The Large Time-Frequency Analysis Toolbox 2.0

Zdeněk Průša[1](✉), Peter L. Søndergaard[2], Nicki Holighaus[1], Christoph Wiesmeyr[3], and Peter Balazs[1]

[1] Acoustics Research Institute, Austrian Academy of Sciences, Wohllebengasse 12–14, 1040 Vienna, Austria
{zdenek.prusa,nicki.holighaus,peter.balazs}@oeaw.ac.at
[2] Oticon A/S, Kongebakken 9, 2765 Smørum, Denmark
pesg@oticon.dk
[3] Numerical Harmonic Analysis Group, Faculty of Mathematics, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
christoph.wiesmeyr@univie.ac.at

Abstract. The Large Time Frequency Analysis Toolbox (LTFAT) is a modern Octave/Matlab toolbox for time-frequency analysis, synthesis, coefficient manipulation and visualization. It's purpose is to serve as a tool for achieving new scientific developments as well as an educational tool. The present paper introduces main features of the second major release of the toolbox which includes: generalizations of the Gabor transform, the wavelets module, the frames framework and the real-time block processing framework.

Keywords: Frames · Fourier transform · Gabor transform · Wavelet transform · Real-time audio processing

1 Introduction

Time-Frequency analysis is a very important tool for signal processing and its applications in audio, video and acoustics. It allows a representation showing simultaneously (to some extent) the frequency and time content of a signal. Typical representations are the Gabor [16] or wavelet [17] transforms. In recent years more flexible transforms, in the form of adapted and adaptive representations were a very active topic of research, see e.g. [3]. For all those concepts the mathematical theory of frames has proven to be highly significant, as frames allow a very flexible approach, a wide range of possible analysis parameters and properties, while still guaranteeing perfect reconstruction. For applications in particular the implementation of related algorithms are important, an efficient, and possibly real-time, realization being preferable. For a reproducible research it is important to have a stable, well-documented toolbox.

Dealing with those concepts, LTFAT is an open-source Matlab/Octave toolbox freely available at http://ltfat.sourceforge.net/. The toolbox is well documented both in the code itself and in the form of a documentation web page.

© Springer International Publishing Switzerland 2014
M. Aramaki et al. (Eds.): CMMR 2013, LNCS 8905, pp. 419–442, 2014.
DOI: 10.1007/978-3-319-12976-1_25

The features of the first version of the toolbox were presented in [37] which was focused mainly on the *Discrete Gabor Transform – DGT* and window design. This paper focuses on the new additions, which are generalizations of the Gabor transform, both changing the lattice, as well as allowing varying windows; the wavelet modules, including wavelet filterbank trees and wavelet packet transforms; the frames framework, also dealing with multipliers and sparsity; and real-time block processing. The rest of the paper is organized as follows: Sect. 2 gives a brief overview of the frame theory in a finite setting and introduces the frames framework which allows users to effectively work with different transforms using a common interface. Sections 3 and 4 describe generalizations of Gabor systems to systems defined on non-separable and on non-regular time-frequency grids respectively. Section 5 deals with the discrete wavelet transform and derived algorithms. Section 7 provides examples for those sections. Section 6 contains description of several algorithms generalizing the Fourier transform. Section 8 describes the block-stream processing framework which enables real-time audio processing directly in Matlab/Octave. Section 9 discusses the design of the toolbox and states further plans.

1.1 Notation

To be consistent with [37], we use the same notation and assumptions. We regard all signals, windows and transforms as finite-dimensional and periodic. This assumption greatly simplifies the formulas and produces the fastest algorithms but at the same time introduces an unnatural behavior at signal boundaries. The signals are represented as vectors $x = \{x(0), x(1), \ldots, x(L-1)\} \in \mathbb{C}^L$ which are assumed to be column vectors with cyclic indexing such that $x(l + kL) = x(l)$ for $l, k \in \mathbb{Z}$. By $\overline{x}$ we denote a complex conjugation of each element in x. The scalar product on $\mathbb{C}^L$ is defined as $\langle x, y\rangle = \sum_{l=0}^{L-1} x(l)\overline{y}(l)$ and the induced norm as $\|x\| = \sqrt{\langle x, x\rangle}$. A linear operator $\mathcal{O} : \mathbb{C}^L \longrightarrow \mathbb{C}^M$ is represented by a $M \times L$ matrix vector multiplication $(\mathcal{O}x)(m) = \sum_{l=0}^{L-1} o(m, l)x(l)$ for $m \in \{0, \ldots, M-1\}$. All operators mentioned are linear. Finally, we denote $\widehat{x}(k) = \frac{1}{\sqrt{L}} \sum_{l=0}^{L-1} x(l)\mathrm{e}^{-2\pi ikl/L}$ for $k \in \{0, \ldots, L-1\}$ as a (unitary) Discrete Fourier Transform (DFT) of x.

Here we give a brief summary of the DGT which maps a signal $f \in \mathbb{C}^L$ to a set of coefficients $c \in \mathbb{C}^{M\times N}$ using (circular) time shifts and modulations of a window $g \in \mathbb{C}^L$ such that

$$c\,(m, n) = \sum_{l=0}^{L-1} f(l)\mathrm{e}^{-2\pi ilm/M}\overline{g}(l - an) \tag{1}$$

assuming $L = Mb = Na$. Here M denotes the number of frequency channels and a denotes the time step or a hop size in samples. The input length restrictions can be handled either by truncating or by padding[1] of f. The coefficients capture

[1] The toolbox does a zero padding implicitly.

a time-frequency representation of the signal allowing one to study its time-frequency distribution. The choice of g, a and M determines the time-frequency localization of the signal. The windows g can be either full-length or be nonzero only on some smaller interval (FIR). In order to be able to reconstruct signals from their coefficients, $MN \geq L$ is required. This condition is necessary for the system

$$g_{m,n}(l) = \left\{ \mathrm{e}^{2\pi i l m/M} g(l - an) \right\} \tag{2}$$

with $m \in \{0, \ldots, M-1\}$, $n \in \{0, \ldots, L/a - 1\}$ and $l \in \{0, \ldots, L-1\}$ to form a Gabor *frame* for $\mathbb{C}^L$, see also Sect. 2. In this case, the reconstruction can be done using the same parameters a, M but this time using a dual window $\widetilde{g}$ such that

$$f(l) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} c(m,n) \mathrm{e}^{2\pi i m l/M} \widetilde{g}(l - an). \tag{3}$$

The fact that the (canonical) dual frame of a Gabor frame has the same structure is a central property of Gabor frames. An overview of the theory of Gabor frames can be found in [21].

Actual Matlab/Octave functions are referred to in a typewriter style (`functionname`).

2 The Frames Framework

A *frame* in $\mathbb{C}^L$ is a collection of vectors $\Psi = \{\psi_\lambda\}_{\lambda \in \{0,\ldots,\Lambda-1\}}$, $\psi_\lambda \in \mathbb{C}^L$ such that *frame bounds* $0 < A \leq B < \infty$ exist with

$$A\|f\|^2 \leq \sum_{\lambda=0}^{\Lambda-1} |\langle f, \psi_\lambda \rangle|^2 \leq B\|f\|^2,$$

for all $f \in \mathbb{C}^L$. A frame is redundant (oversampled) if $\Lambda > L$ and it is called *tight* if $A = B$. The basic operators associated with frames are the *analysis* and *synthesis operators* which take the form of matrix multiplications. The analysis operator acts as follows: $c = \mathbf{C}_\Psi f = \{\langle f, \psi_\lambda \rangle\}_{\lambda \in \{0,\ldots,\Lambda-1\}}$, where $c \in \mathbb{C}^\Lambda$ is a $\Lambda \times 1$ vector, $\mathbf{C}_\Psi \in \mathbb{C}^{\Lambda \times L}$ is a $\Lambda \times L$ matrix

$$\mathbf{C}_\Psi = \begin{pmatrix} - & \overline{\psi_0} & - \\ - & \overline{\psi_1} & - \\ & \vdots & \\ - & \overline{\psi_{\Lambda-1}} & - \end{pmatrix}$$

and $f \in \mathbb{C}^L$ is a $L \times 1$ vector. The synthesis operator act as $f = \mathbf{D}_\Psi c = \sum_{\lambda=0}^{\Lambda-1} c(\lambda)\psi_\lambda$, where $\mathbf{D}_\Psi \in \mathbb{C}^{L \times \Lambda}$ is a $L \times \Lambda$ matrix being the conjugate transpose of the analysis matrix such that $\mathbf{D}_\Psi = \mathbf{C}_\Psi^*$. Their concatenation $\mathbf{S}_\Psi = \mathbf{D}_\Psi \mathbf{C}_\Psi$ is referred to as the *frame operator* $\mathbf{S}_\Psi \in \mathbb{C}^{L \times L}$. Any frame admits a, possibly non-unique, dual frame, i.e. a frame Ψ^{d} such that the identity can be represented

as $\mathbf{I} = \mathbf{D}_{\Psi^{\mathrm{d}}}\mathbf{C}_{\Psi} = \mathbf{D}_{\Psi}\mathbf{C}_{\Psi^{\mathrm{d}}}$. The most widely used dual is the so called *canonical dual* that can be obtained by applying the inverse frame operator $\mathbf{S}_{\Psi}^{-1}$ to the frame elements. When we prefer to have a tight system for both analysis and synthesis, we can instead use the *canonical tight frame* $\Psi^{\mathrm{t}} = \{\psi_\lambda^{\mathrm{t}}\}_{\lambda\in\{0,\ldots,\Lambda-1\}}$, defined by $\psi_\lambda^{\mathrm{t}} = \mathbf{S}_{\Psi}^{-\frac{1}{2}}\psi_\lambda$ and satisfying $\mathbf{I} = \mathbf{D}_{\Psi^{\mathrm{t}}}\mathbf{C}_{\Psi^{\mathrm{t}}}$. See e.g. [2] for more detailed description of frames in the finite setting.

