

Sliding DFT with Kernel Windowing

I. SCOPE

THE sliding discrete Fourier transform (SDFT) is an efficient method for computing the N -point discrete Fourier transform (DFT) of a given signal starting at a given sample from the N -point DFT of the same signal starting at the previous sample [?]. However, the SDFT does not allow the use of a window function, generally incorporated in the computation of the DFT to reduce spectral leakage, as it would break its sliding property. This lecture-notes article will show how windowing can be included in the SDFT by using a kernel derived from the window function, while keeping the process computationally efficient. In addition, this approach allows to turn other transforms, such as the modified discrete cosine transform (MDCT), into efficient sliding versions of themselves.

II. RELEVANCE

The SDFT can be used to perform spectral analysis on successive samples in a signal without having to compute a new DFT from scratch every time, provided that windowing can be incorporated into the computation of the DFTs without harming the efficiency of the method. A notable application of the SDFT with windowing can then be framing detection in audio signals which have undergone lossy compression, in the context of audio compression identification [?]. A lossy compression algorithm will typically introduce traces of compression in the signal being encoded, which can become visible in the time-frequency representation when using the same parameters and framing that were used for the encoding. The parameters and framing can therefore be recovered by computing time-frequency representations at successive samples in the signal and identifying when traces of compression become visible. This demanding process can be translated into an efficient one by using the SDFT with kernel windowing.

III. PREREQUISITES

Basic knowledge of digital signal processing is required to understand this article, in particular, concepts such as the DFT, windowing, and general spectral analysis. More details about the SDFT and lossy audio compression identification can also be found in [?] and [?], respectively.

IV. PROBLEM STATEMENT AND SOLUTION

A. Problem Statement

The SDFT allows to compute the N -point DFT of a signal from the N -point DFT of the same signal starting one sample earlier, in a sense by sliding a rectangular window of length N one sample forward. The SDFT essentially relies on the shift theorem which states that multiplying a signal by a linear phase is equivalent to a circular shift in the corresponding DFT.

Equation 1 shows the derivation of $X_k^{(i)}$, the N -point DFT of signal x starting at sample i , from $X_k^{(i-1)}$, the N -point DFT of x starting at $i-1$, process hence known as SDFT.

$$\begin{aligned}
 X_k^{(i)} &= \sum_{n=0}^{N-1} x_{i+n} e^{-j\frac{2\pi nk}{N}} \\
 &= \sum_{n=0}^{N-1} x_{i+n} e^{-j\frac{2\pi(n+1)k}{N}} e^{j\frac{2\pi k}{N}} \\
 &= \sum_{n=1}^N x_{i-1+n} e^{-j\frac{2\pi nk}{N}} e^{j\frac{2\pi k}{N}} \\
 &= \left(\sum_{n=0}^{N-1} x_{i-1+n} e^{-j\frac{2\pi nk}{N}} - x_{i-1} + x_{i+N-1} \right) e^{j\frac{2\pi k}{N}} \\
 &= \left(X_k^{(i-1)} - x_{i-1} + x_{i+N-1} \right) e^{j\frac{2\pi k}{N}}
 \end{aligned} \tag{1}$$

The SDFT thus only requires $2N$ additions and N multiplications, leading to a linear time complexity of $\mathcal{O}(N)$, while the full and direct computation of the DFT and the fast Fourier transform (FFT) are $\mathcal{O}(N^2)$ and $\mathcal{O}(N \log N)$, respectively.

Transforms such as the DFT typically use a window function in their computation to reduce spectral leakage and enhance spectral analysis. However, the SDFT does not allow the incorporation of a window function as it will break the process shown in Equation 1. One solution would be to perform the windowing in the frequency domain, that is, on the derived DFT through convolution. A practical window function for that matter could be the Hanning window, as the corresponding windowing in the frequency domain equals a simple three-point convolution [?]. Other window functions however may not be as practical, as the corresponding convolutions may involve many more operations, which will ultimately hurt the computational efficiency of the SDFT. The problem is therefore to incorporate any window function into the computation of the DFTs in an efficient manner without breaking the SDFT process.

B. Solution: Kernel Windowing

The idea of performing the windowing in the frequency domain can still be exploited, by reformulating the convolution as a multiplication by a kernel which can be derived from any window function. Such a kernel will be independent from the signal to be processed and only need to be computed once. It will typically have a very small number of values which would be significant, which means that most of the values can then be ignored. This would lead to a very sparse kernel which can then be applied to the DFT of the signal, producing results virtually equivalent to the DFT of the same signal modified by the corresponding window function, while preserving the computational efficiency of the SDFT.

