A FLEXIBLE IMPLEMENTATION OF MATCHING PURSUIT FOR GABOR DICTIONARIES

S. E. FERRANDO, L. A. KOLASA, AND N. KOVAČEVIĆ

Abstract. The matching pursuit algorithm of Mallat et. al. is introduced in the context of frame theory. The technical report then proceeds to describe a software implementation for the dictionary of Gabor functions on an interval. The implementation written in $C++$ can be used in many practical situations given its flexibility and generality.

1. Organization of the Report

In Section 2 we indicate the common setting for the Matching Pursuit (MP) algorithm and the theory of frames. Section 3 and subsections describe briefly the MP algorithm for Gabor dictionaries. Section 4 gives some of the mathematical details used in our software implementation of MP (included in *Wave++*); the subsections mention more details of the implementation. A simple demo can be found in ‘‘`wave++/demos/demoMP.cc`’’. The main points of our implementation are of a technical nature and the reader may need to read Sections 3 and 4 before being able to apreciate the main features of our contribution which are listed below.

• We use Gabor vectors on an interval. They require extra computations but are more useful for applications where boundary effects may be important.

• We offer two implementations, one based on the Fast Fourier Transform (faster) and a more flexible version that permits to select a set of scales for the windows in an arbitrary way.

• Using basic calculus we are able to optimize away one of the four parameters indexing the Gabor dictionary. This fact seems to have been unnoticed in previous implementations.

2. Introduction

In this section we follow [12]. Frame theory gives conditions for discretizing the windowed Fourier transform while retaining a complete and stable representation. The windowed Fourier transform of $f \in \mathbf{L^2}(\mathbb{R})$ is defined by

$$Sf(u,\xi) = \langle f, g_{u\xi} \rangle,$$

with

$$g_{u,\xi}(t) = g(t-u)\, e^{i\xi t}.$$

Research partially supported by NSERC.

Setting $\|g\| = 1$ implies that $\|g_{u,\xi}\| = 1$. A discrete windowed Fourier transform representation

$$\{Sf(u_n, \xi_k) = \langle f, g_{u_n,\xi_k} \rangle\}_{(n,k)\in\mathbb{Z}^2}$$

is complete and stable if $\{g_{u_n,\xi_k}\}_{(n,k)\in\mathbb{Z}^2}$ is a frame of $\mathbf{L^2}(\mathbb{R})$.

The time and frequency parameters (u, ξ) are discretized over a rectangular grid with time and frequency intervals of size u_0 and ξ_0. Let us denote

$$g_{n,k}(t) = g(t - nu_0)\, e^{ik\xi_0 t}.$$

The sampling intervals (u_0, ξ_0) must be adjusted to the time-frequency spread of g.

Window scaling Suppose that $\{g_{n,k}\}_{(n,k)\in\mathbb{Z}^2}$ is a frame of $\mathbf{L^2}(\mathbb{R})$ with frame bounds A and B. Let us dilate the window $g_s(t) = \frac{1}{\sqrt{s}} g(\frac{t}{s})$. It increases by s the time width of the Heisenberg box of g and reduces by s its frequency width. We thus obtain the same cover of the time-frequency plane by increasing u_0 by s and reducing ξ_0 by s. Let

$$g_{s,n,k}(t) = g_s(t - nsu_0)\, e^{ik\frac{\xi_0}{s}t}. \tag{1}$$

It follows that $\{g_{s,n,k}\}_{(n,k)\in\mathbb{Z}^2}$ satisfies the same frame inequalities as $\{g_{n,k}\}_{(n,k)\in\mathbb{Z}^2}$, with the same frame bounds A and B, by a change of variable $t' = ts$ in the inner product integrals.

Necessary conditions Daubechies [5] proved several necessary conditions on g, u_0 and ξ_0 to guarantee that $\{g_{n,k}\}_{(n,k)\in\mathbb{Z}^2}$ is a frame of $\mathbf{L^2}(\mathbb{R})$. We do not reproduce the proofs, but summarize the main results.

Theorem 1 (Daubechies)**.** *The windowed Fourier family $\{g_{n,k}\}_{(n,k)\in\mathbb{Z}^2}$ is a frame only if*

$$\frac{2\pi}{u_0\,\xi_0} \geq 1. \tag{2}$$

The frame bounds A and B necessarily satisfy

$$A \leq \frac{2\pi}{u_0\,\xi_0} \leq B, \tag{3}$$

$$\forall t \in \mathbb{R}\ ,\ A \leq \frac{2\pi}{\xi_0} \sum_{n=-\infty}^{+\infty} |g(t - nu_0)|^2 \leq B, \tag{4}$$

$$\forall \omega \in \mathbb{R}\ ,\ A \leq \frac{1}{u_0} \sum_{k=-\infty}^{+\infty} |\hat{g}(\omega - k\xi_0)|^2 \leq B. \tag{5}$$

The ratio $\frac{2\pi}{u_0\xi_0}$ measures the density of windowed Fourier atoms in the time-frequency plane. The first condition (2) ensures that this density is greater than 1 because the covering ability of each atom is limited. The inequalities (4) and (5) are proved in full generality by Chui and Shi [3]. They show that the uniform time translations of g must completely cover the time axis, and the frequency translations of its Fourier transform $\hat{g}$ must similarly cover the frequency axis.

