length substantially shorter than the length of the DFT. After applying overlap-and-add, this yields the time series of CQT-coefficients. The whole signal processing chain is block-based. By using the parameter γ, the time resolution towards lower frequencies can be increased. This leads to a decrease of the Q-factor towards lower frequencies.

Furthermore a CQT-based tuner algorithm is proposed. By semitone-wise summing the CQT-coefficient's absolute values below, at and above the tuning frequency bin, fixed through the set tuning frequency, an average amplitude distribution over all semitones is calculated. The three summed values are taken as the representatives at the frequencies below, at and above the tuning frequency bin. These and the aforementioned frequencies are used for the parabolic interpolation shown in eq. (22). With the parabolic interpolation a more accurate estimation of the input-signal's tuning frequency is apprehended. The mistuning of the estimated tuning frequency in respect to the tuning-frequency, set in the Plugin's Tuner section, is calculated and displayed in cents.

For further information on the VST-Plugin's structure and a more detailed insight into the created framework refer to the documentation placed at `https://git.iem.at/audioplugins/cqt-analyzer`.

References

[1] Brown, J. C., "Calculation of a constant Q spectral transform," *Journal of the Acoustical Society of America*, 89, pp. 425 – 434, 1991.

[2] Velasco, G. A., Holighaus, N., Dörfler, M., and Grill, T., "Constructing an invertible constant-Q transform with nonstationary Gabor frames," in *Proc. of the 14th International Conference on Digital Audio Effects*, Paris, France, 2011.

[3] Holighaus, N., Dörfler, M., Velasco, G. A., and Grill, T., "A Framework for Invertible, Real-Time Constant-Q Transforms," *IEEE Transactions on Audio, Speech, and Language Processing*, 21(4), p. 775–785, 2013, doi:10.1109/tasl.2012.2234114.

[4] Schörkhuber, C., Klapuri, A., Holighaus, N., and Dörfler, M., "A Matlab toolbox for efficient perfect reconstruction time-frequency transforms with log-frequency resolution," in *AES 53rd International Conference*, London, UK, 2014.

[5] Storer, J. and ROLI, "JUCE - Jules' Utility Class Extensions," 2019, Version 5.4.5, available at `https://github.com/WeAreROLI/JUCE`.

[6] Glasberg, B. R. and Moore, B. C., "Derivation of auditory filter shapes from notched-noise data," *Hearing Research*, 47(1-2), pp. 103–138, 1990, doi:10.1016/0378-5955(90)90170-t.

[7] Wang, R., "Harvey Mudd College, Lecture Notes, Parabolic Interpolation," 2015, Ruye Wang 2015-02-12, available at `http://fourier.eng.hmc.edu/e176/lectures/NM/node25.html`, last accessed on 26.03.2020.