It is usually not computationally feasible to work with the matrices directly, when considering processing e.g. audio signals, as they normally consist of many thousand samples. Therefore, the frames framework provides an operator-like interface for working with frames without explicitly creating the matrices exploiting fast algorithms whenever they are possible.

2.1 Frames and Object Oriented Programming

The notion of a frame fits very well with the notion of a *class* in the object oriented programming paradigm. A class is a collection of methods and variables that together form a logical entity. A class can be *derived* from another class, in such a case that the derived class *inherits* properties of the original class, and it can extend them in some way. It can supply an implementation of abstract methods or override the existing ones. The derived class can still be referred to as the parent class and thus the same code can be used to work with different derived classes in a unified way. In the frame framework presented in this paper, the frame class serves as the abstract base class from which all other classes are derived. In the following text, we give an overview of the framework interface.

An object of type `frame` is instantiated by the user providing information about which type of frame is desired, and any additional parameters (like a window function, the number of channels etc.) necessary to construct the frame object. This is usually not enough information to construct a frame for $\mathbb{C}^L$ in the mathematical sense, as the dimensionality L of the space is not supplied. Instead, when the analysis operator of a frame object is presented with an input signal, it determines a value of L larger than or equal to the length of the input signal and only at this point is the mathematical frame fully defined. The construction was conceived this way to simplify work with different signal lengths without the need for a new frame for each signal length.

Therefore, each frame type must supply the `framelength` method, which returns the next larger length for which the frame can be instantiated. For instance, a dyadic wavelet frame with J levels only treats signal lengths which are multiples of 2^J. An input signal is simply zero-padded until it has admissible length, but never truncated. Some frames may only work for a fixed length L.

The `frameaccel` method will fix a frame to only work for one specific space $\mathbb{C}^L$. For some frame types, this involves precomputing the data structures to speed up the repeated application of the analysis and synthesis operators. This is highly useful for iterative algorithms, block processing or other types of processing where a predetermined signal length is used repeatedly.

Basic information about a frame can be obtained from the `framebounds` methods, returning the frame bounds, and the `framered` method returning the redundancy $\frac{\Lambda}{L}$ of the frame.

2.2 Analysis and Synthesis

The workhorses of the frame framework are the `frana` and `frsyn` methods, providing the analysis and synthesis operators $\mathbf{C}_\Psi$, $\mathbf{D}_\Psi$ of the frame Ψ respectively. These methods use a fast algorithm if available for the given frame. They are the preferred way of interacting with the frame when writing algorithms. However, if a direct access to the operators is needed, the `frsynmatrix` method returns a matrix representation of the synthesis operator.

The `framedual` and `frametight` methods represent the $\mathbf{S}_\Psi^{-1}$ and $\mathbf{S}_\Psi^{-\frac{1}{2}}$ operators respectively. Again, the matrices are not created and inverted explicitly if a fast algorithm exists. For some frame types, e.g. `filterbank` and `nsdgt`, the canonical dual frame is not necessarily again a frame with the same structure, and therefore it cannot be realized with a fast algorithm. Nonetheless, analysis and synthesis with the canonical dual frame can be realized iteratively. The `franaiter` method implements iterative computation of the canonical dual analysis coefficients using the frame operator's self-adjointness via the equation $\langle f, \mathbf{S}_\Psi^{-1}\psi_\lambda\rangle = \langle \mathbf{S}_\Psi^{-1} f, \psi_\lambda\rangle$. More precisely, a conjugate gradient method (`pcg`) is employed to apply the inverse frame operator $\mathbf{S}_\Psi^{-1}$ to the signal f iteratively, such that the analysis coefficients can be computed quickly by the `frana` method. Note that each conjugate gradient iteration applies both `frana` and `frsyn` once. The method `frsyniter` works in a similar fashion to provide the action of the inverse of the frame analysis operator. Furthermore, for some frame types the diagonal of the frame operator $\mathbf{S}$ calculated by `framediag` can be used as a preconditioner, providing significant speedup whenever the frame operator is diagonally dominant, see e.g. [6].

While both methods `franaiter` and `frsyniter` are available for all frames, they are recommended only if no means of efficient, direct computation of the canonical dual frame exists or its storage is not feasible. Their performance is highly dependent on the frame bounds and the efficiency of `frana` and `frsyn` for the frame type used.

2.3 Advanced Operations with Frames

A *frame multiplier* [4] is an operator constructed by multiplying frame coefficients with a symbol $s \in \mathbb{C}^\Lambda$ such that

$$\mathbf{M}_s f = \sum_{\lambda=0}^{\Lambda-1} s(\lambda)\, \langle f, \psi_\lambda^{\mathrm{a}}\rangle\, \psi_\lambda^{\mathrm{s}},$$

where $\psi_\lambda^{\mathrm{a}}$ and $\psi_\lambda^{\mathrm{s}}$ are simply the λth elements of the analysis and synthesis frames, respectively. The analysis and synthesis frames need not be of the same

type, but they must have exactly the same redundancy. Under which conditions a frame multiplier is invertible, and when this again is a frame multiplier, are non-trivial questions [39,40]. In the LTFAT the inverse `iframemul` is generally computed iteratively by a conjugate gradient method `pcg`.

For a frame Ψ and an input signal $f \in \mathbb{C}^L$, the `franalasso` function returns coefficients $c \in \mathbb{C}^\Lambda$ which minimize the following objective function

$$\frac{1}{2}\left\|f - \mathbf{D}_\Psi c\right\|^2 + \gamma \left\|c\right\|_1, \tag{4}$$

where $\|c\|_1 = \sum_{\lambda=0}^{\Lambda-1} |c(\lambda)|$ and $\gamma \geq 0$ is a penalization coefficient which controls a tradeoff between the "sparsity" of c and the approximation error. The actual minimization is done using the Fast Iterative Soft Thresholding algorithm [8,13]. Another function `franagrouplasso` works similarly but the objective function employs a mixed norm [24] enforcing sparsity along the time or the frequency axis. Currently, this routine only works with frames which have a regular time-frequency distribution of atoms.

Sometimes, the phase of the frame coefficients is lost. For a generic frame more than 4 times redundant, the signal can be reconstructed from the magnitude of the coefficients only [1]. The `frsynabs` function attempts to reconstruct the signal using the iterative Griffin-Lim algorithm [20] or it's fast version [29].

Examples for the mentioned operations can be found in Sect. 7.

3 Discrete Gabor Transform on Non-separable Grids

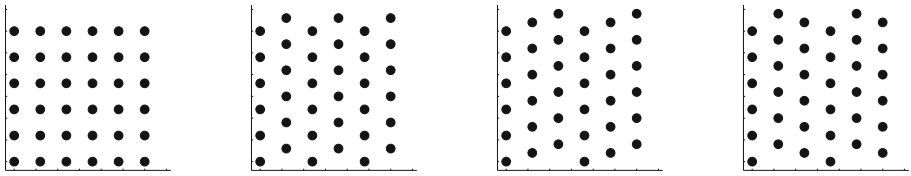
The parameters a and M used in the classical DGT result in a regular, i.e. rectangular grid in the time-frequency plane. On the other hand, Gabor systems on general subgroup lattices $\Lambda \leq \mathbb{Z}_L \times \mathbb{Z}_L$ retain all theoretical properties of Gabor systems on rectangular grids, e.g. that the canonical dual of any *Gabor frame* is again a Gabor frame with respect to the same lattice.