Since all windowed Fourier vectors are normalized, the frame is an orthogonal basis only if $A = B = 1$. The frame bound condition (3) shows that this is possible only at the critical sampling density $u_0\xi_0 = 2\pi$.

Gaussian window The Gaussian window

$$g(t) = e^{-\frac{t^2}{2}} \tag{6}$$

has a Fourier transform $\hat{g}$ that is a Gaussian with the same variance. The time and frequency spreads of this window are identical. We therefore choose equal sampling intervals in time and frequency: $u_0 = \xi_0$. For the same product $u_0\xi_0$ other choices would degrade the frame bounds. If g is dilated by s then the time and frequency sampling intervals must become su_0 and $\frac{\xi_0}{s}$.

If the time-frequency sampling density is above the critical value: $\frac{2\pi}{u_0\,\xi_0} > 1$, then Daubechies [4] proves that $\{g_{n,k}\}_{(n,k)\in\mathbb{Z}^2}$ is a frame. When $u_0\xi_0$ tends to 2π, the frame bound A tends to 0. For $u_0\xi_0 = 2\pi$, the family $\{g_{n,k}\}_{(n,k)\in\mathbb{Z}^2}$ is complete in $\mathbf{L^2}(\mathbb{R})$, which means that any $f \in \mathbf{L^2}(\mathbb{R})$ is entirely characterized by the inner products $\{\langle f, g_{n,k}\rangle\}_{(n,k)\in\mathbb{Z}^2}$. Reference [15] presents some results for the discrete window Fourier transform computed with a discretized Gaussian window.

In the context of frames, analysis is performed by means of the frame algorithm, fast versions of this algorithm for periodized discrete Gabor functions is given in [8] and [14]. The expansion given by the frame algorithm actually achieves the *singular value decomposition*, i.e. solves a least squares problem for an overcomplete system of linear equations under the constrain that the sum of the squares of the coefficients is a minimum. This type of solution does not produce good compression of the signal and, therefore, precludes the possibility of useful synthesis for applications to denoising and *super resolution* (sparsity). This fact has been documented in [1], [2] and [7]. Attempts to remedy this problem can be found in [6]. Multi windows (or multi scale) approaches to this problem are discussed from different points of views in [16], [10] and [13] . The matching pursuit algorithm described next can be seen in the light of this problem, namely it offers a greedy algorithm to obtain sparse representations. It is a general algorithm that does not claim any (global) optimization property and does not use (explicitly) any particular structure offered by the setting of frames.

3. The MP Algorithm

In this section we review the essential aspects of the matching pursuit algorithm as discussed in [11]. Let $\mathbf{H}$ be a Hilbert space. We define a dictionary as a family $\mathcal{D} = \{g_\gamma : \gamma \in \Gamma\}$ of vectors in $\mathbf{H}$ such that $\|g_\gamma\| = 1$. Let $\mathbf{V}$ be the closed linear span of the dictionary vectors. We say that the dictionary is complete if $\mathbf{V} = \mathbf{H}$.

We want to compute a linear expansion of $f \in \mathbf{H}$ over a set of vectors selected from $\mathcal{D}$ in a way that describes f as simply as possible in terms of vectors from $\mathcal{D}$. In MP this is done by successive approximations of f by orthogonal projections on elements of $\mathcal{D}$; i.e., given $g_{\gamma_0} \in \mathcal{D}$ the vector f can be written as

$$f = \langle f, g_{\gamma_0}\rangle g_{\gamma_0} + Rf \tag{7}$$

where Rf is the residual vector left after approximating f in the direction of g_{γ_0}. Clearly g_{γ_0} is orthogonal to Rf, so

$$\|f\|^2 = |\langle f, g_{\gamma_0}\rangle|^2 + \|Rf\|^2. \tag{8}$$

To minimize $\|Rf\|$ we must maximize $|\langle f, g_{\gamma_0}\rangle|$ over $g_{\gamma_0} \in \mathcal{D}$. In general, it is only computationally feasible to find an "almost optimal" vector g_{γ_0} in the sense that

$$|\langle f, g_{\gamma_0}\rangle| = \max_{\gamma\in\Gamma_\alpha} |\langle f, g_\gamma\rangle| \geq \alpha \sup_{\gamma\in\Gamma} |\langle f, g_\gamma\rangle|, \tag{9}$$

where $\Gamma_\alpha \subseteq \Gamma$ and α is an optimality factor which satisfies $0 < \alpha \leq 1$. The construction of Γ_α depends on the dictionary; typically, if the dictionary is indexed by a set of continuous parameters Γ, then Γ_α will be a discrete grid of some sort in Γ.

