An Efficient Algorithm for the Discrete Gabor Transform using full length Windows

Peter L. Søndergaard

Abstract

This paper extends the efficient factorization of the Gabor frame operator developed by Strohmer in [1] to the Gabor analysis/synthesis operator. This provides a fast method for computing the discrete Gabor transform (DGT) and several algorithms associated with it. The algorithm is used for the case when the involved window and signal have the same length.

I. Introduction

THE finite, discrete Gabor transform (DGT) of a signal f of length L is given by

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

Here g is a window that localizes the signal in time and in frequency. The DGT is equivalent to a Fourier modulated filter bank with M channels and decimation in time a, [2].

Efficient computation of a DGT can be done by several methods: If the window g has short support, a filter bank based approach can be used. We shall instead focus on the case when g and f are equally long. In this case a factorization approach developed by Zibulski and Zeevi in [3] for the case of the frame operator of Gabor frames for $L^2(\mathbb{R})$ can be used. The method was adapted for the finite, discrete setting by Bastiaans and Geilen in [4], and extended to cover also the analysis/synthesis operator. A simple, but not so efficient, method was developed for the Gabor analysis/synthesis operator by Prinz in [5] and later extended by Strohmer [1] to provide the fastest known method for computing the Gabor frame operator. This paper extends Prinz' and Strohmer's method to also cover the Gabor analysis and synthesis methods on multisignals.

The advantage of the method developed in this paper as compared to the one developed in [4], is that it works with FFTs of shorter length, and do not require multiplication by complex exponentials. The asymptotic running time of the two method are the same.

We shall study the DGT applied to multiple signals at once. This is a common case for instance when computing a multidimensional, separable DGT, then this can be done by applying several multisignal DGTs. The DGT defined by (1) works on a multisignal $f \in \mathbb{C}^{L\times W}$, where $W \in \mathbb{N}$ is the number of signals.

II. Definitions

We shall denote the set of integers between zero and some number L by

$$\langle L\rangle = 0,\ldots,L-1. \tag{2}$$

The Discrete Fourier Transform (DFT) of a signal $f \in \mathbb{C}^L$ is defined by

$$(\mathcal{F}_L f)(k) = \frac{1}{\sqrt{L}}\sum_{l=0}^{L-1} f(l)e^{-2\pi i k l/L}. \tag{3}$$

We shall use the $\cdot$ notation in conjunction with the DFT to denote the variable over which the transform is to be applied.

The folding $f * g$ of two functions $f,g \in \mathbb{C}^L$ and the involution f^* is given by

$$(f*g)(l) = \sum_{k=0}^{L-1} f(k)\,g(l-k)\,,\quad l\in\langle L\rangle \tag{4}$$

$$f^*(l) = \overline{f(-l)},\quad l\in\langle L\rangle\,. \tag{5}$$

Both folding and involution has special properties with respect to the Fourier transform

$$\widehat{f*g} = \sqrt{L}\hat{f}\hat{g} \tag{6}$$

$$\widehat{f^*} = \overline{\hat{f}} \tag{7}$$

Manuscript received March 14, 2007; revised XXX XX, 2007.

The Poisson summation formula in the finite, discrete setting is given by

$$\mathcal{F}_M\left(\sum_{k=0}^{b-1} g(\cdot + kM)\right)(m) \quad = \quad \sqrt{b}\left(\mathcal{F}_L g\right)(mb), \tag{8}$$

where $g \in \mathbb{C}^L$, $L = Mb$ with $b, M \in \mathbb{N}$.

A family of vectors e_j, $j \in \langle J \rangle$ of length L is called a *frame* if constants $0 < A \leq B$ exist such that

$$A\left\|f\right\|^2 \leq \sum_{j=0}^{J-1} \left|\langle f, e_j \rangle\right|^2 \leq B\left\|f\right\|^2, \quad \forall f \in \mathbb{C}^L. \tag{9}$$

The constants A and B are called lower and upper frame bounds. If $A = B$, the frame is called *tight*. If $J > L$, the frame is redundant (oversampled). Finite- and infinite dimensional frames are described in [6].

