SoftwareX 8 (2018) 5–8

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Short-Time Fourier Transform with the Window Size Fixed in the Frequency Domain (STFT-FD): Implementation

Carlos Mateo [a,*], Juan Antonio Talavera [b]

[a] Institute for Research in Technology, School of Engineering (ICAI), Universidad Pontificia Comillas, C/Santa Cruz de Marcenado, 26, 28015 Madrid, Spain
[b] School of Engineering (ICAI), Universidad Pontificia Comillas, C/Alberto Aguilera, 23, 28015 Madrid, Spain

ARTICLE INFO

Article history:
Received 15 November 2017
Accepted 15 November 2017

Keywords:
Short Time Fourier Transform
Time–frequency domain
Window Size
Multi-resolution
Wavelet
Electrocardiogram

ABSTRACT

The Short-Time Fourier Transform (STFT) is widely used to convert signals from the time domain into a time–frequency representation. This representation has well known limitations regarding time–frequency resolution. In this paper, we present a set of MATLAB functions to compute a transform, which uses the basic concept of the Short-Time Fourier Transform, but fixes the window size in the frequency domain instead of in the time domain. This approach is simpler than similar existing methods, such as adaptive STFT or multi-resolution STFT, and in particular it requires neither the filters of multi-resolution techniques, nor the evaluation of the local signal characteristics of adaptive techniques. An illustrative example is presented and compared with the Morlet wavelet transform.

Code metadata

Current code version	V1.4
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-17-00087
Code Ocean compute capsule	https://codeocean.com/2017/11/25/short-time-fourier-transform-with-the-window-size-fixed-in-the-frequency-domain-lpar-stft-fd-rpar-colon-code/codedomain-lpar-stft-fd-rpar-colon-code/code
Legal Code License	GNU General Public License, version 3.0 (GPL-3.0)
Code versioning system used	None
Software code languages, tools, and services used	MATLAB R1012a or higher
Compilation requirements, operating environments & dependencies	Communication toolbox is optionally required to add Gaussian noise to the synthetic signal. File ecg.txt containing an electrocardiogram signal has to be downloaded from http://eleceng.dit.ie/dorran/matlab/ecg.txt (original source: https://physionet.org) in order to test that case study. In any case, there are additional self-generating test case studies that can be directly analyzed using this software.
If available Link to developer documentation/manual	https://github.com/ElsevierSoftwareX/SOFTX-D-17-00087/blob/master/STFT_FD_Documentation_v1_4.pdf
Support email for questions	cmateo@comillas.edu

1. Motivation and significance

The Short-Time Fourier Transform (STFT) can be used to convert signals whose frequency content changes over time into the time–frequency domain although it may introduce some redundant information [1,2]. However, this representation has well known limitations regarding time–frequency resolution mainly due to using a fixed window size.

Many alternatives and paradigms have been proposed. For example, wavelets use a mother wavelet function which can be scaled through frequency and shifted in time [3]. The paper contrasts the results obtained using our proposed approach with those derived from a wavelet method.

DOI of original article: http://dx.doi.org/10.1016/j.dsp.2017.11.003.
* Corresponding author.
E-mail address: cmateo@comillas.edu (C. Mateo).

https://doi.org/10.1016/j.softx.2017.11.005

Table 1
STFT_FD1 implementation, which relies on use of the FFT algorithm to calculate the DFT according to Eq. (6) in [8].

```
function [v_time,v_frec,stft_fd,elapsed_time]=STFT_FD1(v_t,x,Ts,NC)
% Short Time Fourier Transform with the Window Size Fixed in the Frequency Domain
% Solution based on the FFT
%
    initial_time=cputime;
    NS=size(x,2);
    v_time = v_t(1+(2*NC/2):NS-(2*NC)/2);
    v_frec = [];
    v_p = [];
    stft_fd=[];
    for p=2:NS/NC
        v_frec = [v_frec, 1/(p*Ts)];
        v_p=[v_p,p];
        NW=p*NC;
        window=hamming(NW)';
        for t=NW/2:NS-NW/2
            wx=window(1:NW).*x(t-NW/2+1:t+NW/2);
            fft_wx=fft(wx);
            stft_fd(t,p)=fft_wx(1+NC);
        end;
    end;
    elapsed_time=cputime-initial_time;
```