The constant-Q transform (CQT) is a transform with a logarithmic frequency resolution which was proposed as a more adapted alternative to the Fourier transform for analyzing music signals [?]. A fast algorithm was soon after proposed which translated the slow computation of the CQT into the multiplication of a DFT and a kernel, the former one which can be efficiently computed using the FFT and the latter one which is computed once beforehand and is typically very sparse [?]. The idea was to use Parseval's theorem to turn the direct computation in the time domain into a multiplication between a DFT and a kernel in the frequency domain, essentially demonstrating the property of energy conservation between the time and the frequency domains [?].

Equation 2 recalls Parseval's theorem. X is the N -point DFT of x and \bar{x} represents the complex conjugate of x .

$$\sum_{n=0}^{N-1} x_n \bar{y}_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \bar{Y}_k \quad (2)$$

Following a similar idea, we propose to use Parseval's theorem to translate the DFT of a windowed signal into the DFT of the signal, multiplied by a kernel which is derived from the corresponding window function, a multiplication which will happen after the SDFT process. Unlike in the fast CQT case, the purpose here is not to speed up the computation of the transform by taking advantage of the efficiency of the FFT algorithm in conjunction with the use of sparse kernel, but to extract the windowing operation from the DFT computation, so that the SDFT process shown in Equation 1 still holds.

Equation 3 shows the computation of $\mathcal{X}^{(i)}$, the N -point DFT of the signal x starting at sample i and modified by the window function w , from $X^{(i-1)}$, the N -point DFT of x starting at $i-1$ (without windowing), multiplied by the kernel K which is derived from w .

$$\begin{aligned} \mathcal{X}_k^{(i)} &= \sum_{n=0}^{N-1} x_{i+n} \underbrace{w_n e^{-\frac{j2\pi nk}{N}}}_{\bar{y}_n} \\ &= \sum_{k'=0}^{N-1} X_{k'}^{(i)} \underbrace{K_{k,k'}}_{\frac{1}{N} \bar{Y}_{k'}} \\ &= \sum_{k'=0}^{N-1} \left[\left(X_{k'}^{(i-1)} - x_{i-1} + x_{i+N-1} \right) e^{\frac{j2\pi k'}{N}} \right] K_{k,k'} \\ K_{k,k'} &= \frac{1}{N} \bar{Y}_{k'} = \frac{1}{N} \sum_{n=0}^{N-1} y_n e^{-\frac{j2\pi nk'}{N}} \\ &= \frac{1}{N} \sum_{n=0}^{N-1} w_n e^{-\frac{j2\pi nk}{N}} e^{-\frac{j2\pi nk'}{N}} \\ &= \frac{1}{N} \sum_{n=0}^{N-1} w_n e^{\frac{j2\pi n(k'-k)}{N}} \end{aligned} \quad (3)$$

As we can see in Equation 3, the kernel is completely independent from the signal, therefore it only needs to be computed once, before the SDFT process. Furthermore, given

the nature of such kernel, typically only a very small number of its values will be significant, which means that most of the values can then be zeroed (given some threshold) leading to a very sparse kernel. The multiplication of the derived DFT by such kernel will thus only involve few more operations, keeping the whole process computationally efficient.



Fig. 1. Kernels derived from the Hanning (top-left), Blackman (top-right), triangular (center-left), Parzen (center-right), Gaussian (with $\alpha = 2.5$) (bottom-left), and Kaiser (with $\beta = 0.5$) (bottom-right) windows. The kernels were derived for an N -point DFT where $N = 2048$ samples. Only the first 100 coefficients at the bottom-left corner of the N -by- N kernels are shown. The values are displayed in log of amplitude.

Figure 1 shows the kernels derived from some common window functions, namely Hanning, Blackman, triangular, Gaussian, Parzen, and Kaiser windows. As we can see, the Hanning window kernel shows only three non-zero values per row, confirming that the corresponding windowing in the frequency domain equals a simple three-point convolution, while the Blackman window kernel shows five non-zero values per row. Both those windows are actually special cases of the generalized cosine window whose corresponding windowing in the frequency domain equals convolutions with typically only few points. Unlike the Hanning and Blackman window kernels, the triangular, Parzen, Gaussian, and Kaiser window kernels show additional non-zero values around their main diagonal, suggesting that the corresponding windowings in the frequency domain equal convolutions with many more points. However, most of those non-zero values have very small magnitudes ($\ll 0.01$) and could then be ignored without significantly affecting the actual windowing process. By using an appropriate threshold, those kernels can therefore be made very sparse with only a few meaningful values per row, in the same manner as in the fast CQT case [?].