A general lattice can be uniquely defined by using a third parameter $\lambda = \lambda_1/\lambda_2$ in addition to a and M. The *lattice type* λ is an irreducible fraction describing the displacement of neighboring (nonempty) columns in the lattice, relative to the frequency shift $\frac{L}{M}$, see Fig. 1 for an illustration. The corresponding Gabor system is

$$g_{m,n}(l) = \left\{ \mathrm{e}^{2\pi i (m + w(n)) l/M} g(l - an) \right\}, \tag{5}$$

with m, n as before and $w(n) = \mathrm{mod}(n\lambda_1, \lambda_2)/\lambda_2$. More details on Gabor systems on general lattices and their implementation can be found in [44].

In the toolbox, both the classical and the non-separable DGT are available as `dgt`, `idgt` and dual Gabor windows can be computed with `gabdual`. For rectangular and quincunx grids only, `dgtreal` and `idgtreal` facilitate analysis and synthesis of purely real-valued signals, ignoring time-frequency coefficients on negative frequency channels.



(a) $\lambda_1/\lambda_2 = 0$ (b) $\lambda_1/\lambda_2 = 1/2$ (c) $\lambda_1/\lambda_2 = 1/3$ (d) $\lambda_1/\lambda_2 = 2/3$

Fig. 1. The figure shows the placement of the Gabor atoms for four different lattice types in the time-frequency plane. The displayed Gabor system has parameters $a = 6$, $M = 6$ and $L = 36$. The lattice (a) is called *rectangular* or *separable* and the lattice (b) is known as the *quincunx* lattice.

4 Nonstationary Discrete Gabor Transform and Filterbanks

The nonstationary Gabor transform (NSGT) theory [5] generalizes the classical Gabor theory, where the window g, the time step a and the number of frequency channels M are fixed; to systems with evolving properties over either time or frequency. A central result of [5] is the definition of conditions on the window properties which result in *painless* nonstationary Gabor frames which admit an efficient computation of the canonical dual system with the same structure. In this setup, the frame operator is diagonal and its inversion is a simple operation. In the non-painless case, reconstruction is still possible, assuming the system is a frame, but computation of the dual system is not straightforward and it might not retain the original structure.

The painless conditions can be applied either in time or in frequency domain. To avoid confusion, both cases will be shown separately.

4.1 Changing Resolution over Time

Instead of a single window with the fixed time step a, assume a set of N windows $\{g_n\}_{n\in\{0,\dots,N-1\}}$, with g_n centered around the origin and considering M_n frequency channels. The resulting discrete nonstationary Gabor system is given by

$$g_{m,n}(l) = \left\{ \mathrm{e}^{2\pi i(l-a_n)m/M_n} g_n(l-a_n) \right\}, \tag{6}$$

for $n \in \{0,\dots,N-1\}, m \in \{0,\dots,M_n-1\}$ and $l \in \{0,\dots,L-1\}$. In contrast to (2) the complex exponentials shift along with the windows due to the $(l-a_n)$ term. This *phase locked* convention was chosen to simplify the impementation. The system is *painless* given the following conditions are satisfied:

1. Each of the windows g_n is compactly supported with support length being less or equal to M_n. This means that the windows have nonzero values only in some area around the time position.
2. The adjacent windows overlap so that $0 < A \leq \sum_{n=0}^{N-1} |g_n(l-a_n)|^2 \leq B < \infty$, for some positive A and B, for all $l \in \{0,\dots,L-1\}$.

Such systems can be designed to adapt the frequency resolution over time in order to better capture characteristics of an analyzed signal and still provide perfect reconstruction. The NSGT in this setting is implemented in the toolbox as `nsdgt`, its inverse as `insdgt`.

4.2 Changing Resolution over Frequency

Exploiting the duality in time and frequency domains, we assume M compactly supported windows $\{\widehat{g_m}\}_{m\in\{0,\dots,M-1\}}$ in the frequency domain centered around frequency 0. Again, if the frequency support of each $\widehat{g_m}$ is less or equal to N_m and if the windows overlap sufficiently and cover the whole frequency spectrum, the collection

$$\widehat{g_{m,n}}(l) = \left\{ \mathrm{e}^{-2\pi i l n/N_m} \widehat{g_m}(l - b_m) \right\}, \tag{7}$$

for $m \in \{0,\dots,M-1\}$, $n \in \{0,\dots,N_m-1\}$ and $l \in \{0,\dots,L-1\}$ defines the DFT of painless nonstationary Gabor system atoms. An alternative interpretation of this result is that $\{\widehat{g_m}\}$ are band-limited frequency responses of filters in a perfect reconstruction filterbank and each of them is followed by a subsampling operation with a possibly non-integer factor $a_m = \frac{L}{N_m}$. In digital signal processing terms, the analysis filterbank does not introduce aliasing in subbands, and therefore no aliasing cancellation property of the synthesis filterbank is needed. This construction proved to be very useful, because it allows designing perfect reconstruction filterbanks with frequency bands adapted to a specific needs, e.g. the constant-Q Transform (CQT) in [5]. The filters in a CQT are placed along the frequency axis with a constant ratio of center frequency to bandwidth, or Q-factor. This transform is particularly interesting for an acoustic signal processing because it can be tuned to mimic the musical scale allowing to choose the octave resolution (number of filters per octave). An example of a CQT spectrogram is in Fig. 2 on the left.

Another application of the frequency adapted NSGT is the ERBlet transform [27] in which the filters are tuned to mimic the psychoacoustic ERB scale (`erblett`). An example of the ERBlet spectrogram is in Fig. 2 on the right.

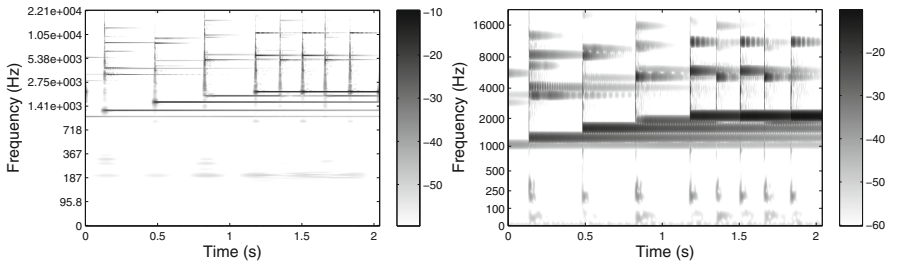


Fig. 2. Examples of the CQT spectrogram (left) and the ERBlet spectrogram (right) of an excerpt of the `gspi` test signal. The figures can be reproduced by running `demo_filterbanks`.

In the toolbox, the frequency adapted nonstationary Gabor systems are implemented in the context of the more general `filterbank` and `ifilterbank` routines.

4.3 Uniform Nonstationary Gabor Systems

Nonstationary Gabor systems are uniform if $M_n = \text{const.}$ or $a_m = \text{const.}$ in the time and frequency adapted settings respectively. Such systems admit another way of computing the canonical dual systems by inverting a polyphase frame matrix [9]. Internally, the toolbox favors the painless algorithm over the uniform one if both are suitable.

5 Discrete Wavelet Transform

The Discrete Wavelet Transform (DWT) associated with a multiresolution analysis provides a dyadic decomposition of $f \in \mathbb{C}^L$ into J wavelet (detail) bands d_j and a single scaling (approximation) band a_J such that the coefficients are obtained by

$$d_j(n) = \sum_{l=0}^{L-1} f(l)\overline{g_j}(l - 2^j n), \quad \text{and} \quad a_J(n) = \sum_{l=0}^{L-1} f(l)\overline{h_J}(l - 2^J n) \tag{8}$$

for $n \in \{0, \dots, N_j - 1\}$, where $N_j = \frac{L}{2^j}$ and $j \in \{1, \dots, J\}$, assuming $L = 2^J N_J$ for some integer N_J and $N_J + \sum_j N_j = L$. Here $g_j, h_J \in \mathbb{C}^L$ are obtained from a pair of characteristic basic wavelet vectors, the scaling sequence h_1 and the wavelet sequence g_1 recursively such that

$$h_{j+1}(l) = \sum_{k=0}^{N_j-1} h_1(k)h_j(l - 2^j k), \qquad g_{j+1}(l) = \sum_{k=0}^{N_j-1} g_1(k)h_j(l - 2^j k). \tag{9}$$

The formulas are sometimes referred to as a *discrete scaling*. In this dyadic setting, the DWT is non-redundant and the reconstruction from the coefficients is possible if h_1 and g_1 and the dual filters $\widetilde{h}_1$ and $\widetilde{g}_1$ form a perfect reconstruction orthogonal (paraunitary) or biorthogonal filterbank, such that h_1 ($\widetilde{h}_1$) and g_1 ($\widetilde{g}_1$) are half-band low-pass and high-pass filters, respectively. Such filters will be further referred to as the *basic wavelet* filters. The dual filters differ from the original ones only in the biorthogonal case. The reconstruction is given by

$$f(l) = \sum_{j=1}^{J} \sum_{n=0}^{N_j} d_j(n)\widetilde{g}_j(l - 2^j n) + \sum_{n=0}^{N_J} a_J(n)\widetilde{h}_J(l - 2^J n), \tag{10}$$

where $\widetilde{h}_J$ and $\widetilde{g}_j$ are derived from the dual filters in the similar manner as in (9).