Other techniques revisit the basic Fourier Transform and propose new alternatives to the standard Short-Time Fourier (STFT), such as adaptive STFT [4,5] and multi-resolution STFT [6,7]. The idea of multi-resolution STFT is based on using different window sizes for different frequencies. Multi-resolution STFT uses band-pass filters to divide the signal into frequency bands. Therefore, multi-resolution STFT relies on the use of filters to obtain the representation. Adaptive STFT adjusts the window size for each time instant depending on the local signal characteristics.

The software presented computes a transform using the basic concept of STFT, but fixing the window size in the frequency domain (STFT-FD). Our approach is similar to multi-resolution STFT and wavelets, but the methodology is different. A key element is the definition of the window size as the number of cycles of the frequency component being considered rather than in seconds. In this way, it is possible to define the transform with an equation, requiring neither the filters of multi-resolution STFT, nor the evaluation of local characteristics of adaptive techniques. Further methodological discussions can be found in the complementary paper of Digital Signal Processing issue [8].

2. Software description

2.1. Software introduction

The software architecture consists of the following functions. The main function is *test_all_examples*, which analyzes a synthetic signal (generated in the *synthetic_signal* function), a two linear chirp signal (generated in the *chirp2_signal* function) and an electrocardiogram (ECG) signal (loaded in the *ecg_signal* function). The three signals are presented and analyzed in [8]. The signals can be analyzed with the proposed STFT-FD transform, using the *analyze_stft_fd* function. The transform is computed using two variant implementations, *STFT_FD1*, which is based on the use of the Fast Fourier Transform, and *STFT_FD2*, which directly computes the equation of the proposed transform, as explained in [8]. The signals are represented in the time domain using the *plot_signal* function, and in the time–frequency domain using the *plot_stft_fd* function. The computation time required to compute the transform with the two approaches is also displayed.

The first approach, (*STFT_FD1*), relies on the definition of the transform, based on the Discrete Fourier Transform (DFT) according to Eq. (6) in [8], and computed using the Fast Fourier Transform (FFT) algorithm. In standard STFT, the window size is fixed in seconds. Therefore, in that case, all the STFTs can be computed using the same FFT for a given time instant. The consequence of the window size being frequency-dependent is that we have to compute each frequency component from a different FFT. We take the (1+NC) component of the windowed signal for the STFT-FD, where NC is the number of cycles inside the window function.

The second way to compute the transform, (*STFT_FD2*), is obtained after expanding that equation, and obtaining Eq. (7) in [8]. This equation can be directly used to compute the transform and is implemented in function *STFT_FD2*.

In our implementation we have selected Hamming windows, although the software could very easily be modified to use other types of window functions. In the same way that we have to select a mother wavelet in wavelets, with the proposed method we have to select a window function (e.g. Hamming) and parameter NC.

Finally, the transform is converted into the time–frequency domain using Eq. (2) in [8].

2.2. Software implementation

The code for *STFT_FD1* is summarized in Table 1. In the complete version of the code that has been uploaded (see Code metadata table) there are additional comments explaining in detail each of the steps and the interface of the functions. Variable *v_t* is the time vector, the signal is defined in variable *x*, *Ts* is the sampling period and NC is the number of cycles inside the window function. The vectors of time, frequency and number of samples per cycle are computed in *v_time*, *v_frec* and *v_p*. The number of samples of the window (NW) is defined according to Eq. (3) in [8]. A Hamming window is calculated using the *Hamming* function and the number of samples of the window. The input signal after applying the window function is *wx*, which is obtained by multiplying the corresponding interval of the input signal (*x*) by the Hamming window (*window*). Finally the *fft* of the signal is computed, and the value of the (NC+1) term of the FFT transform is stored in the resulting *stft_fd*. A parallel *for* can be used in the inner loop, but for the signal tested it does not improve computation time.