As proposed in [?], we computed for each of those kernels the error in keeping the values greater than a chosen threshold, by dividing the sum of the magnitudes of the values after

thresholding by the sum of the magnitudes of all the values before thresholding. A threshold of 0.01 will thus give very small errors of 0.049, 0.009, 0.020, and 0.015, for the triangular, Parzen, Gaussian, and Kaiser window kernels, respectively, when derived for an N -point DFT with $N = 2048$. With such threshold, the first three kernels will then only have five non-zero values per row, while the latter one will have three non-zero values per row. This shows that only a very small number of values are actually significant in such kernels. The multiplication of the DFT by those very sparse kernels will then only involve KN multiplications and KN additions, with $K = 3$ or 5 , barely affecting the computational efficiency of the SDFT, still maintaining a linear complexity of $\mathcal{O}(N)$, and producing results virtually equivalent to taking the DFT of the signal modified by the corresponding window functions.

V. COMPUTATIONAL EXAMPLES

A. Framing Detection and Lossy Audio Coding

The SDFT with kernel windowing can be particularly useful for fast framing detection, in the context of audio compression identification. Audio compression identification is the recovery of information regarding the data compression that an audio signal has undergone. In particular, the recovery of the parameters and framing used at the time-frequency decomposition stage of the encoding could allow to identify the coding format or detect alterations in audio signals which have undergone lossy compression [?], [?], [?], [?], [?]. Lossy compression algorithms typically introduce traces of compression in the audio signal being encoded, in the form of time-frequency coefficients quantized to zeros which can become visible when using the same parameters and framing that were used for the encoding. One approach to identify if and when lossy compression was used would then be to compute the time-frequency representation at successive samples in the audio signal and search for traces of compression every time, given a set of parameters associated with a known coding format, such as time-frequency transform, window length, and window function, a process also known as framing detection.

Lossy audio coding formats, perhaps the most popular ones being MP3, AAC, AC-3, Vorbis, and WMA, are widely used for storage, e.g., in music and video files, or transmission, e.g., in radio and television broadcasting. Compression algorithms which can encode to such formats first transform the audio signal into a time-frequency representation, derive a psychoacoustic model to locate regions of perceptually less significance, then quantize the data given the psychoacoustic model, and finally convert it into a bitstream. The transform used at the time-frequency decomposition stage is typically based on the MDCT, and a variety of window lengths and window functions can be used depending on the coding format. In particular, specialized window functions such as the sine, slope and Kaiser-Bessel-derived (KBD) windows are generally required for the MDCT to be invertible. For more information about lossy audio coding, the reader is referred to [?].

Equation 4 shows the computation of the MDCT (without windowing).

$$Y_k = \sum_{n=0}^{N-1} x_n \cos \left[\frac{2\pi}{N} \left(n + \frac{1}{2} + \frac{N}{4} \right) \left(k + \frac{1}{2} \right) \right] \quad (4)$$

