Published in Image Processing On Line on YYYY–MM–DD.
ISSN 2105–1232
This article is available online with supplementary materials, software, datasets and online demo at
http://www.ipol.im/pub/algo/gjmr_line_segment_detector/

PREPRINT September 18, 2013

STFT time-frequency visualization Application to sound signals

Eva Wesfreid

CMLA, ENS Cachan, France
(eva@cmla.ens-cachan.fr)

Abstract

The short time Fourier transform (STFT) is the most widely used algorithm for analyzing non-stationary signals. This paper describes two different STFT methods and their time-frequency visualization with application to sound signals.

An ANSI C implementation is provided and available from the web page of this article. The code is distributed under GPL license.

1 Introduction

STFT [1] [2] is a powerful algorithm to show a signal time-frequency behavior. Two different STFT methods are described in this paper, the first one used by Guoshen et al. [3] for audio signals denoising and the other one used by Gabriel Peyré [4] for signal processing. Non-stationary signals are analyzed locally using window functions on overlapping time-intervals. The STFT algorithm computes the Fourier transform of windowed signals (signals multiplied by overlapping window functions). The windowed signal is written in a real matrix of p columns times w rows where p is the number of windows, and w is the window length. The signal STFT is the Fourier transform of the windowed signal computed column by column. Therefore the STFT coefficients are written in a complex matrix of p columns times w rows. This matrix is then transformed into a color matrix and finally into a time-frequency image, the spectrogram. This algorithm is applied to audio signals, some examples of sound spectrograms with different choices of window length are shown on-line.

2 STFT algorithm

Windowed signal

A non stationary signal Fourier transform is analyzed locally using translated window functions. The Hann window (commonly called *Hanning Window*) of length w

$$g[n] = \begin{cases} \frac{1}{2}\left(1 - \cos\left(\frac{2\pi n}{w-1}\right)\right) & \text{for} \quad 0 \le n < w \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

verifies

$$g[0] = g[w-1] = 0$$

and

$$g[q] = 1$$

if w is odd and $q = \frac{w-1}{2}$ (half window length). This is a smooth window function in the sense that

$$h[t] = \begin{cases} \frac{1}{2}\left(1 - \cos\left(\frac{2\pi t}{w-1}\right)\right) & \text{for} \quad 0 \leq t < w \\ 0 & \text{otherwise} \end{cases}$$

is a $C^1(\mathbb{R})$ function.
The p shifted window functions

$$g_j[n] = g[n - jq] \tag{2}$$

with $0 \leq j \leq p-1$ realize a partition of unity

$$\sum_{j=0}^{p-1} g_j[n] = 1$$

for $q \leq n \leq pq$. This partition of unity is not verified for even w length windows.
Figure 1 shows this partition for $p = 4$, $w = 257$, $q = 128$ (g_0, g_1, g_2, g_3). This figure displays the window $g_0 = g$ defined in (1) over $[0, w[$ with three other shifted windows g_1, g_2, g_3 (2).

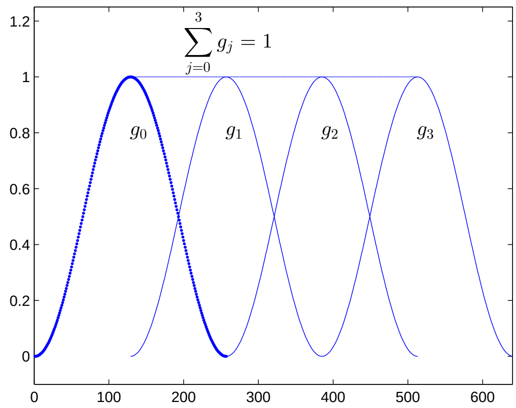

Figure 1: Hann overlapping windows

If f is a signal of length $N = (p+1)q$ defined over $[0, N[$ then f is windowed

$$f = \sum_{j=0}^{p-1} g_j f$$

over $[q, pq]$. If f is a real column array then the windowed signal $(g_j f)_{0 \leq j < p}$ is a real matrix of p columns and w rows (w is the window length)

$$(g_j[n] f[n])_{\ 0 \leq n < w,\ 0 \leq j < p}$$

Furthermore, the signal can be extended to $[-q, N+q]$ and windowed with $p+2$ window functions

$$f = \sum_{j=-1}^{p} g_j f$$

$$\sum_{j=-1}^{p} g_j[n] = 1$$

over $[0, (p+1)q]$.