The commonly used basic wavelet filters are short FIR filters with smooth and slowly decaying frequency responses. This fact exhibits in a poor frequency

selectivity. Combined with the octave-only frequency division coming from the dyadic structure, this makes the DWT seemingly not attractive from the audio signal processing point of view. Nevertheless, the DWT was used in a number of applications dealing with audio signals see e.g. the literature survey in [26]. Moreover, there is a body of wavelet filterbank-based transforms improving upon the DWT properties which are described in the rest of this section.

Fast Wavelet Transform – Mallat's algorithm (`fwt, ifwt`): The Eq. (9) are in fact an enabling factor for the well-known Mallat's algorithm (also known as the fast wavelet transform). The algorithm comprises of an iterative application of the involuted (time reversed and conjugated) elementary two-channel filterbank followed by subsampling by a factor of two

$$d_{j+1} = (a_j * \overline{g_1}(.-l))_{\downarrow 2}\,, \qquad a_{j+1} = \left(a_j * \overline{h_1}(.-l)\right)_{\downarrow 2}\,, \tag{11}$$

where $*$ is the convolution operation and $a_0 = f$. The iterative application of the elementary filterbank forms a tree-shaped filterbank, where just the low-pass output is iterated. The signal reconstruction from the coefficients is then done by applying a mirrored filterbank tree using the dual basic filters $\tilde{g}_1$ and $\tilde{h}_1$. An example of the discrete wavelet representation of a test signal using $J = 11$ levels is depicted in Fig. 3 on the left.

In addition, the routines are capable of working in a more general setting allowing arbitrary number of filters followed by arbitrary subsampling factors in the elementary filterbank as it is required by some generalized wavelet filters constructions e.g. M-band wavelets [38], dual-density wavelets [35], framelets [14] and others.

The toolbox contains an easily extendible collection of routines for generating a number of wavelet filters families (see functions with the `wfilt_` prefix).

Wavelet Filterbank Tree (`wfbt, iwfbt`): Wavelet filterbank trees generalize the DWT filter tree by allowing further recursive decomposition of the high-pass filter output and by allowing a direct definition of a basic filterbank in each of the tree nodes. A flexible frequency covering can be achieved this way. It can also be used to construct more involved wavelet tree filterbanks like the ones used in dual-tree (M-band) complex wavelet transforms [7,36]. An example of a depth 8 full tree decomposition is shown in Fig. 3 on the right.

Wavelet Packet Transform, Best Tree Selection (`wpfbt, iwpfbt, wpbest`): The wavelet packet transform coefficients are formed by outputs of each of the node in the wavelet filterbank tree. Such a representation is highly redundant but leafs of any admissible subtree form a non-redundant representation – a basis, assuming the basic wavelet filterbank in each node is critically subsampled. The best subtree (basis) search algorithm relies on comparing a cost function associated witch each wavelet packet coefficient subband. Both additive [43] and non-additive [42] measures are incorporated in the wavelet module. The search proceeds by pruning the full tree as follows: first, the depth J full wavelet packet decomposition of a signal is performed. Then, the nodes are traversed in the breadth-first order starting from the highest level. At each node, the input cost

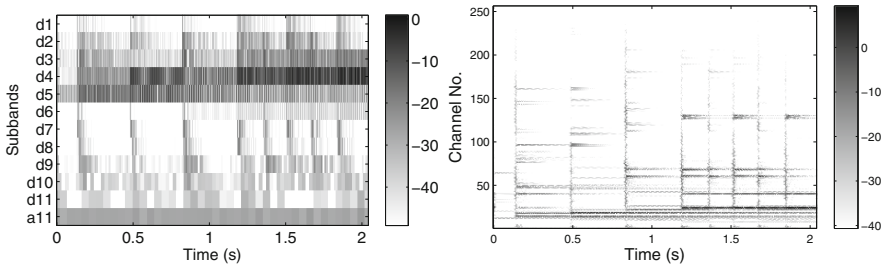


Fig. 3. On the left, the amplitude of DWT of an excerpt of the `gspi` test signal using $J = 11$ levels and the 16tap symlet basic wavelet filters (see help for `wfilt_sym`) is displayed. On the right, there is a representation obtained by a depth 8 full tree decomposition. Both representations are non-redundant. The figures can be reproduced by running `demo_wavelets`.

and the combined output costs are compared. If the input cost is less than the output cost, the current node and all possible descendant nodes are marked to be deleted, if not, the input is assigned the combined output cost. After traversing the whole tree, the marked nodes are removed and the resulting tree is considered to be the best tree (or near-best when using the non-additive cost functions) with respect to the chosen cost function. An example of such a representation is in Fig. 4 on the left. The right plot shows the depth of the node in the tree for the current channel. The lower this number is, the broader is the frequency band and the higher is the number of coefficients in the subband.

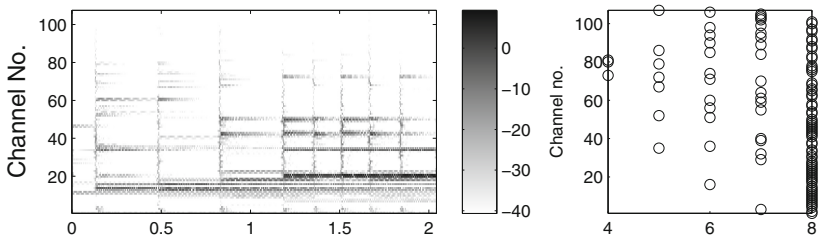


Fig. 4. The best basis representation starting from a full depth-8 wavelet packet tree using the Shannon entropy as the cost function. The figure can be reproduced by running the example in the help section of the `wpbest` function.

There have been several attempts to use wavelet filterbank trees and wavelet packets to process audio signals, mainly in the context of audio compression. The authors of [25] used M-band wavelet filterbanks in order to divide the frequency band into nonlinear frequency bands reminiscent of the tempered musical frequency scale or into an auditory frequency scale. See `demo_wfbt` from the toolbox.

All wavelet-type transformations mentioned so far are also available in undecimated versions in the toolbox (undecimated is sometimes referred to as stationary in the literature). These representations are very redundant, shift-invariant and the subbands are aliasing-free. The lack of aliasing makes the reconstruction more robust against coefficient modifications. A fast À-trous algorithm [23] is used when computing such transforms.

There are several boundary extension techniques available for wavelet based filterbanks implemented in the toolbox. Apart from the default periodic extension, the toolbox supports two types of symmetric extension and extension with zeros, which might lessen the effect of boundary conditions in some situations. Since the wavelet filters are exclusively short FIR filters, the information about the necessary samples beyond the boundaries can be stored in additional coefficients. The downside of this algorithmic approach is that the underlying frame abstraction becomes unclear.

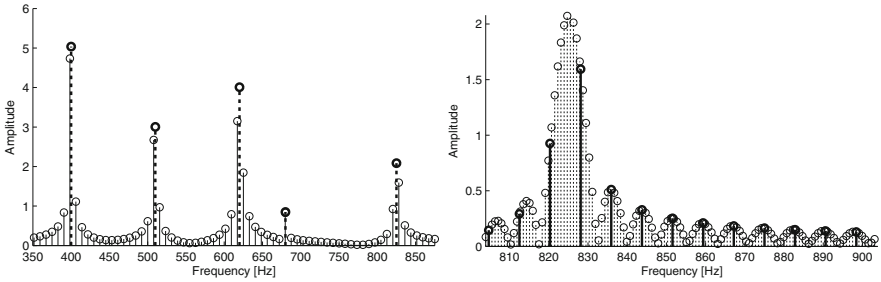
6 Generalized Fourier Transform

The Generalized Goertzel algorithm (`gga`): The traditional Goertzel algorithm [19] (introduced in 1958) is a fast algorithm for evaluating individual samples of the DFT of $f \in \mathbb{C}^L$ i.e.

$$c(k) = \sum_{l=0}^{L-1} f(l) e^{-2\pi i k l / L}. \tag{12}$$

The number of real floating point operations required by the Goertzel algorithm is approximately three-quarters of the operations used in the direct evaluation. The Goertzel algorithm also does not require explicit evaluation of all the complex exponentials. A generalization of the Goertzel algorithm was presented in [41]. It allows obtaining individual values at an arbitrary position on the unit circle such that k in (12) does not have to be an integer, with no increase of the computational complexity. The algorithm can be useful for detecting the presence of harmonic signals with frequencies not being multiples of the fundamental frequency. An example in Fig. 5a shows regular DFT samples (solid lines) of a test signal consisting of a sum of harmonic components and samples obtained by the generalized Goertzel algorithm (bold dashed lines) selecting k to coincide with the known frequencies.