We then continue the matching pursuit by induction. Let $R^0 f = f$. Suppose that we have computed $R^n f$, the residue of order n, for some $n \geq 0$. We then choose an element $g_{\gamma_n} \in \mathcal{D}_\alpha = \{g_\gamma : \gamma \in \Gamma_\alpha\}$ which closely matches the residue $R^n f$:

$$|\langle R^n f, g_{\gamma_n}\rangle| \geq \alpha \sup_{\gamma\in\Gamma} |\langle R^n f, g_\gamma\rangle|. \tag{10}$$

The residue $R^n f$ is decomposed as

$$R^n f = \langle R^n f, g_{\gamma_n}\rangle g_{\gamma_n} + R^{n+1} f, \tag{11}$$

which defines $R^{n+1} f$, the residue of order $n+1$. Since $R^{n+1} f$ is orthogonal to g_{γ_n},

$$\|R^n f\|^2 = |\langle R^n f, g_{\gamma_n}\rangle|^2 + \|R^{n+1} f\|^2. \tag{12}$$

Let us repeat this decomposition m times. Writing f in terms of the residues $R^n f$, $n = 0, 1, \ldots, m$ and applying (11) yields

$$f = \sum_{n=0}^{m-1} \langle R^n f, g_{\gamma_n}\rangle g_{\gamma_n} + R^m f. \tag{13}$$

The following theorem is fundamental to the MP algorithm [11].

Theorem 2. *If $\mathcal{D}$ is a complete dictionary and if $f \in \mathbf{H}$ then*

$$f = \sum_{k=0}^{\infty} \left\langle R^k f, g_{\gamma_k}\right\rangle g_{\gamma_k} \tag{14}$$

and

$$\|f\|^2 = \sum_{k=0}^{\infty} \left|\left\langle R^k f, g_{\gamma_k}\right\rangle\right|^2. \tag{15}$$

3.1. **Gabor Dictionaries.** Below we introduce a specific family $\mathcal{D}$ in $\mathbf{H} = L^2(\mathbb{R})$ consisting of Gabor functions, "windowed" trigonometric functions with infinite exponentially decreasing tails. We start by introducing continuous Gabor functions, having in mind that for the actual implementation all functions must be discretized.

3.1.1. *Aperiodic Continuous Gabor Functions.* As the window function $g(t)$ we use a Gaussian given by

$$g(t) = e^{-\pi t^2}. \tag{16}$$

notice that this is a bit different from (6), the reason for this is just to keep some compatibility with the notation and the discrete dictionary used in [11]. For any $\gamma = (s, u, v) \in \mathbb{R}^+ \times \mathbb{R}^2 = \Gamma$, let the Gabor function g_γ be given by

$$g_\gamma(t) = \frac{1}{\sqrt{s}} g\left(\frac{t-u}{s}\right) e^{ivt}. \tag{17}$$

The factor $1/\sqrt{s}$ normalizes $g_\gamma(t)$. Here $s > 0$ is called the *scale* of the function, u its *translation* and v its *frequency modulation*. $g_\gamma(t)$ is centered at the abscissa u and its energy is mostly concentrated in a neighborhood of u of size proportional

to s. Finite linear expansions of Gabor functions are dense in $L^2(\mathbb{R})$, hence this dictionary is complete.

In order to obtain a decomposition with real expansion coefficients when the signal $f(t)$ is real, we will use dictionaries of real time-frequency functions. For any $\gamma = (s, u, v)$ with $s > 0$ and for any phase $w \in [0, 2\pi)$, define

$$\begin{aligned} g_{(\gamma,w)}(t) &= \frac{K_{(\gamma,w)}}{2}(e^{iw} g_{(s,u,v)}(t) + e^{-iw} g_{(s,u,-v)}(t)) \\ &= \frac{K_{(\gamma,w)}}{\sqrt{s}} g\left(\frac{t-u}{s}\right) \cos(vt + w) \end{aligned} \tag{18}$$

where the positive constant $K_{(\gamma,w)}$ is determined by the condition $\|g_{(\gamma,w)}\| = 1$. The phase w which was hidden in the complex expansion coefficients now appears explicitly as a parameter of the real Gabor vectors. The dictionary of real time-frequency vectors is defined by $\mathcal{D}_R = \{g_{(\gamma,w)} : (\gamma, w) \in \Lambda = \Gamma \times [0, 2\pi)\}$. For convenience we use the notation $\beta = (\gamma, w)$. Matching pursuit performed with this dictionary decomposes any real signal $f(t)$ into the sum

$$f(t) = \sum_{n=0}^{\infty} \langle R^n f, g_{\beta_n} \rangle g_{\beta_n}(t) \tag{19}$$

where the indices $\beta_n = (s_n, u_n, v_n, w_n)$ are chosen by maximizing $|\langle R^n f, g_{\beta_n} \rangle|$ over Λ.