Figure 2 shows $\sum_{j=-1}^{p} g_j[n] = 1$ over $[0, 5q]$.

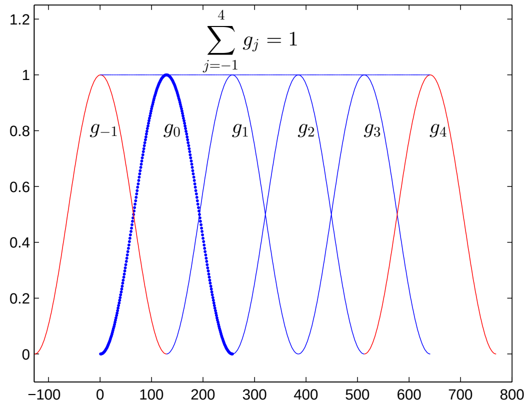

Figure 2: Hann overlapping windows (including boundary windows)

Forward transform

The STFT computes the Fourier transform of a windowed signal, column by column

$$f \longrightarrow \{g_j f\}_{0\le j<p} \longrightarrow \{\widehat{g_j f}\}_{0\le j<p}$$

where $\widehat{g_j f} = (\widehat{g_j f}[k])_{0\le k<w}$ for each window index (time index) j (k is the frequency index),

$$S_f[k,j] = \widehat{g_j f}[k] = \sum_{n=0}^{w-1} f[n+jq]g[n]\exp\left(-\frac{2i\pi nk}{w}\right)$$

for $0 \le k < w$ and $0 \le j < p$.

Therefore, the complex matrix S_f of p columns and w rows is the STFT of a signal f of length $N = (p+1)q$ defined over $[0, N[$ (not extended to $[-q, N+q]$).

If $g_{k,j}[n] = g[n-jq]\exp(\frac{2i\pi[n-jq]k}{w})$ and $< .,. >$ denotes the scalar product in $\mathbb{C}^w$ then

$$S_f[k,j] = < f, g_{k,j} >$$

Backward transform with Hann window function

The backward STFT computes first the inverse Fourier transform of each column $S_f[.,j] = \widehat{g_j f}$

$$\{g_j f\}_{0\le j<p} \longleftarrow \{\widehat{g_j f}\}_{0\le j<p}$$

then f is reconstructed using the partition of unity over $[q, pq]$ (for Hann window functions)

$$f = \sum_{j=0}^{p-1} g_j f \longleftarrow \{g_j f\}_{0\le j<p} \longleftarrow \{\widehat{g_j f}\}_{0\le j<p}$$

and

$$f[n] = \frac{1}{w}\sum_{j=0}^{p-1}\sum_{k=0}^{w-1}\widehat{g_j f}[k]\exp(\frac{2i\pi nk}{w}) = \frac{1}{w}\sum_{j=0}^{p-1}\sum_{k=0}^{w-1}S_f[k,j]\exp(\frac{2i\pi nk}{w})$$

over $[q, pq]$.
This backward transform is used in [3] and [6] for audio signal denoising.

STFT with renormalized window function

If $V = (V[n])_{n\in Z}$ is a window function such that $V[n] \geq 0$ for $0 \leq n < w$ and zero otherwise (w is the window length) then the translated

$$\tilde{V}_j = V[n - jq] \tag{3}$$

and renormalized [4] window function

$$\tilde{V}_j[n] = \frac{V_j[n]}{\sqrt{\sum_{j=0}^{p-1} V_j^2[n]}} \tag{4}$$

verifies

$$\sum_{j=0}^{p-1}\tilde{V}_j^2[n] = 1 \tag{5}$$

and

$$f[n] = \sum_{j=0}^{p-1}\tilde{V}_j^2[n]f[n] \tag{6}$$

over $[q, pq]$ and for a signal f of length $N = (p+1)q$ defined over $[0, N[$.