The chirp Z transform (`chirpzt`): The toolbox also contains an implementation of a similar purpose algorithm called the chirp Z transform [34], sometimes incorrectly called the fractional Fourier transform. The algorithm can be used for fast evaluation of K equispaced samples on the unit circle (or more generally on a spiral in the Z-domain) starting at an arbitrary (possibly non-integer) position such that $k = k_0 + nk_d$ in (12) for $n \in \{0, \ldots, K-1\}$ and $k_0, k_d \in \mathbb{R}$. The effective implementation of the algorithm is based on a fast convolution of the signal with a chirp and some pre- and post-processing operations. The algorithm can be used for "zooming" to a specific frequency range as is shown in Fig. 5b



(a) Generalized Goertzel algorithm example

(b) Chirp Z transform example

Fig. 5. Frequency content of a test signal of length 1024 sampled at rate 8 kHz consisting of the sum of five real harmonic components at 400, 510, 620, 680, 825 Hz with amplitudes 5, 3, 4, 1, 2 respectively. The figures can be reproduced by the examples in the help section of the `gga` and `chirpzt` functions respectively.

where the bold solid lines represent the regular DFT samples and the dashed ones are obtained by the chirp Z transform.

Fractional Fourier transform (`dfracft,ffracft`): The *fractional Fourier transform* (FRFT) is a generalization of the classical Fourier transform and has received considerable attention in the literature, the most complete survey to date can be found in [28]. The FRFT of a function depends on the parameter α, for $\alpha = 1$ it coincides with the ordinary FT. We will denote the FRFT as $\mathcal{F}_\alpha$. The parameter α (or more precisely $\alpha\pi/2$) is also referred to as *angle* of the transform, since $\mathcal{F}_\alpha f$ can be regarded as a rotation of the signal f in the TF-plane by the angle $\alpha\pi/2$. The Fourier Transform is a special case and is a rotation by $\pi/2$. For a survey paper on computational aspects of the FRFT we refer to [10], we express our gratitude towards the authors of this paper for allowing us to integrate their code.

Currently there are two different methods available in LTFAT to compute the FRFT for a given angle. The first one is based on the computation of a discrete set of Hermite functions by diagonalizing a discretized version of the Hamiltonian operator (`dfracft`). The (quantum mechanical) Hamiltonian operator is the sum of the squares of the position and the momentum operators. The square of the continuous momentum operator $\mathcal{D}^2 = d^2/dt^2$ can be approximated by the second difference operator $\tilde{\mathcal{D}}^2$ acting on C^L. The square of the momentum operator can consequently be approximated by $\tilde{\mathcal{D}}^2$ on the Fourier side. This leads to an eigenvalue problem of size $L \times L$ for the computation of the discrete Hermite functions, the detailed construction can be found in [11]. Basing the computation of the FRFT on this set of discrete Hermite functions, the transform has all the desirable properties, such as unitarity and index additivity ($\mathcal{F}_\alpha \mathcal{F}_\beta = \mathcal{F}_{\alpha+\beta}$).

The continuous FRFT can be written in several different ways, one of them is through composition of a chirp multiplication followed by a chirp convolution and another chirp multiplication. The integrals can be approximated using

quadrature formulas, which leads to a finite dimensional computational procedure (`ffracft`) with complexity $L \log L$, which is faster than the FRFT based on diagonalization of an $L \times L$ matrix. However, this approximation of the integrals has the disadvantage of being neither precisely unitary, nor does it satisfy the index additivity exactly (Fig. 6).

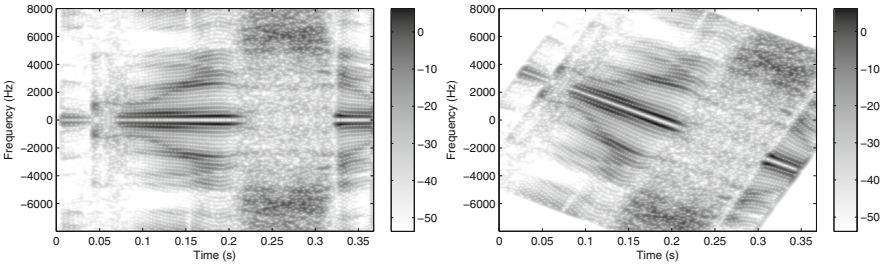
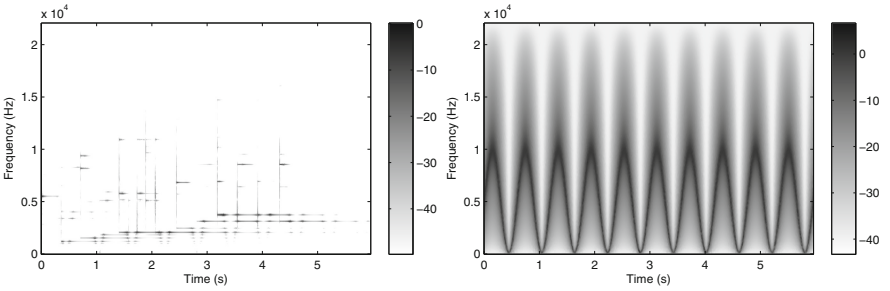


Fig. 6. DGT spectrograms of the `greasy` test signal before (left) and after (right) applying the fractional Fourier transform with $\alpha = 0.3$ using the `ffracft` function.

7 Examples

7.1 Applying a Frame Multiplier

Frame multipliers are useful for separating, deleting and selective enhancement of objects in the time-frequency plane and for approximating responses of linear time-varying systems. Figure 7a shows a spectrogram of a result of applying a frame multiplier using a tight Gabor frame ($a = 200$, $M = 1000$, 20 ms Hann window) with the symbol depicted in Fig. 7b. The symbol approximates a band-pass filter with varying center frequency over time.

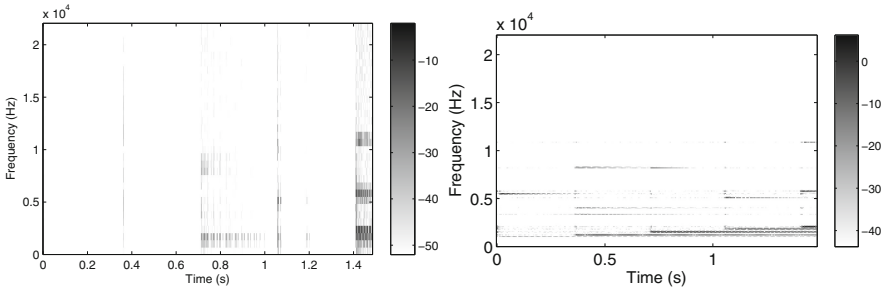


(a) Spectrogram after re-synthesis (b) The symbol

Fig. 7. Simulating an audio effect using a frame multiplier applied to the `gspi` test signal. The figures can be reproduced by running `demo_bpframemul.`

7.2 Enforcing Sparsity

Different signal characteristics like transients or the harmonic structure can be made more prominent by applying transforms with appropriate time-frequency resolutions combined with a procedure which enforces the representation to be *group* sparse in time or frequency. Figure 8a shows the transient part and Fig. 8b the tonal part of an excerpt of the `gspi` test signal. The tonal part is obtained using higher number of frequency channels and forcing the representation to be sparse in frequency and vice versa the transient part.

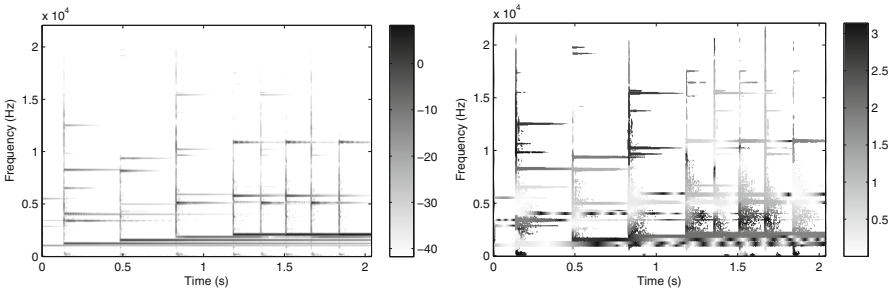


(a) Transients (b) Tonal part

Fig. 8. Separation of the transient and the tonal components using group sparsity. The figures can be reproduced by running `demo_audioshrink`.