A finite, discrete *Gabor system* (g, a, M) is a family of functions $g_{m,n} \in \mathbb{C}^L$ of the following form

$$g_{m,n}(l) = e^{2\pi i l m/M} g\left(l - na\right), \tag{10}$$

for $m \in \langle M \rangle$ and $n \in \langle N \rangle$ where $L = aN$ and $M/L \in \mathbb{N}$. A Gabor system that is also a frame is called a *Gabor frame*. The analysis operator $C_g : \mathbb{C}^L \mapsto \mathbb{C}^{M \times N}$ associated to a Gabor system (g, a, M) is given by

$$c(m,n) = C_g f = \sum_{l=0}^{L-1} f(k) e^{-2\pi i m l/M} \overline{g(l - an)}. \tag{11}$$

This is exactly the DGT from (1). The adjoint operator is the Gabor synthesis operator $D_\gamma : \mathbb{C}^{M \times N} \mapsto \mathbb{C}^L$ associated to a Gabor system (γ, a, M) given by

$$f = D_\gamma c = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} c(m,n)\, e^{2\pi i m l/M} \gamma(l - an). \tag{12}$$

In (1) it must hold that $L = Na = Mb$ for some $M, N \in \mathbb{N}$. Additionally, we define $c, d, p, q \in \mathbb{N}$ by

$$c = \gcd(a, M) \quad , \quad d = \gcd(b, N), \tag{13}$$

$$p = \frac{a}{c} = \frac{b}{d} \quad , \quad q = \frac{M}{c} = \frac{N}{d}, \tag{14}$$

where GCD denotes the greatest common divisor of two natural numbers. With these numbers, the redundancy of the transform can be written as $L/(ab) = q/p$, where q/p is an irreducible fraction. It holds that $L = cdpq$.The Gabor *frame operator* $S_g : \mathbb{C}^L \mapsto \mathbb{C}^L$ of a Gabor frame (g, a, M) is given by the composition of the analysis and synthesis operators $S_g = D_g C_g$. The Gabor frame operator is important because it can be used to find the *canonical dual window* $g^d = S_g^{-1} g$ and the *canonical tight window* $g^t = S_g^{-1/2} g$ of a Gabor frame. The canonical dual window is important because C_g and D_{g^t} are each others inverses. This gives an easy way to construct the inverse transform of the DGT. Similarly, then C_{g^t} and D_{g^t} are each others inverses. For more information on Gabor systems and properties of the operators C, D and S see [7], [8], [9].

III. The method

We wish to make an efficient calculation of all the coefficients of the DGT. Using (1) literally to compute all coefficients $c(m, n, w)$ would require $8MNLW$ flops.

To derive a faster DGT, an approach would be to consider the analysis operator C_g as a matrix, and derive a faster algorithm through unitary matrix factorizations of this matrix. This is indeed the approach taken by [10], [1]. Unfortunately, this approach tends to introduce many permutation matrices and Kronecker product matrices, so in this paper we have chosen to derive the algorithm by directly manipulating the sums of the definition.

To find more efficient algorithms, the first step is to recognize that the summation and the modulation term in (1) can be expressed as a DFT:

$$c(m,n,w) = \sqrt{L}\mathcal{F}_L\left(f(\cdot, w)\overline{g(\cdot - an)}\right)(mb). \tag{17}$$

We can improve on this because we do not need all the coefficients computed by the Fourier transform appearing in (17), only every b'th coefficient. Therefore, we can rewrite by the Poisson summation formula (8):

$$\begin{aligned} & c(m,n,w) \\ = \quad & \sqrt{M}\mathcal{F}_M\left(\sum_{\tilde{m}=0}^{b-1} f(\cdot + \tilde{m}M, w)\overline{g(\cdot + \tilde{m}M - an)}\right)(m) \\ = \quad & \left(\mathcal{F}_M K(\cdot, n, w)\right)(m), \end{aligned} \tag{18}$$

Algorithm 1 Multisignal DGT by matrix/matrix products.

We wish to compute the DGT $c(m,n,w) \in \mathbb{C}^{M\times N\times W}$ of $f \in \mathbb{C}^{L\times W}$ using the window $g \in \mathbb{C}^{L}$ and lattice determined by a and M.

1) Define $\Psi^{f}_{r,s}(k,l) \in \mathbb{C}^{p\times qW}$ and $\Phi^{g}_{r,s}(k,l) \in \mathbb{C}^{p\times q}$ for $r \in \langle c\rangle$, $s \in \langle d\rangle$ and $w \in \langle W\rangle$ by

$$\begin{aligned} & \tilde{\Psi}^{f}_{r,\tilde{s}}(k,l+qw) \\ = \; & f(r+kM+\tilde{s}pM-lh_a a,w), \\ & \tilde{\Phi}^{g}_{r,\tilde{s}}(k,l) \\ = \; & \sqrt{Md}\,g(r+kM+\tilde{s}pM-la). \end{aligned}$$