In practice this maximization is not feasible and an approximation scheme as indicated in equation (9) has to be used. Having in mind signal analysis as a primary application of MP, which in turn implies f being given as a discrete sample, we must resolve the additional problem of discretazing our dictionary. As we indicate in the next section we depart from previous implementations ([12]) that periodize the Gabor function.

4. Implementing MP With Discretized Gabor Dictionaries

4.0.2. *Discrete Gabor Functions on Interval.* Suppose f is given as a discrete sample of dimension `dim`. Then

$$f = (f[0], f[1], \ldots, f[\texttt{dim} - 1])$$

and we can create discretized samples of our dictionary elements by sampling them on integers $t \in \{0, 1, \ldots, \texttt{dim} - 1\}$. This way, the entire problem is translated into the euclidean space: we are approximating f by elements of an overcomplete basis of $\mathbb{R}^{\texttt{dim}}$, where the distance is measured by the standard l_2 norm.

Recall that the dictionary elements must be normalized, this time in the l_2 sense. The normalization constants used for continuous Gabor can be dropped out from the formulae. In other words, we will say that a real Gabor function $g_{(s,u,v,w)}$ is defined by

$$g_{(s,u,v,w)}(t) = \frac{g\left(\frac{t-u}{s}\right) \cos(vt + w)}{||g\left(\frac{\cdot - u}{s}\right) \cos(v\cdot + w)||}. \tag{20}$$

As before let $\gamma = (s, u, v)$. We define the following functions

$$P_\gamma(t) = g\left(\frac{t-u}{s}\right) \cos(vt), \qquad Q_\gamma(t) = g(\frac{t-u}{s}) \sin(vt). \tag{21}$$

Then

$$g_{(\gamma,w)} = \frac{P_\gamma \cos w - Q_\gamma \sin w}{\|P_\gamma \cos w - Q_\gamma \sin w\|}. \tag{22}$$

Therefore inner products will be given by

$$\langle f, g_{(\gamma,w)} \rangle = \frac{\langle f, P_\gamma \rangle \cos w + \langle f, Q_\gamma \rangle \sin w}{\|P_\gamma \cos w - Q_\gamma \sin w\|}. \tag{23}$$

Recall that at each step of the MP algorithm we seek $\max_{\gamma,w} |\langle R^n f, g_{(\gamma,w)} \rangle|$. The following proposition explains the way of finding an optimal w for given $R^n f$ and γ . As a result, the fourth parameter w is uniquely determined and the size of the dictionary depends on partitions in the first three parameters s, u and v only.

Proposition 1. *Given $f = (f[0], \ldots, f[\mathtt{dim}-1])$ and $\gamma = (s, u, v)$ denote $P = P_\gamma$, $Q = Q_\gamma$ and*

$$\begin{aligned} &a = \langle f, P \rangle \qquad b = \langle f, Q \rangle \\ &a_1 = a\|Q\|^2 - b\langle P, Q \rangle \quad b_1 = b\|P\|^2 - a\langle P, Q \rangle. \end{aligned} \tag{24}$$

Then an optimal w_0, i.e. one for which $max|\langle f, g_{(\gamma,w)} \rangle|$ is attained is given by:
(i) If $v \neq 0$ and $a_1 \neq 0$, then

$$tan\, w_0 = -\frac{b_1}{a_1} \quad and \quad \langle f, g_{(\gamma,w_0)} \rangle = \frac{aa_1 + bb_1}{\|Pa_1 + Qb_1\|}. \tag{25}$$

(ii) If $v = 0$, then

$$w_0 = 0, \ and \ \langle f, g_{(\gamma,w_0)} \rangle = \frac{a}{\|P\|}. \tag{26}$$

(iii) If $a_1 = 0$, then

$$w_0 = \frac{\pi}{2}, \ and \ \langle f, g_{(\gamma,w_0)} \rangle = -\frac{b}{\|Q\|}. \tag{27}$$

The proof reduces to maximazing function of one variable $h(x) = \langle f, g_{(\gamma,w)} \rangle^2$.

$$h(x) = \frac{(a - bx)^2}{\|P - Qx\|^2} = \frac{(a - bx)^2}{\|P\|^2 + \|Q\|^2 x^2 - 2\langle P, Q \rangle x} \tag{28}$$

where $x = \tan(w)$.

Define the discretized complex dictionary by $\mathcal{D}_\alpha = \{g_\gamma : \gamma \in \Gamma_\alpha\}$, a subset of the complex Gabor dictionary where the index set Γ_α is composed of all $\gamma = (a^j, pa^j \Delta u, ka^{-j} \Delta v)$, with $a = 2$, $\Delta u = \frac{1}{2}$, $\Delta v = \pi$, $0 < j < \log_2 N$, $0 \leq p < N2^{-j+1}$ and $0 \leq k < 2^{j+1}$.