7.3 Reconstruction from Magnitude only

Figure 9a depicts the original DGT spectrogram (magnitude of the coefficients) of an excerpt of the `gspi` test signal, whereas Fig. 9b is a visualization of the phase difference between the original phase and the phase reconstructed iteratively using the Griffin-Lim algorithm. The difference is zeroed for coefficients

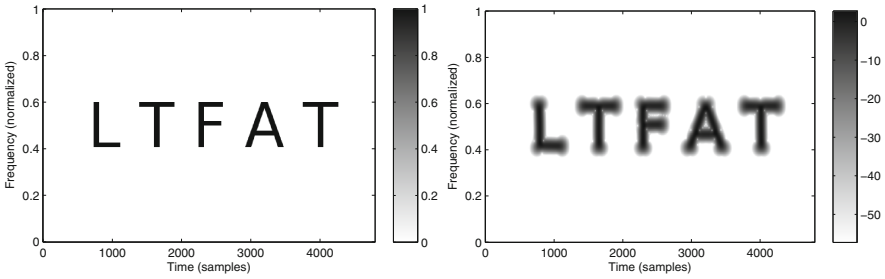


(a) Original spectrogram (b) Phase difference

Fig. 9. Reconstructing a signal from the magnitude of the coefficients only. The figures can be reproduced by running `demo_phaseret`.

smaller than −50 dB. Clearly some regions of the phase in the spectrogram were reconstructed with a constant phase shift, other exhibit a periodically reoccurring patterns in the phase difference.

The reconstruction from the magnitude of the coefficients can be also used for synthetic spectrograms. Figure 10 shows an example of creating an audible sound from an image. The DGT (real) time-frequency grid was used with $a = 8$ and $M = 800$. The iterative algorithm gives a more pleasant sound than the mere direct reconstruction with phase set to zero.



(a) Original spectrogram (linear scale) (b) Spectrogram of the signal with a new phase.

Fig. 10. The figures can be reproduced by running `demo_frsynabs`.

8 Block-Stream Processing Framework

Audio processing experiments become much more impressive when it is allowed to change parameters during the playback or when the data are obtained directly from a microphone, processed and immediately played. In such cases, it is necessary to process small chunks of data to keep the processing delay low. LTFAT provides a simple and unified framework for producing such streams of blocks in order to achieve the real-time data stream capture and the playback directly from the Matlab/Octave without a need for any additional toolbox or low-level programming.

The programming flow consists of setting up the block stream parameters like the data source, audio device, input/output channels etc. using the `block` function and creating the control window object `blockpanel` prior to entering the processing loop. The loop condition checks the value of a state variable, which is unset when the control panel is closed by the user or the Ctrl-C keyboard shortcut is pressed. The loop condition can be altered to check the end of the stream if expected e.g. when using an audio file or a data vector as the source. In each iteration, a new data block is obtained by the `blockread` function and it is enqueued to be played by `blockplay`. The minimal working example is displayed in Fig. 11.

The real-time processing capabilities of Matlab/Octave are quite limited when compared to the professional low-level solutions, therefore we cannot

```
block('playrec');
p = blockpanel({'GdB','Gain',-20,20,0,21});
while p.flag
   gain = blockpanelget(p,'GdB');
   f = blockread();
   blockplay(f*10^(gain/20));
end
blockdone(p);
```

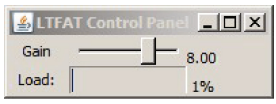


Fig. 11. A minimal block-stream processing example: take input from a microphone and route it trough the loop to the speakers allowing setting gain in the range from −20 to 20 dB during playback using the slider in the panel (on the right).

recommend using the block-processing framework in settings where glitch-less playback is required. Nevertheless, the block-stream framework allows quick prototyping of algorithms in a real-time setting with a minimal programming effort.

8.1 Transform Domain Processing

The block-stream processing framework was designed to be used in conjunction with the frames framework introduced in Sect. 2 in order to provide means for a real-time time-frequency analysis, visualization, modification and synthesis. There are two obstacles when considering applying transforms from the toolbox on a real-time stream of data blocks:

1. The computational complexity of the desired operation.
2. The processed signal periodicity assumption.

The fast execution is achieved by pre-computing all the fixed data by means of the `blockframeaccel` or `blockframepairacel` functions prior entering the processing loop complemented with an efficient C implementation of algorithms. The periodicity assumption goes against the way how data are actually obtained in a real-time setting. Therefore the direct naive application of the transform to individual blocks will produce "bad" coefficients not fit to be directly manipulated or plotted even though the block can be reconstructed perfectly. LTFAT supports two approaches for adapting the transforms to avoid or at least lessen the impact of the assumed periodicity: the combined *overlap-save/overlap-add* approach for transforms using FIR windows/filters and the *slicing window* approach for all other transforms.

(a) The combined overlap-save/overlap-add approach is conceptually similar to the one in [33], where it was used for developing an algorithm for an error-free block-wise discrete wavelet transform. The algorithm is based on a principle of overlap-save (also known as the overlap-discard) type block convolution for the analysis and overlap-add type block convolution for the synthesis. The necessary overlap lengths can be determined exactly from the finite windows/filter lengths the transform is based on.

The algorithm as described here holds for the following assumptions:

1. The window hop size a (or the subsampling factor) is uniform for all windows.
2. The window length is $L^{\mathrm{w}} = k2a+1$ for k being some positive integer and the origin is at the middle sample.
3. The blocks have uniform lengths $L^{\mathrm{b}} = la$ for l being some positive integer.
4. $L^{\mathrm{b}} > L^{\mathrm{w}}$.

These assumptions are often too restrictive in practice, but more general settings require rather large number of additional operations the description of would obscure the principal idea of the algorithm. The current implementation requires the first assumption and uniform length (though arbitrary) FIR windows having the same position of the origin. Moreover, the implementation is able to handle blocks with varying lengths.

Analysis part:

1. Read a block of data, extend it from the left side by the $L^{\mathrm{w}} - 1$ last samples form the previous block.
2. Apply the transform to the extended block.
3. From the resulting coefficients, keep only those at indexes $\{k, \ldots, l + k - 1\}$ starting counting from zero.

The last step discards coefficients which are time-aliased due to the implicit periodic boundary handling. The discarding is done from the both sides because the windows used in the LTFAT computation routines are not causal. The remaining coefficients are equal to the corresponding coefficients from a transform of a signal without dividing it into blocks. The cropped coefficients, possibly modified, form the input for the synthesis part of the algorithm.

Synthesis part:

1. Create zero arrays of length $2k + l$ for each channel and copy the coefficients obtained by the analysis procedure to time positions $\{k, \ldots, l + k - 1\}$.
2. Apply the inverse transformation to the extended coefficients.
3. Recall the $L^{\mathrm{w}} - 1$ last samples from the previous block and add them to the first $L^{\mathrm{w}} - 1$ samples of the current block.
4. Store the last $L^{\mathrm{w}} - 1$ samples as the overlap used in the next block.

A toy example of applying the algorithm is depicted in Figs. 12 and 13 for the analysis and synthesis parts respectively.

(b) The Slicing window method was originally presented in [22]. In contrast to the previous approach, the slicing window method does not try to determine overlaps exactly, but instead employs a slicing window to weigh blocks of samples producing *slices*. After windowing, the slice is optionally symmetrically zero-padded to lessen the effect of the time aliasing. After reconstruction, the slice is weighted by a dual slicing window and the reconstructed signal assembled in an overlap-add manner. The same slicing window can be used for dividing the signal and for the assembly, if the squares of all it's time shifts form a partition

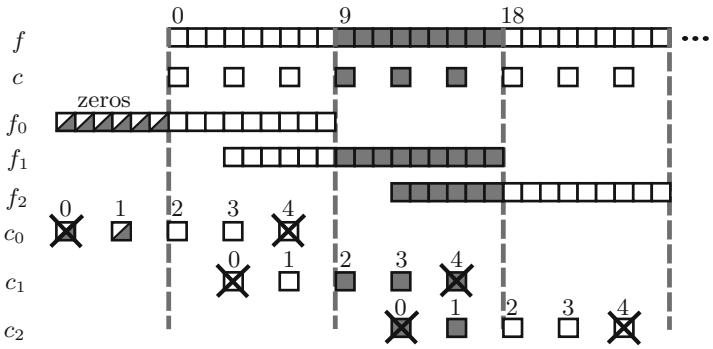


Fig. 12. An example of the analysis part of the combined overlap-save/overlap-add algorithm for $a = 3$, $L^{\mathrm{b}} = 9$ and $L^{\mathrm{w}} = 7$ ($l = 3$, $k = 1$). The figure shows the first three blocks of the signal f and the true time positions of coefficients c. Each of the blocks f_0, f_1 and f_2 is extended from the left side by $L^{\mathrm{w}} - 1 = 6$ samples and transformed. The respective coefficients c_0, c_1, c_2 are obtained and only the ones with indexes $\{1, 2, 3\}$ are retained. Note the algorithm produces k additional coefficients at the beginning when compared to the true coefficients c.