2) Compute their DFTs along $\tilde{s}$:

$$\begin{aligned} \Psi^{f}_{r,s}(k,l+qw) &= \mathcal{F}_d\left(\tilde{\Psi}^{f}_{r,\cdot}(k,l+qw)\right)(s), \\ \Phi^{g}_{r,s}(k,l) &= \mathcal{F}_d\left(\tilde{\Phi}^{g}_{r,\cdot}(k,l)\right)(s). \end{aligned}$$

3) Multiply the matrices for each r,s:

$$\Upsilon_{r,s} = \left(\Phi^{g}_{r,s}\right)^{*}\Psi^{f}_{r,s}$$

4) Compute the inverse DFT of $\Upsilon_{r,s}$ along s:

$$\tilde{\Upsilon}_{r,\tilde{s}}(u,l+wq) = \mathcal{F}^{-1}_{d}\left(\Upsilon_{r,\cdot}(u,l+wq)\right)(\tilde{s}).$$

5) Compute $K \in \mathbb{C}^{M\times N\times W}$ as:

$$K(r+lc,u+\tilde{s}q-lh_a,w) = \tilde{\Upsilon}_{r,\tilde{s}}(u,l+wq) \tag{15}$$

6) Finally, the result is given by a DFT of K along the first dimension:

$$c(m,n,w) = \mathcal{F}_M\left(K(\cdot,n,w)\right)(m). \tag{16}$$

---

where

$$K(j,n,w) = \sqrt{M}\sum_{\tilde{m}=0}^{b-1} f(j+\tilde{m}M,w)\,\overline{g}(j+\tilde{m}M-na), \tag{19}$$

for $j \in \langle M\rangle$ and $n \in \langle N\rangle$. From (18) it can be seen that computing the DGT of a signal f can be done by computing K followed by a DFT along the first dimension of K.

We split j as $j = r+lc$ with $r \in \langle c\rangle$, $l \in \langle q\rangle$ and introduce $h_a, h_M \in \mathbb{Z}$ such that the following is satisfied:

$$c = h_M M - h_a a. \tag{20}$$

The two integers h_a, h_M can be found by the extended Euclid algorithm for computing the GCD of a and M.

Using (20) and the splitting of j we can express (19) as

$$\begin{aligned} & K(r+lc,n,w) \\ = \; & \sqrt{M}\sum_{\tilde{m}=0}^{b-1} f(r+lc+\tilde{m}M,w)\times \\ & \times\overline{g}(r+l(h_M M-h_a a)+\tilde{m}M-na) \qquad (21) \\ = \; & \sqrt{M}\sum_{\tilde{m}=0}^{b-1} f(r+lc+\tilde{m}M,w)\times \\ & \times\overline{g}(r+(\tilde{m}+lh_m)M-(n+lh_a)a) \qquad (22) \end{aligned}$$

We set $\tilde{m}' = \tilde{m}+lh_m$ and $n' = n+lh_a$ and get

$$\begin{aligned} & K(r+lc,n'-lh_a,w) \\ = \; & \sqrt{M}\sum_{\tilde{m}'=0}^{b-1} f(r+lc+(\tilde{m}'-lh_m)M,w)\times \\ & \times\overline{g}(r+\tilde{m}'M-n'a) \end{aligned} \tag{23}$$

$$\begin{aligned} &= \sqrt{M}\sum_{\tilde{m}'=0}^{b-1} f\left(r+\tilde{m}'M+l\left(c-h_mM\right),w\right)\times \\ &\times\overline{g}\left(r+\tilde{m}'M-n'a\right) \end{aligned} \tag{24}$$

For simplicity, we continue without the primes in (24). We split $\tilde{m}=k+\tilde{s}p$ with $k\in\langle p\rangle$ and $\tilde{s}\in\langle d\rangle$ and $n=u+sq$ with $u\in\langle q\rangle$ and $s\in\langle d\rangle$ and use that $M=cq$, $a=cp$ and $c-h_mM=-h_aa$:

$$\begin{aligned} &K\left(r+lc,u+sq-lh_a,w\right) \\ &= \sqrt{M}\sum_{k=0}^{p-1}\sum_{\tilde{s}=0}^{d-1} f\left(r+kM+\tilde{s}pM-lh_aa,w\right)\times \\ &\times\overline{g}\left(r+kM-ua+\left(\tilde{s}-s\right)pM\right) \end{aligned} \tag{25}$$

Define

$$\tilde{\Psi}^f_{r,\tilde{s}}\left(k,l+wq\right) = f\left(r+kM+\tilde{s}pM-lh_aa,w\right), \tag{26}$$

$$\tilde{\Phi}^g_{r,\tilde{s}}\left(k,u\right) = \sqrt{M}g\left(r+kM+\tilde{s}pM-ua\right), \tag{27}$$