The code for *STFT_FD2* is summarized in Table 2. Again, the vectors of time, frequency and number of samples per cycle are computed in *v_time*, *v_frec* and *v_p*, and the number of samples of the window (NW), the Hamming window and the windowed

Table 2
STFT_FD2 implementation, according to Eq. (7) in [8].

```
function [v_time,v_frec,stft_fd,elapsed_time]=STFT_FD2(v_t,x,Ts,NC)
% Short Time Fourier Transform with the Window Size Fixed in the Frequency Domain
% Solution based on the STFT-FD formula

    initial_time=cputime;
    NS=size(x,2);
    v_time = v_t(1+(2*NC/2):NS-(2*NC)/2);
    v_frec = [];
    v_p = [];
    stft_fd=[];
    for p=2:NS/NC
        v_frec = [v_frec, 1/(p*Ts)];
        v_p=[v_p,p];
        NW=p*NC;
        window=hamming(NW)';
        for t=NW/2:NS-NW/2
            wx=window(1:NW).*x(t-NW/2+1:t+NW/2);
            n=1:NW;
            stft_fd(t,p) = sum (wx.* exp(-j*2*pi*NC*(n-1)/NW));
        end;
    end;
    elapsed_time=cputime-initial_time;
```

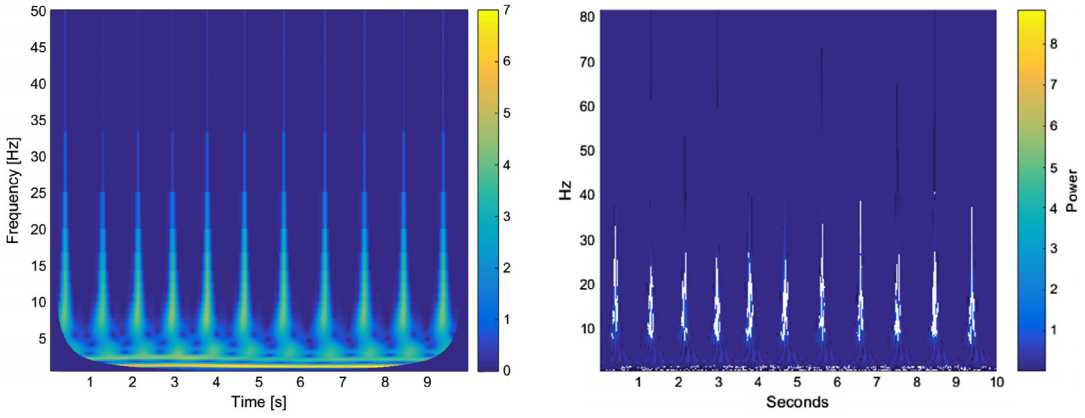

Fig. 1. Representation of the ECG signal with (a) the proposed STFT-FD in the time–frequency domain, NC = 4, (on the left), and (b) Morlet wavelet transform (on the right).

signal (*wx*) are evaluated. However, in this case, the coefficients of the STFT-FD are not computed using the FFT function, but directly using Eq. (7) in [8].

3. Illustrative examples

We illustrate the transform with an ECG signal.[1] The signal in the time domain is represented in [8]. The proposed transform of the signal using the *test_all_examples* function (computed using *STFT_FD1* or *STFT_FD2*) is illustrated in Fig. 1a, where the high frequency and low frequency components of the signal are clearly observed in the time–frequency domain. This representation can be contrasted with the Morlet wavelet transform represented in Fig. 1b, where all the frequency components are also correctly detected.