In [11] it is proven (in the continuous case) that if the parameters (s, u, v) are discretized as indicated above there exists an $\alpha > 0$ such that the MP algorithm is sub-optimal with respect to α, i.e., (9) holds.

We present two implementations of MP based on discretized Gabor functions.

- Implementation A, which is slower but works with arbitrary dimensions of the input signal (e.g. dim doesn't have to be a power of 2). Also there is more flexibility in choosing partitions.
- Implementation B, which requires that `dim` is a power of 2 and is restricted to the dictionary as described above. The algorithm is based on FFT (Fast Fourier Transform) and is very fast.

4.1. **Basic Classes.** A reader will notice that we do not use encapsulation in our code. The reason is very simple: we are interested in minimizing overhead. The MP algorithm does an exhaustive search over a large set, so it is of vital interest to optimize performance wherever possible. Besides, this implemntation of MP is not a comercial software, it is aimed at researchers and therefore we do not feel any special need to hide any portions of the code.

We use two typedefs:

- `typedef real double;`
- `typedef integer long;`

Naturally, users can change these definitions according to their concerns for memory usage and precision requirements.

We use the basic class `Interval` from *Wave++*([9]). This class is designed to represent vectors of real numbers, with indices being any set of consecutive integers. A particularly convenient utility member function is `Set`. `Set` is used in constructors, but it can also be used for redefining an existing interval. One typical use of `Interval` is for storing an input signal f. As a rule, input signals are indexed from `0` to $\texttt{dim}-1$. This rule reflects the idea that we can think of an input signal f as a sample of some function evaluated at times $\texttt{t}=0, \texttt{t}=1, \ldots, \texttt{t}=\texttt{dim}-1$.

`RealGabor` is a class which is designed to encode real Gabor functions. Its data members s, u, v and w have obvious interpretations as scale, center, frequency and phase. Member function `evaluate(real t)` returns $g_{(s,u,v,w)}(t) = g\left(\frac{t-u}{s}\right)\cos(vt+w)$. When needed, data member `Sample` is set by calling member function `createSample(Interval I)` which first samples $g_{(s,u,v,w)}$ using *evaluate* on elements of I, and then normalizes the sample.

4.2. **Implementation A.** This implementation can accomodate any partition in s, u, v satisfying the following conditions:

- s can take any desired values, say $s[j]$ for $1 \le j \le n$. $s[j]$'s need not be integers.
- Once we decided on what are the leftmost and rightmost values of u, denoted `lmu` and `rmu` respectively, then for any fixed $s[j]$ partition in u is defined by a constant increment $du = du[j]$, which depends on $s[j]$ only. More precisely, $u \in \{\texttt{lmu} + p du \mid p \in \mathbb{Z},\ 0 \le p,\ \texttt{lmu} + p\,du \le \texttt{rmu}\}$.
- The leftmost value of v is set to be 0 always. Once we decided what is the rightmost value of v, denoted `rmv`, then for a fixed $s[j]$ partition in v is defined by a constant increment $dv = dv[j]$, which depends on $s[j]$ only. More precisely, $v \in \{k dv \mid k \in \mathbb{Z},\ 0 \le k,\ k\,dv < \texttt{rmv}\}$.

We encode all this information in a class named `Partition`. Its only constructor `Partition(integer dim, real a)` is designed to create the following partition suitable for signals of dimension `dim` :

- $s[j] = a^j$, for $1 \le j \le n$, where n is the largest integer power of a such that $a^n \le \texttt{dim}$
- $du[j] = s[j]/2$ and $v[j] = \pi/s[j]$
- $\texttt{lmu} = 0$, $\texttt{rmu} = \texttt{dim} - 1$ and $\texttt{rmv} = 2\pi$.

For example, Mallat's partition is obtained by using this constructor with $a = 2$. Users can decide to define a completely different partition as long as it complies with rules stated above. In this case they would have to set all data members "by hand".

We implement the MP algorithm using two functions: `getOptimalGabor` and `RunGaborMP`. The first one is the core function: given $R^n f$ it searches the entire dictionary for a Gabor function which has a maximal inner product with $R^n f$. The second function is a wrapper function which runs MP algorithm by repeated calls to getOptimalShiftGabor.