We can then write (25) as

$$\begin{aligned} &K\left(r+lc,u+sq-lh_a,w\right) \\ &= \sum_{k=0}^{p-1}\sum_{\tilde{s}=0}^{d-1}\tilde{\Psi}^f_{r,\tilde{s}}\left(k,l+wq\right)\overline{\tilde{\Phi}^g_{r,\tilde{s}-s}\left(k,u\right)} \end{aligned} \tag{28}$$

The sum over $\tilde{s}$ can be seen as a special form of convolution. The combination (6) and (7) yields

$$\left(f*g^*\right)(l) = \sum_{k=0}^{L-1} f(k)\,\overline{g}(k-l) \tag{29}$$

$$\widehat{f*g^*} = \sqrt{L}\hat{f}\overline{\hat{g}}, \tag{30}$$

and it is the kind of convolution as in (29) the we shall use. Fully written out (30) is

$$\left(f*g^*\right)(l) = \sqrt{L}\mathcal{F}_L^{-1}\left(\hat{f}\left(\cdot\right)\overline{\hat{g}}\left(\cdot\right)\right)(l).$$

Define the Fourier transforms along $\tilde{s}$ of $\tilde{\Psi}$ and $\tilde{\Phi}$ by

$$\Psi^f_{r,s}\left(k,l\right) = \left(\mathcal{F}_d\tilde{\Psi}^f_{r,\cdot}\left(k,l\right)\right)(s) \tag{31}$$

$$\Phi^g_{r,s}\left(k,u\right) = \left(\mathcal{F}_d\tilde{\Phi}^g_{r,\cdot}\left(k,u\right)\right)(s) \tag{32}$$

Using (30) we can now write (28) as

$$\begin{aligned} &K\left(r+lc,u+\tilde{s}q-lh_a,w\right) \\ &= \sqrt{d}\sum_{k=0}^{p-1}\mathcal{F}_d^{-1}\left(\Psi^f_{r,\cdot}\left(k,l+wq\right)\overline{\Phi^g_{r,\cdot}\left(k,u\right)}\right)(\tilde{s}) \qquad (33) \\ &= \sqrt{d}\mathcal{F}_d^{-1}\left(\sum_{k=0}^{p-1}\Psi^f_{r,\cdot}\left(k,l+wq\right)\overline{\Phi^g_{r,\cdot}\left(k,u\right)}\right)(\tilde{s}) \qquad (34) \end{aligned}$$

If we consider $\Psi^f_{r,s}$ and $\Phi^g_{r,s}$ as matrices for each r and s, then sum over k in the last line can be written as matrix products. Algorithm 1 follows from this.

IV. Extensions

The algorithm just developed can also be used to calculate the synthesis operator D_γ. This is done by applying Algorithm 1 in the reverse order and inverting each step in the algorithm. All the steps can be trivially inverted except step 3, which becomes

$$\Psi^f_{r,s} = \left(\Phi^\gamma_{r,s}\right)\Upsilon_{r,s}. \tag{35}$$

If the matrices $\Phi^\gamma_{r,s}$ are all left-inverses of the matrices $\Phi^g_{r,s}$ then (35) will invert step 3 in Algorithm 1. This is the case if γ is a dual Gabor window of the Gabor frame (g,a,M). It also holds that all dual Gabor windows γ of a Gabor frame (g,a,M) must satisfy that $\Phi^\gamma_{r,s}$ are left-inverses of $\Phi^g_{r,s}$. This criterion was reported in [11], [12].

A special left-inverse in the *Moore-Penrose pseudo-inverse*. Taking the pseudo-inverses of $\Phi^g_{r,s}$ yields the factorization associated with the canonical dual window of (g, a, M), [13]. Taking the polar decomposition of each matrix in $\Phi^g_{r,s}$ yields a factorization of the canonical tight window (g, a, M). For more information on these methods, as well as iterative methods for computing the canonical dual/tight windows, see [14].

V. Special cases

We shall consider some special cases of the algorithm:

1) Integer oversampling. When the redundancy is an integer then $p = 1$. Because of this we see that $c = a$ and $d = b$. This gives (20) the appearance
$$a = h_M qa - h_a a,$$
indicating that $h_M = 0$ and $h_a = -1$ solves the equation for all a and q. The algorithm simplifies accordingly, and reduces to the well known Zak-transform algorithm for this case, [15].
2) Short time Fourier transform. In this case $a = b = 1$, $M = N = L$, $c = d = 1$, $p = 1$, $q = L$ and as in the previous special case $h_M = 0$ and $h_a = -1$. In this case the algorithm reduces to the very simple, and well known algorithm, for computing the STFT.