Therefore we have the following reversible sequence

$$f \longrightarrow \{\tilde{V}_j f\}_{0\leq j<p} \longrightarrow \{\widehat{\tilde{V}_j f}\}_{0\leq j<p} \longrightarrow \{\tilde{V}_j f\}_{0\leq j<p} \longrightarrow \{\tilde{V}_j^2 f\}_{0\leq j<p} \longrightarrow f$$

over $[q, pq]$.

Furthermore, if

$$\tilde{V}_{k,j}[n] = \tilde{V}[n - jq]\exp(\frac{2i\pi(n - jq)k}{w})$$

then

$$S_f[k, j] =< f, \tilde{V}_{k,j} >= \sum_{n=0}^{w-1} f[n + jq]\tilde{V}[n]\exp(-\frac{2i\pi nk}{w}) = \widehat{\tilde{V}_j f}[k]$$

Thus

$$f = \sum_{j,k} < f, \tilde{V}_{k,j} > \tilde{V}_{k,j}$$

and

$$f[n] = \sum_{j=0}^{p-1}\tilde{V}_j[n]\sum_{k=0}^{w-1}S_f[k, j]\exp(\frac{2i\pi nk}{w}) \tag{7}$$

over $[q, pq]$.

In consequence, the signal f can be decomposed as follows

$$f = \sum_{j,k} S_f[k,j]\tilde{V}_{k,j} \tag{8}$$

(the set of functions $(\tilde{V}_{k,j})_{k,j}$ is a tight frame [5] and $S_f[k,j]$ the matrix of coefficients).
The signals are usually extended to $[-q, N+q[$ by symmetry or periodicity to reduce undesirable boundary effects. In this case, the reconstructed functions (**??**) and (**??**) are defined over $[0,(p+1)q[$

Time-frequency representation

The time-frequency behavior of a real signal can be visualized with an image of the coefficient matrix $S_f[k,j]$ in absolute value. Since this matrix is symmetric with respect to a middle row and the *log* values provide a better visualization, the spectrogram is the image of the following matrix

$$I = log(|S_f[k,j]| + \epsilon)_{0\leq j<p,\ k>q}$$

where $\epsilon \approx 10^{-12}$ is used to hide null or quasi-null $S_f[k,j]$ values.

For a better interpretation, this image is shown with the corresponding signal in the same graph. Figure 3 shows an audio signal with noise clicks, and below its spectrogram with windows of 5 milliseconds length. Since this signal is sampled with 44100 Hz, the window length is 221 samples (the number of samples is equal to time window length(in ms)/1000 $*$ 44100).

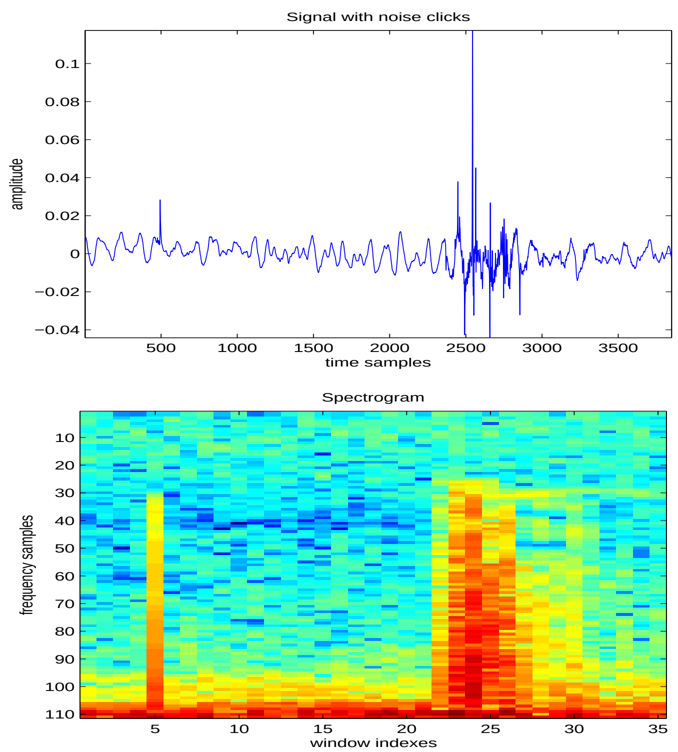

Figure 3: Audio signal (with clicks) and its spectrogram

A speech signal, bird sound and piano chords with theirs spectrograms are shown in Figure 4, 5, 6.