Input parameters of `getOptimalShiftGabor` are: signal f and partition `Part`. Its output parameters are real Gabor function G and scalar `coef`. On output G receives the optimal Gabor function from the dictionary defined by $Part$ and `coef` $= \langle G, f \rangle$. The algorithm in its most basic form performs the search in the following way:

`coef = 0`
`for` $j = 1;\ j \leq n;\ j++$
$s = s[j]$
`for` $v = 0;\ v \leq$ `rmv;` $v += dv[j]$
`for` $u =$ `lmu;` $u \leq$ `rmu;` $u += du[j]$
`set` $P = P_{(s,u,v)}$ `and` $Q = Q_{(s,u,v)}$
`calculate` a, b, a_1, b_1 `according to (24)`
`find optimal phase w using Proposition and`
`calculate` $product = \langle f, g_{(s,u,v,w)} \rangle$
`if` $|product| >$ `coef`
`set coef` $= product$ `and G.set`(s, u, v, w)

The the actual code of the function `getOptimalGabor` is quite unreadable, due to the fact that we completely avoided calls to the sine and cosine functions. All cosine and sine evaluations corresponding to changes in frequency v are done through a carefully designed updating scheme. Just for illustration purposes let us look at the level of the inner most loop where we need to evaluate

$$\begin{aligned} P_{(s,u,v)}[t] &= g\left(\frac{t-u}{s}\right)\cos(vt) \\ &= g\left(\frac{t-p\,du}{s}\right)\cos(k\,dv\,t) \end{aligned} \tag{29}$$

where p and k are integers indicating how many passes through u and v loops respectively have been made. Obviously

$$\cos(k\,dv\,t) = \cos((k-1)\,dv\,t)\cos(dv\,t) - \sin((k-1)\,dv\,t)\sin(dv\,t)$$

so its value can be updated from the preceeding position $k-1$ of v loop.

We used another trick regarding changes in translation values u. For this purpose we need the $du[j]$'s to be integers, eventhough they are defined as reals in the class `Partition`. So `getOptimalGabor` will use integer casting on $du[j]$'s (which is essentialy flooring). Once this is done we can make use of the following identities:

$$\begin{aligned} P_{(s,c+\delta,v)}[t] &= P_{(s,c,v)}[t-\delta]\cos(v\delta) - Q_{(s,c,v)}[t-\delta]\sin(v\delta) \\ Q_{(s,c+\delta,v)}[t] &= P_{(s,c,v)}[t-\delta]\sin(v\delta) + Q_{(s,c,v)}[t-\delta]\cos(v\delta) \end{aligned} \tag{30}$$

Therefore, for fixed s and v, we can evaluate only one P and Q, say $P_{(s,c,v)}$ and $Q_{(s,c,v)}$, where c is some fixed value (we took $c =$ `dim`$/2$). Then for arbitrary u, set

$\delta = u - c$ and then

$$\begin{aligned}(31)\quad \langle f, P_{(s,u,v)}\rangle &= \cos(v\delta)\sum_t f[t]P_{(s,c,v)}[t-\delta] - \sin(v\delta)\sum_t f[t]Q_{(s,c,v)}[t-\delta],\\ \langle f, Q_{(s,u,v)}\rangle &= \sin(v\delta)\sum_t f[t]P_{(s,c,v)}[t-\delta] + \cos(v\delta)\sum_t f[t]Q_{(s,c,v)}[t-\delta].\end{aligned}$$

Obviously, the δ's must be integers.

Now let us examine the wrapper function `RunGaborMP`. Its input parameters are: partition `Part`, input signal f, maximal number of iterations `max_iter` and precision ϵ. Its output parameters are: vector of real Gabor functions `G`, vector of scalars `Gcoef` corresponding to elements of `G`, output signal f_{approx} and residual Rf. `RunGaborMP` will keep calling the core function `getOptimalGabor` and after each such call, G is appended with one more Gabor function and `Gcoef` is appended with the corresponding coefficient. On the output we will have

$$\begin{aligned}(32)\qquad f_{approx} &= \sum_{i=0}^{n} \texttt{Gcoef[i]} * \texttt{G[i]},\\ Rf &= f - f_{approx},\end{aligned}$$

where n denotes the number of iterrations performed. If ϵ-precision is achived in $n < max\_iter$ steps, i.e. $\|Rf\| < \epsilon$, then the function returns early, otherwise $n = max\_iter$.

It should be noted that Gabors comprising G have their samples set after they come fresh from `RunShiftGabor`. So if users want to watch how MP progressed from iterration to iterration, they can plot intermediate steps. For example, using `Interval` arithmetics we can find j-th approximation of f as $\sum_{i=0}^{j} G[i].Sample * Gcoef[i]$.

4.3. **Implementation B.** We may also implement a fast version of the Matching Pursuit Algorithm on an interval by taking advantage of the Fast Fourier Transform (FFT) implementation of the Discrete Fourier Transform (DFT). The increase in speed is one order of magnitude: for `dim` data points the above algorithm takes order of magnitude $\texttt{dim}^2 \log(\texttt{dim})$ calculatiuons, while with a fast implementation it takes $\texttt{dim} \log^2(\texttt{dim})$ calculations. The price to be paid, however, is that we loose much of the flexibility in choosing the dictionary.