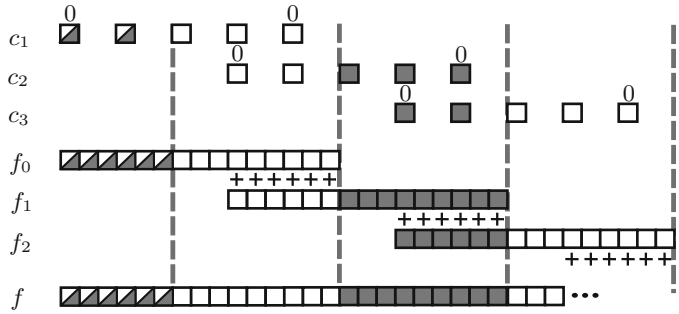


Fig. 13. An example of the synthesis part of the combined overlap-save/overlap-add algorithm. The coefficients c_0,c_1 and c_2 are extended with zeros and the reconstructed blocks f_0,f_1 and f_2 are overlapped. Note the overall delay of the algorithm is $L^{\mathrm{w}} - 1 = 6$ samples.

of unity. More general combinations of windows are also possible, see [22] for the details. Note that the coefficients reflect the shape of the slicing window, so non-linear processing like thresholding applied directly to the coefficients may introduce blocking artifacts. In the toolbox, the method is implemented in such a way the slicing window is applied to a concatenation of the previous and the current block so the slicing window shift is L^{b} effectively. By default, a square-root of a peak-normalized Hann window is used but the programming interface allows specifying customized slicing windows.

In the toolbox, the demos with the `demo_blockproc_` prefix show the block-stream processing framework in action. Figure 14 is a screenshot of one of the demos doing a real-time visualization of the discrete Gabor transform spectrogram of the `gspi` test signal played in the loop. The coefficients are obtained using a FIR window and the analysis part of the combined overlap-save/overlap-add method. Another demo plots a real-time CQT spectrogram. The same test signal is used in the screenshot in Fig. 15. The coefficients are obtained by the slicing window method.

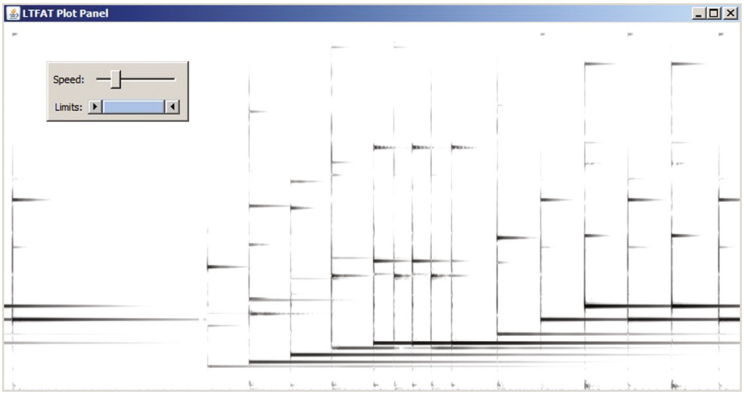


Fig. 14. Real-time DGT spectrogram of the (looped) `gspi` test signal (262144 samples, sampling frequency 44.1 kHz) using $L^{\mathrm{b}} = 1024$, $a = 100$, $M = 3000$ and 20 ms Hann window. For the visualization purposes, absolute values of coefficients are limited to the range -70 dB... 20 dB which is linearly mapped to the inverted grayscale colormap.

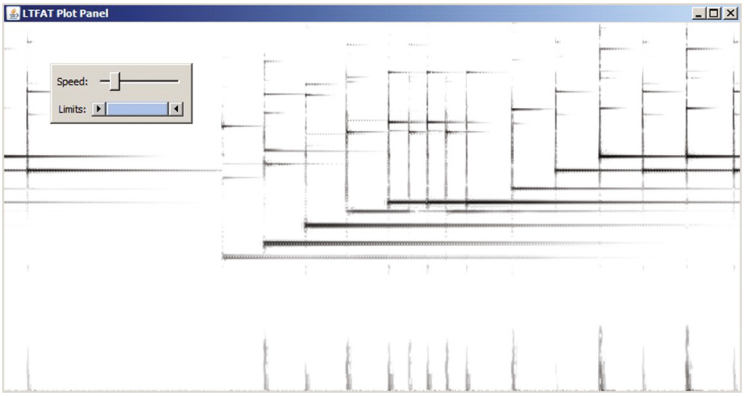


Fig. 15. Real-time CQT spectrogram of the `gspi` test signal. Only frequencies in the range 200 Hz... 20 kHz are displayed using 48 bins per octave (320 bands), $L^{\mathrm{b}} = 1024$, slicing window length $2L^{\mathrm{b}}$ and additional L^{b} zeros of a symmetric zero padding. For visualization purposes, coefficients from consecutive blocks were overlapped to better approximate the true coefficients.

9 Design and Implementation

The toolbox is designed in such a way that the functions forming the programming interface in most cases only check and format the user defined inputs and pass them further to the routines with the `comp_` prefix, that perform the actual computations. The majority of the `comp_` functions can be replaced (shadowed) by compiling the MEX/OCT files with the identical name to speedup the computations. The MEX/OCT files themselves do not contain the computations, but again just format the inputs (unify data types, change complex numbers memory layout) and obtain data pointers and call the actual computational routine(s) from the separately compiled backend C library to which they link to. The backend library depends on the FFTW library and on the BLAS and LAPACK libraries, which are usually already contained in the Matlab/Octave installation. On Windows systems, a manual installation of the FFTW library is necessary at the moment when using Matlab. In order to minimize the code repetition, the backend library is built in such a way the actual code is independent of the desired data type (floating data types with different precision) and even of the real or the complex data type where possible.

The regular toolbox functionality can be used without the backend library and MEX/OCT interfaces. The block-stream processing framework however requires compiling the MEX interface `playrec` (http://www.playrec.co.uk, contained in LTFAT), which depends on the Portaudio library (http://www.portaudio.com), which is again distributed with recent versions of Matlab. The compilation process is automated via the `ltfatmex` command, but pre-built packages can be downloaded from the toolbox webpage. An additional possibility for Octave users is installing LTFAT directly trough the integrated package management system `pkg`, which takes care of compiling everything during the installation process.

10 Outlook

This section describes possible enhancements and features of the toolbox which might be included in the future versions.

Quadratic time-frequency distributions – Although the family of quadratic time-frequency distributions cannot be associated with frames because of their non-linear character, they offer yet another way of studying audio signal features [12]. Moreover, algorithms for synthesizing signals from their quadratic representations exist, so there is a possibility for doing modifications on the coefficients in a similar manner as with frame multipliers.

Modern algorithms for phase-less reconstruction – A reconstruction from only the magnitude of frame coefficients is currently a very active topic in research. We plan to implement some of the modern algorithms, see e.g. [15], to complement the Griffin-Lim algorithm.

Gabor dual windows using convex optimization – Explicit formulas for Gabor dual windows are known only for the canonical dual frame. If the frame system is redundant, infinitely many dual windows exists. Using results from [30,31] we will add an option to search for the optimal dual window given a prior criterion.

Algorithms for computation of optimal dual uniform FIR filterbank frames – The current algorithm for computing dual uniform FIR filterbank (based on [9]) frames in LTFAT suffers from two drawbacks. First, it is capable of computing the canonical dual frame only and it does not preserve the FIR property. The plan is to include results from [18] to allow more freedom in choosing the optimal FIR filterbank dual frames.

Acknowledgments. The authors would like to thank the people that made contributions to the toolbox: Remi Decorsiere, Monika Dörfler, Nina Engelputzeder, Hans Feichtinger, Thomas Hrycak, Florent Jaillet, A.J.E.M. Janssen, Norbert Kaiblinger, Matthieu Kowalski, Ewa Matusiak, Piotr Majdak, Nathanaël Perraudin, Pavel Rajmic, Thomas Strohmer, Bruno Torrésani, Jordy van Velthoven and Tobias Werther.

We would like to express our gratitude towards authors of the Uvi Wave toolbox [32], from which we have taken some wavelet filters generation routines.

The work on this paper was partly supported by the Austrian Science Fund (FWF) START-project FLAME ('Frames and Linear Operators for Acoustical Modeling and Parameter Estimation'; Y 551-N13).