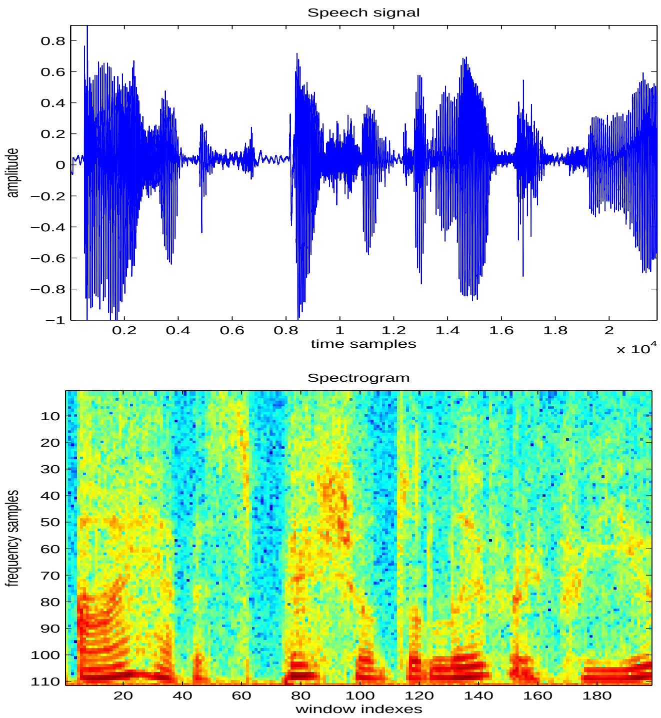

Figure 4: Speech signal and its spectrogram

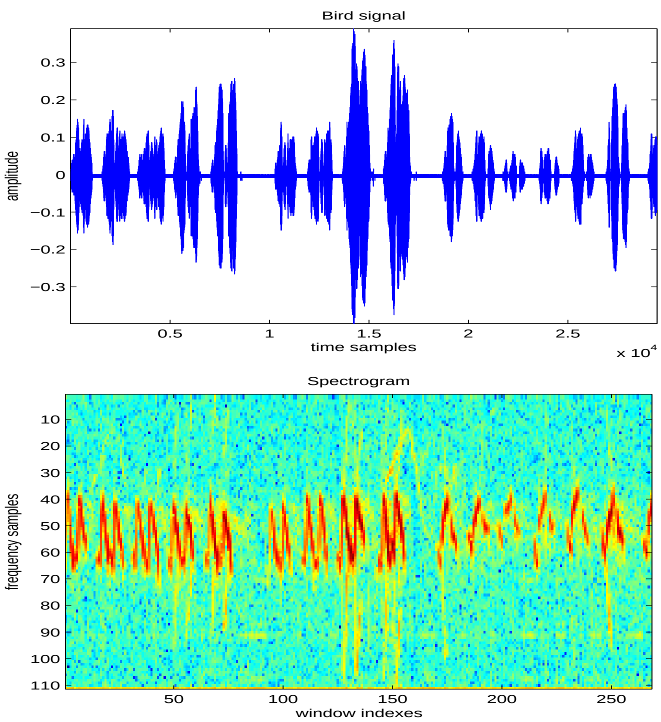

Figure 5: Bird sound with its spectrogram

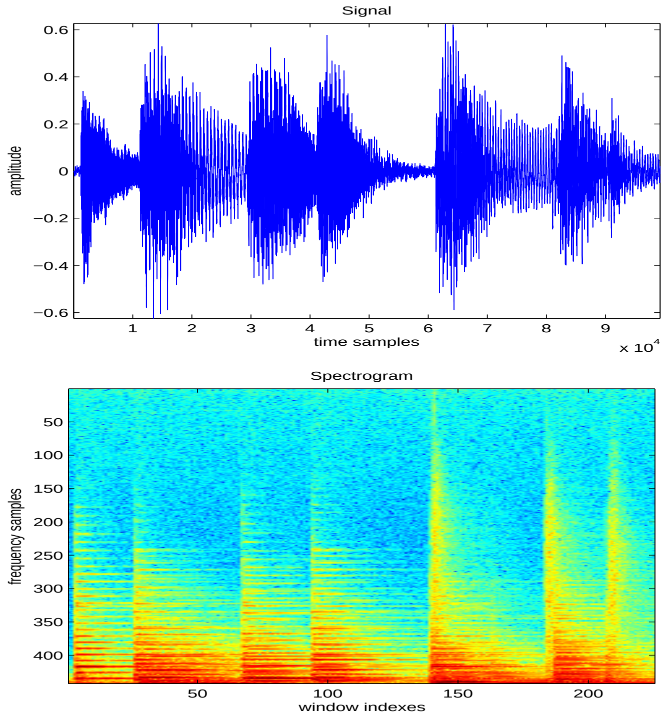

Figure 6: Piano chords and their spectrogram

3 Implementation for sound signals

An ANSI C implementation is available to:

- read and write audio files,
- compute a STFT matrix of coefficients,
- compute a spectrogram (time-frequency image) for different window lengths.

using the SNDFILE and FFTW3 libraries.

The *readSound* function transforms an audio file into a C structure which contains the number of channels, the samples in each channel, its length, sample frequency and number of bits.

The *writeSound* function transforms this C structure into an audiofile.

The *stft* function computes a STFT coefficient matrix of the structure channel array.

The *spectrogram* function transforms this STFT coefficient matrix into an integer color matrix.

The *savepng* function transforms this color matrix into an image matrix.

audio file $\longrightarrow$ C structure (S) $\longrightarrow$ S.channel $\longrightarrow$ coefficient matrix $\longrightarrow$ time-frequency image

The *readSound* and *writeSound* C functions use SNDFILE and the *stft* C function uses FFTW3. In both cases, the implemented functions facilitate the use of these libraries, providing a user-friendly interface".

Algorithm 1: STFT using renormalized window function

Input:

- f: 1D signal (sampled vector),
- N: signal length,
- nW: window length.

Output: STFT matrix

```
Make the vector w containing the window function.