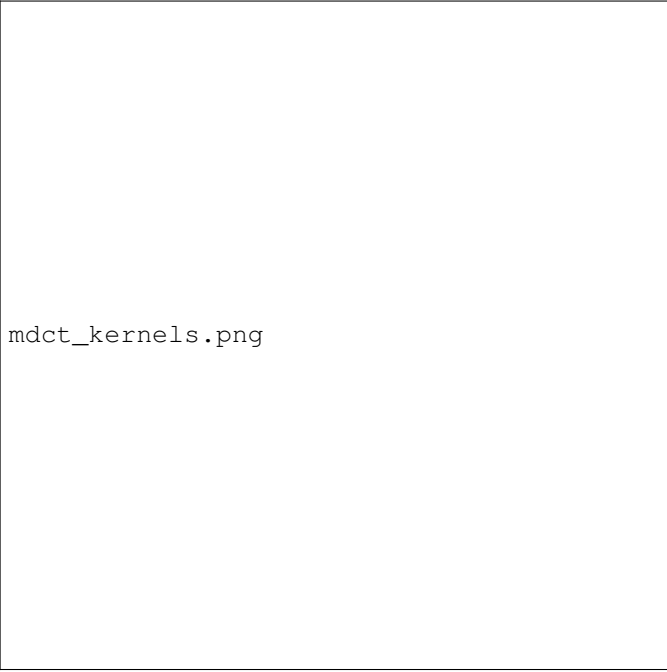
B. Sliding MDCT with kernel windowing

In this context, performing framing detection for audio compression identification would involve computing an MDCT for every set of window length and window function associated with a known lossy coding format, at successive samples in the audio signal, and search for time-frequency coefficients quantized to zero, until one of the set shows visible traces of compression for a specific framing of the signal. We can see that such process will be computationally demanding, as a full transform would have to be computed every time. The direct computation of the MDCT, including the windowing using one of the specialized window functions presented earlier, can actually be translated into an SDFT with kernel windowing, by incorporating the computation of the window function, and a part of the MDCT, into a kernel which will still happen to be very sparse, thus making the process computationally efficient.

Equation 5 shows the computation of $\mathcal{Y}^{(i)}$, the N -point MDCT of signal x starting at sample i and modified by the window function w , from $X^{(i-1)}$, the N -point DFT of x starting at $i-1$ (without windowing), multiplied by the kernel K which is derived from w .

$$\begin{aligned} \mathcal{Y}_k^{(i)} &= \sum_{n=0}^{N-1} x_{i+n} \underbrace{w_n \cos \left[\frac{2\pi}{N} \left(n + \frac{1}{2} + \frac{N}{4} \right) \left(k + \frac{1}{2} \right) \right]}_{\bar{y}_n} \\ &= \sum_{k'=0}^{N-1} X_{k'}^{(i)} \underbrace{K_{k,k'}}_{\frac{1}{N} \bar{Y}_{k'}} \\ &= \sum_{k'=0}^{N-1} \left[\left(X_{k'}^{(i-1)} - x_{i-1} + x_{i+N-1} \right) e^{\frac{j2\pi k'}{N}} \right] K_{k,k'} \\ K_{k,k'} &= \frac{1}{N} \bar{Y}_{k'} = \frac{1}{N} \sum_{n=0}^{N-1} \overline{y_n} e^{-\frac{j2\pi n k'}{N}} \\ &= \frac{1}{N} \sum_{n=0}^{N-1} w_n \cos \left[\frac{2\pi}{N} \left(n + \frac{1}{2} + \frac{N}{4} \right) \left(k + \frac{1}{2} \right) \right] e^{-\frac{j2\pi n k'}{N}} \\ &= \frac{1}{N} \sum_{n=0}^{N-1} w_n \cos \left[\frac{2\pi}{N} \left(n + \frac{1}{2} + \frac{N}{4} \right) \left(k + \frac{1}{2} \right) \right] e^{\frac{j2\pi n k'}{N}} \end{aligned} \quad (5)$$

Figure 2 shows the kernels derived for an N -point MDCT, from the sine window where $N = 1152$ samples, as in MP3, from the slope window where $N = 2048$ samples, as in Vorbis, and from the KBD window where $N = 512$ samples, as in AC-3. As we can see, most of the values in those kernels appear to have negligible magnitudes, while the very few values with significant magnitudes appear to be concentrated around two diagonals, one going from the bottom-left to the top-center and one going from the top-center to the bottom-right. As in [?], we computed for each of those kernels the error in



mdct_kernels.png

Fig. 2. Kernels derived for an N -point MDCT, from the sine window where $N = 1152$ samples (top), the slope window where $N = 2048$ samples (center), and the KBD window where $N = 512$ samples (bottom). The values are displayed in log of amplitude.

keeping the values greater than 0.01 and obtained very small errors of 0.000, 0.022, and 0.013, for the sine, slope, and KBD window kernel, respectively. With such threshold, the sine kernel will only have around two non-zero values per row, and the slope and KBD kernels around six non-zero values per row. These very sparse kernels will therefore barely affect the computational efficiency of the SDFT, while still producing results equivalent to taking the MDCT of the signal modified by the corresponding window functions.

VI. WHAT WE HAVE LEARNED

We have shown that the SDFT can incorporate windowing in its computation by using a kernel which can be derived from any window function and which can be made very sparse. This SDFT with kernel windowing will produce results equivalent to the DFT of the signal modified by the corresponding window function, while keeping the process computationally efficient. This approach can find application in audio compression identification, in particular by making the process of framing detection much more efficient, allowing to translate a transform such as the MDCT into an efficient sliding version of itself.

VII. AUTHOR

Zafar Rafii (zafar.rafi@nielsen.com) is a senior research engineer at Gracenote, with a PhD in Electrical Engineering and Computer Science from Northwestern University.