When compared, the two approaches (*STFT_FD1* and *STFT_FD2*) produce identical results, but for the proposed signal, *STFT_FD1* requires 1.4 s while *STFT_FD2* requires 3.9 s. This is an unexpected result, because with *STFT_FD1* we calculate all the coefficients of the FFT instead of just the terms required. However, the optimizations in the calculation of the FFT make this approach more efficient. Nevertheless, in the same way that *STFT_FD1* is optimized by using FFT, *STFT_FD2* is still of interest as it does not rely on the use of FFT, and could be further optimized in future research. The proposed transform is contrasted with other methods in the complementary paper [8], using a synthetic signal, a two linear chirp signal, a clean ECG signal and a noisy ECG signal.

4. Impact

This paper proposes a software package to compute the Short-Time Fourier Transform with the Window Size Fixed in the Frequency Domain (STFT-FD). The approach is novel although has certain similarities with existing techniques such as multi-resolution or wavelets. However, its different conceptual base provides certain important advantages such it does not requires the band-pass filter of multi-resolution techniques nor the evaluation of local characteristics of adaptive techniques. Besides, the generality and the inherent simplicity of the proposed methodology make it particularly suited for being integrated into a broad range of signal analysis applications.

There is, however, a pending issue related to this transform in terms of computing efficiency for long signals. In its present form, the method is not computationally efficient. In fact, the main problem of the transform is that the matrix to be computed can be

[1] The signal corresponds to the initial 10 s of an ECG signal downloaded from http://eleceng.dit.ie/dorran/matlab/ecg.txt (original source: https://physionet.org), with a sample frequency of 100 Hz.

huge in the case of very long signals. Optimizations should, therefore, probably focus on reducing the redundancy of the information of the transform; particularly the number of coefficients that are calculated.

5. Conclusions

The software proposed in this paper has similar characteristics to existing techniques such as multi-resolution STFT and wavelets, but the methodology is essentially different. The implementation of this approach is very straightforward increasing the value of the proposed method, and facilitating reproducibility and reuse by other authors.

Two alternative functions have been proposed to compute the transform (*STFT_FD1* and *STFT_FD2*). The results are the same using both functions. However, the required computing time is lower with *STFT_FD1* implementation, which is based on an evaluation of the FFT. Nevertheless, *STFT_FD2* relies on a simple expression to compute the transform, and in this case, therefore, there might be still more margin to optimize the function. Some inefficiencies of the proposed approach still have to be solved, so that it can be successfully applied on longer signals. Thus, our future lines of research are aiming to reduce the redundancy of the information of the transform and to diminish the number of coefficients computed.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2017.11.005.

References

[1] Zhang WY, Hao T, Chang Y, Zhao YH. Time-frequency analysis of enhanced GPR detection of RF tagged buried plastic pipes. NDTE Int 2017;92:88–96. http://dx.doi.org/10.1016/j.ndteint.2017.07.013.
[2] Liu H, Li L, Ma J. Rolling bearing fault diagnosis based on STFT-deep learning and sound signals. Shock Vib 2016;2016.
[3] Daubechies I. The wavelet transform time-frequency localization and signal analysis. IEEE Trans Inform Theory 1990;36:961–1005.
[4] Xie H, Lin J, Lei Y, Liao Y. Fast-varying AM–FM components extraction based on an adaptive STFT. Digit Signal Process 2012;22:664–70.
[5] Pei SC, Huang SG. STFT with adaptive window width based on the chirp rate. IEEE Trans Signal Process 2012;60:4065–80.
[6] Gnann V, Becker J. Signal reconstruction from multiresolution STFT magnitudes with mutual initialization. In: 45th int. conf. appl. time-frequency process. audio, 2012 pp. 2–3.
[7] Pihlajamäki T. Multi-resolution short-time Fourier transform implementation of directional audio coding. Helsinki University of Technology; 2009.
[8] Mateo C, Talavera JA. Short time Fourier transform with the window size fixed in the frequency domain (this issue). Digit Signal Process 2017.