VI. Efficient implementation

The reason for defining the algorithm on multisignals, is that the multiple signals can be handled at once in the matrix product in step 3. This is a matrix product of two matrices size $q \times p$ and $p \times qW$, so the second matrix grows when multiple signals are involved. Doing it this way reuse the $\Phi^g_{r,s}$ matrices as much as possible, and this is an advantage on standard, general purpose computers with a deep memory hierarchy, see [16], [17].

The are several ways to execute Algorithm 1. One is of course two execute the steps in the order as they are written. Another possibility comes from the fact that step 1 - 5 can be done in parallel over the variable r. This may be exploited as follows:

1) The algorithm can be split such that each processor on a parallel machine handles step 1-5 for a specific range of values for r.
2) One may loop over step 1-5 for each value of r. This lowers the memory requirement because only a subset of the matrices $\Psi^f_{r,s}$ and $\Phi^g_{r,s}$ needs to be kept in memory at once.
3) One may do the algorithm as it is written, doing the loop over r as the innermost loop. This make the memory access more efficient, because the values indexed by r are stored consecutively in memory. On typical computing machinery, elements stored in consecutive memory locations can be loaded much faster than scattered elements.

Any combination of the above three methods can be done, depending on the number of processors and memory that is available on the machine.

Implementations of the algorithms described in this paper can be found in the Linear Time Frequency Toolbox (LTFAT) available from http://www.univie.ac.at/nuhag-php/ltfat/.

References

[1] T. Strohmer, “Numerical algorithms for discrete Gabor expansions,” in Feichtinger and Strohmer [8], ch. 8, pp. 267–294.
[2] H. Bölcskei, F. Hlawatsch, and H. G. Feichtinger, “Equivalence of DFT filter banks and Gabor expansions,” in *SPIE 95, Wavelet Applications in Signal and Image Processing III*, vol. 2569, part I, (San Diego), july 1995.
[3] Y. Y. Zeevi and M. Zibulski, “Oversampling in the Gabor scheme,” *IEEE Trans. Signal Process.*, vol. 41, no. 8, pp. 2679–2687, 1993.
[4] M. J. Bastiaans and M. C. Geilen, “On the discrete Gabor transform and the discrete Zak transform,” *Signal Process.*, vol. 49, no. 3, pp. 151–166, 1996.
[5] P. Prinz, “Calculating the dual Gabor window for general sampling sets,” *IEEE Trans. Signal Process.*, vol. 44, no. 8, pp. 2078–2082, 1996.
[6] O. Christensen, *An Introduction to Frames and Riesz Bases*. Birkhäuser, 2003.
[7] K. Gröchenig, *Foundations of Time-Frequency Analysis*. Birkhäuser, 2001.
[8] H. G. Feichtinger and T. Strohmer, eds., *Gabor Analysis and Algorithms*. Boston: Birkhäuser, 1998.
[9] H. G. Feichtinger and T. Strohmer, eds., *Advances in Gabor Analysis*. Birkhäuser, 2003.
[10] S. Qiu, “Discrete Gabor transforms: The Gabor-gram matrix approach,” *J. Fourier Anal. Appl.*, vol. 4, no. 1, pp. 1–17, 1998.
[11] A. J. E. M. Janssen, “On rationally oversampled Weyl-Heisenberg frames,” *Signal Process.*, pp. 239–245, 1995.
[12] A. J. E. M. Janssen, “The duality condition for Weyl-Heisenberg frames,” in Feichtinger and Strohmer [8], ch. 1, pp. 33–84.
[13] O. Christensen, “Frames and pseudo-inverses,” *J. Math. Anal. Appl.*, vol. 195, pp. 401–414, 1995.
[14] A. J. E. M. Janssen and P. L. Søndergaard, “Iterative algorithms to approximate canonical Gabor windows: Computational aspects,” *J. Fourier Anal. Appl.*, vol. accepted for publication, 2007.
[15] A. J. E. M. Janssen, “The Zak transform: a signal transform for sampled time-continuous signals,” *Philips Journal of Research*, vol. 43, no. 1, pp. 23–69, 1988.
[16] J. Dongarra, J. Du Croz, S. Hammarling, and I. Duff, “A set of level 3 basic linear algebra subprograms,” *ACM Trans. Math. Software*, vol. 16, no. 1, pp. 1–17, 1990.
[17] R. C. Whaley, A. Petitet, and J. Dongarra, “Automated empirical optimization of software and the ATLAS project,” Tech. Rep. UT-CS-00-448, University of Tennessee, Knoxville, TN, Sept. 2000.