In order to be take advantage of the FFT algorithm we must first have that `dim` is a power of 2, $\texttt{dim} = 2^J$ for some J. Then the only allowable values for the scale parameter s are $s = 2^j,\ j = 1, 2, \ldots, J$. For a given value of s the allowable values of the frequency parameter, v are $v = 2\pi k/N$, where $N = 8s$ when $8s \le \texttt{dim}$, $N = \texttt{dim}$ otherwise, and $k = 0, 1, \ldots N-1$. The values of the translation parameter, u may be chosen as desired; the fast implementation uses the same convention as the more general implementation: $u = ps/2$, where $0 \le p \le 2\texttt{dim}/s$.

The algorithm begins by first choosing $s = 2^j$ and then selecting an allowable u. Having done this define

$$P_k(j) = g\left(\frac{j-u}{s}\right)\cos\left(\frac{2\pi jk}{N}\right),$$

$$Q_k(j) = g\left(\frac{j-u}{s}\right)\sin\left(\frac{2\pi jk}{N}\right).$$

This is just $P_\gamma(t)$ and $Q_\gamma(t)$ of equation (21) with $t = j,\ \gamma = (2^j, u, 2\pi jk/N)$. For each value $k = 0, 1, \ldots N-1$ we may use the FFT to calculate

$$\langle f, P_k\rangle,\ \langle f, Q_k\rangle,\ \langle P_k, P_k\rangle,\ \langle Q_k, P_k\rangle,\ \langle Q_k, Q_k\rangle,$$

for all values of k.

The main idea is that the above quantities are just the real and imaginary parts of the DFT of carefully chosen inputs. With a little finesse we may take advantage in the gain in speed offered by the FFT algorithm. There are, however some technicalities to be dispensed with before we may actually use the FFT

Given the effective support of $g(t/s)$, i.e., that $g \approx 0$ when $|t/s| > 4$, we must distinguish two cases: Case I when $8s > \texttt{dim}$ and Case II when $8s \leq \texttt{dim}$. We must also take into account the shift parameter u and its relation to the right and left hand end points of the interval. Thus case II has three subcases: (a) $0 \leq u \leq 4s$, (b) $4s < u \leq \texttt{dim} - 4s$ and (c) $u > \texttt{dim} - 4s$.

We recall the formula for the DFT of N data points $\{x_j\}_{j=0}^{N-1}$. For $k = 0, 1, \ldots, N-1$ define X_k by

$$X_k = \sum_{j=0}^{N-1} x_j \exp(-2\pi ijk/N) = \sum_{j=0}^{N-1} x_j \cos(2\pi jk/N) - i \sum_{j=0}^{N-1} x_j \sin(2\pi jk/N)$$

Let us make the notation $\text{Output.Re}[k] = Re(X_k)$, $\text{Output.Im}[k] = Im(X_k)$.

Case I is computationally the simplest. The assumption that $8s > \texttt{dim}$ means that the support of g is effectively the whole interval. If we take as our input $x_j = f(j)g(\frac{j-u}{s})$ and have $N = \texttt{dim}$, then for $k = 0, 1, \ldots N-1$.

$$\begin{aligned} \langle f, P_k\rangle, &= \text{Output.Re}[k], \\ \langle f, P_k\rangle, &= -\text{Output.Re}[k]. \end{aligned}$$

By taking the input to the FFT to be $x_j = g(\frac{j-u}{s})^2$ and using the double angle formulas for sines and cosines we have, with $C = \text{Input.Re}[0]$,

$$\begin{aligned} \langle P_k, P_k\rangle &= \frac{1}{2}\left(C + \text{Output.Re}[2k]\right), \\ \langle Q_k, Q_k\rangle &= \frac{1}{2}\left(C - \text{Output.Re}[2k]\right), \\ \langle Q_k, P_k\rangle &= -\frac{1}{2}\,\text{Output.Im}[2k], \end{aligned}$$

for $k = 0, 1, \ldots, \frac{\texttt{dim}}{2} - 1$, and

$$\begin{aligned} \langle P_k, P_k\rangle &= \langle P_{k-\frac{\texttt{dim}}{2}}, P_{k-\frac{\texttt{dim}}{2}}\rangle, \\ \langle Q_k, Q_k\rangle &= \langle Q_{k-\frac{\texttt{dim}}{2}}, Q_{k-\frac{\texttt{dim}}{2}}\rangle, \\ \langle Q_k, P_k\rangle &= \langle Q_{k-\frac{\texttt{dim}}{2}}, P_{k-\frac{\texttt{dim}}{2}}\rangle, \end{aligned}$$

when $k = \frac{\texttt{dim}}{2}, \ldots, \texttt{dim} - 1$.

In case II, $8s \leq \texttt{dim}$, the effective support of the Gaussian $g(t/s)$ is smaller than the interval. We take advantage of the fewer number of data points and let $N = 8s$ in the FFT algorithm. The three subcases account for the center of the Gaussian, which is the main technicality in this algorithm.