References

1. Balan, R., Casazza, P., Edidin, D.: On signal reconstruction without phase. Appl. Comput. Harmon. Anal. **20**(3), 345–356 (2006)
2. Balazs, P.: Frames and finite dimensionality: frame transformation, classification and algorithms. Appl. Math. Sci. **2**(41–44), 2131–2144 (2008)
3. Balazs, P., Dörfler, M., Kowalski, M., Torrésani, B.: Adapted and adaptive linear time-frequency representations: a synthesis point of view. IEEE Signal Process. Mag. (Special Issue: Time-Freq. Anal. Appl.) **30**(6), 20–31 (2013)
4. Balazs, P.: Basic definition and properties of Bessel multipliers. J. Math. Anal. Appl. **325**(1), 571–585 (2007). http://dx.doi.org/10.1016/j.jmaa.2006.02.012
5. Balazs, P., Dörfler, M., Jaillet, F., Holighaus, N., Velasco, G.A.: Theory, implementation and applications of nonstationary Gabor frames. J. Comput. Appl. Math. **236**(6), 1481–1496 (2011). http://ltfat.sourceforge.net/notes/ltfatnote018.pdf
6. Balazs, P., Feichtinger, H.G., Hampejs, M., Kracher, G.: Double preconditioning for Gabor frames. IEEE Trans. Signal Process. **54**(12), 4597–4610 (2006). http://dx.doi.org/10.1109/TSP.2006.882100
7. Bayram, I., Selesnick, I.W.: On the dual-tree complex wavelet packet and M-band transforms. IEEE Trans. Signal Process. **56**(6), 2298–2310 (2008)
8. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sci. **2**(1), 183–202 (2009). http://dx.doi.org/10.1137/080716542
9. Bölcskei, H., Hlawatsch, F., Feichtinger, H.G.: Frame-theoretic analysis of oversampled filter banks. IEEE Trans. Signal Process. **46**(12), 3256–3268 (2002)
10. Bultheel, A., Martínez, S.: Computation of the fractional Fourier transform. Appl. Comput. Harmon. Anal. **16**(3), 182–202 (2004)

11. Candan, C., Kutay, M.A., Ozaktas, H.M.: The discrete fractional Fourier transform. IEEE Trans. Signal Process. **48**(5), 1329–1337 (2000)
12. Cohen, L.: Time-frequency distributions-a review. Proc. IEEE **77**(7), 941–981 (1989)
13. Daubechies, I., Defrise, M., De Mol, C.: An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Commun. Pure Appl. Math. **57**, 1413–1457 (2004)
14. Daubechies, I., Han, B., Ron, A., Shen, Z.: Framelets: MRA-based constructions of wavelet frames. Appl. Comput. Harmon. Anal. **14**(1), 1–46 (2003)
15. Decorsiere, R., Søndergaard, P.L., Buchholz, J., Dau, T.: Modulation filtering using an optimization approach to spectrogram reconstruction. In: Proceedings of Forum Acusticum 2011. European Acoustics Association (2011)
16. Feichtinger, H.G., Strohmer, T. (eds.): Gabor Analysis and Algorithms. Birkhäuser, Boston (1998)
17. Flandrin, P.: Time-Frequency/Time-Scale Analysis, Wavelet Analysis and its Applications, vol. 10. Academic Press Inc., San Diego (1999). (with a preface by Yves Meyer, Translated from the French by Joachim Stöckler)
18. Gauthier, J., Duval, L., Pesquet, J.: Optimization of synthesis oversampled complex filter banks. IEEE Trans. Signal Process. **57**(10), 3827–3843 (2009)
19. Goertzel, G.: An algorithm for the evaluation of finite trigonometric series. Am. Math. Mon. **65**(1), 34–35 (1958)
20. Griffin, D., Lim, J.: Signal estimation from modified short-time Fourier transform. IEEE Trans. Acoust. Speech Signal Process. **32**(2), 236–243 (1984)
21. Gröchenig, K.: Foundations of Time-Frequency Analysis. Birkhäuser, Boston (2001)
22. Holighaus, N., Dörfler, M., Velasco, G.A., Grill, T.: A framework for invertible, real-time constant-Q transforms. IEEE Trans. Audio Speech Lang. Process. **21**(4), 775–785 (2013)
23. Holschneider, M., Kronland-Martinet, R., Morlet, J., Tchamitchian, P.: A real-time algorithm for signal analysis with the help of the wavelet transform. In: Combes, J.M., Grossmann, A., Tchamitchian, P. (eds.) Wavelets. Time-Frequency Methods and Phase Space, pp. 286–297. Springer, Heidelberg (1989)
24. Kowalski, M.: Sparse regression using mixed norms. Appl. Comput. Harmon. Anal. **27**(3), 303–324 (2009). http://hal.archives-ouvertes.fr/hal-00202904/
25. Kurth, F., Clausen, M.: Filter bank tree and M-band wavelet packet algorithms in audio signal processing. IEEE Trans. Signal Process. **47**(2), 549–554 (1999)
26. Merry, R., Steinbuch, M., van de Molengraft, M.: Wavelet theory and applications, a literature study. DCT 2005.53 (2005). http://alexandria.tue.nl/repository/books/612762.pdf
27. Necciari, T., Balazs, P., Holighaus, N.: Søndergaard, P.L.: The ERBlet transform: an auditory-based time-frequency representation with perfect reconstruction. In: Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013), pp. 498–502. IEEE, Vancouver, May 2013
28. Ozaktas, H.M., Zalevsky, Z., Kutay, M.A.: The Fractional Fourier Transform with Applications in Optics and Signal Processing. Wiley, New York (2001)
29. Perraudin, N., Balazs, P., Sondergaard, P.: A fast Griffin-Lim algorithm. In: 2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), pp. 1–4, Oct 2013
30. Perraudin, N., Holighaus, N., Soendergaard, P., Balazs, P.: Gabor dual windows using convex optimization. In: Proceedings of the 10th International Conference on Sampling Theory and Applications (SAMPTA 2013) (2013)

31. Perraudin, N., Holighaus, N., Søndergaard, P.L., Balazs, P.: Designing Gabor windows using convex optimization (2014). arXiv:1401.6033
32. Prelcic, N.G., Márquez, O.W., González, S.: Uvi Wave, the ultimate toolbox for wavelet transforms and filter banks. In: Proceedings of the Fourth Bayona Workshop on Intelligent Methods in Signal Processing and Communications, Bayona, Spain, pp. 224–227 (1996)
33. Průša, Z.: Segmentwise discrete wavelet transform. Ph.D. thesis, Brno University of Technology, Brno (2012)
34. Rabiner, L., Schafer, R., Rader, C.: The chirp Z-transform algorithm. IEEE Trans. Audio Electroacoust. **17**(2), 86–92 (1969)
35. Selesnick, I.W.: The double density DWT. In: Petrosian, A.A., Meyer, F.G. (eds.) Wavelets in Signal and Image Analysis, pp. 39–66. Springer, Amsterdam (2001)
36. Selesnick, I.W.: The double-density dual-tree DWT. IEEE Trans. Signal Process. **52**(5), 1304–1314 (2004)
37. Søndergaard, P.L., Torrésani, B., Balazs, P.: The linear time frequency analysis toolbox. Int. J. Wavelets Multiresolut. Anal. Inf. Process. **10**(4), 1250032-1–1250032-27 (2012)
38. Steffen, P., Heller, P., Gopinath, R., Burrus, C.: Theory of regular M-band wavelet bases. IEEE Trans. Signal Process. **41**(12), 3497–3511 (1993)
39. Stoeva, D.T., Balazs, P.: Invertibility of multipliers. Appl. Comput. Harmon. Anal. **33**(2), 292–299 (2012)
40. Stoeva, D.T., Balazs, P.: Canonical forms of unconditionally convergent multipliers. J. Math. Anal. Appl. **399**, 252–259 (2013)
41. Sysel, P., Rajmic, P.: Goertzel algorithm generalized to non-integer multiples of fundamental frequency. EURASIP J. Adv. Signal Process. **2012**(1), 56 (2012)
42. Taswell, C.: Near-best basis selection algorithms with non-additive information cost functions. In: Proceedings of the IEEE International Symposium on Time-Frequency and Time-Scale Analysis, pp. 13–16. IEEE Press (1994)
43. Wickerhauser, M.V.: Lectures on wavelet packet algorithms. In: Lecture notes, INRIA (1992)
44. Wiesmeyr, C., Holighaus, N., Søndergaard, P.L.: Efficient algorithms for discrete Gabor transforms on a nonseparable lattice. IEEE Trans. Signal Process. **61**(20), 5131–5142 (2013)