foreach j from 1 to number of windows do
    Translate the window V → V_j (3)
    Renormalize this window V_j → Ṽ_j (4).
end
such that (5).
foreach j from 1 to number of windows do
    Multiply the signal by the renormalized window Ṽ_j
    Compute the fft of Ṽ_j f
    Copy this result in column j of the STFT matrix.
end
```

Algorithm 2: Inverse STFT using renormalized window function

Input:

- M: coefficient matrix,
- N: length of the original signal,
- w: vector containing the window function
- nW: window length.

Output: reconstructed signal.

```
Initialize the reconstructed signal x to 0.
foreach j from 1 to the number of columns do
    Put the column j of the matrix M in a complex vector y_j,
    Compute the inverse fft of this complex vector y_j → z_j,
    Multiply z_j by Ṽ_j
    Add the result to the reconstructed signal (7).
end
```

References

[1] Alan V. Oppenheim, Ronald W. Schafer, Mark T. Yoder, Wayne T. Padgett, Disctrete-Time Signal Processing, Prentice Hall, 2009.

[2] W. Koenig, H. K. Dunn, and L. Y. Lacy, The Sound Spectrograph, J. Acoust. Soc. Am. 18, 1, 19-49 (1946).

[3] Guoshen Yu, Stéphane Mallat, and Emmanuel Bacry, Audio Denoising by Time-Frequency Block Thresholding, IEEE Transactions on Signal Processing, vol. 56, no. 5, May 2008.

[4] Gabriel Peyré, The Numerical Tours of Signal Processing - Advanced Computational Signal and Image Processing, Matapli 94 (2011) 41-64.

[5] Stphane Mallat, A Wavelet tour of signal processing, 3rd edition. Academic Press, Dec. 2008.

[6] Marie de Masson d'Autume, Christophe Varray, Eva Wesfreid, Block thresholding audio denoising algorithm. Submit to J-RASP.