Subcase (a), $0 \leq u \leq 4s$, is exactly as the previous case; we do everything as before with $\texttt{dim}$ replaced by $N = 8s$. In this case the support of $g(\frac{t-u}{s})$ still

contains the left hand endpoint of the interval and so the FFT algorithm may be used exactly as before.

In subcase (b), the lefthand endpoint of the interval is no longer in the support of the Gaussian, and in order to use the DFT we must make a translation of indices; let the input to the FFT be $x_j = f(j+u-4s)g(\frac{j-4s}{s})$. A modulation in the frequency domain takes place; let $\alpha_k = 2\pi(u-4s)k/N$. Then,

$$\begin{aligned}\langle f, P_k\rangle, &= \cos(\alpha_k)\,\text{Output. Re}[k] + \sin(\alpha_k)\,\text{Output. Im}[k],\\ \langle f, P_k\rangle, &= \sin(\alpha_k)\,\text{Output. Re}[k] - \cos(\alpha_k)\,\text{Output. Im}[k].\end{aligned}$$

If we now let the input be $x_j = g(\frac{j-4s}{s})^2$, and take C as before, with $k = 0, 1, \ldots, \frac{N}{2}-1$,

$$\begin{aligned}\langle P_k, P_k\rangle &= \frac{1}{2}\left(C + \cos(\alpha_{2k})\,\text{Output. Re}[2k] + \sin(\alpha_{2k})\,\text{Output. Im}[2k]\right),\\ \langle Q_k, Q_k\rangle &= \frac{1}{2}\left(C - \cos(\alpha_{2k})\,\text{Output. Re}[2k] - \sin(\alpha_{2k})\,\text{Output. Im}[2k]\right),\\ \langle Q_k, P_k\rangle &= \frac{1}{2}\left(-\cos(\alpha_{2k})\,\text{Output. Im}[2k] + \sin(\alpha_{2k})\,\text{Output. Re}[2k]\right).\end{aligned}$$

And for $k = \frac{N}{2}, \ldots, N-1$ we use the above formulas with k replaced by $k - \frac{N}{2}$ as before.

Subcase (c) sees the support of the Gaussian going past the right hand endpoint of the interval. To make up for this we only have to pad with zeros. For $j = 0, 1, \ldots, \texttt{dim} - 1 - u + 4s$ we let x_j be defined as before. Otherwise $x_j = 0$.

References

[1] S. Chen and D. L. Donoho, *Atomic decomposition by basis pursuit.* In SPIE International Conference on wavelets, San Diego, July 1995.

[2] S. Chen, D. L. Donoho and M.A.Saunders *Atomic decomposition by basis pursuit.* Technical report 479, Stanford University, 1996.

[3] C. K. Chui and X. Shi *Inequalities of Littlewood-Paley type for frames and wavelets*, SIAM J. Math. Anal.**24**(1): 263-277, January 1993.

[4] I. Daubechies, *The Wavelet Transform, Time-Frequency Localization and Signal Analysis*, IEEE Trans. Info. Theory, **36**(5): 961-1005, September 1990.

[5] I. Daubechies, *Ten Lectures on Wavelets*, SIAM, Philadelphia, PA, 1992.

[6] I. Daubechies, H.J.Landau and Z. Landau, *Gabor time-frequency lattices and the Wexler-Raz identity*, The Journal of Fourirer Analysis and Applications. Vol. 1, Number 4, 1995.

[7] L. Donoho and X.Huo *Uncertainty principles and ideal atomic decompositions.* Preprint.

[8] K. Gröchenig, *Acceleration of the frame algorithm* IEEE Trans. Signal Proc., 41(12):3331-3340, December 1993.

[9] S.E.Ferrando, L.A.Kolasa and N. Kovačević *C++Wavelets: A User's Guide.* Included in distribution of *wave++*.

[10] A.E.J.M. Janssen, *The duality condition for Weyl-Heisenberg frames* in Gabor Analysis and Algorithms, edited by H.G. Feichtinger and T. Strohmer. Birkhäuser, 1998.

[11] S. Mallat and Z. Zhang, *Matching pursuit with time-frequency dictionaries*, IEEE Transactions on Signal Processing **41**:12 (1993), 3397–3415.

[12] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, Boston, MA, 1998.

[13] S. Qian and D. Chen, *Joint Time-Frequncy Analysis: Method and Application*, Prentice Hall, Englewood Cliffs, NJ, 1996.

[14] T. Strohmer, *Numerical algorithms for discrete Gabor expansions* in Gabor Analysis and Algorithms, edited by H.G. Feichtinger and T. Strohmer. Birkhäuser, 1998.

[15] M. Zibulski and Y. Zeevi *Frame analysis of the discrete Gabor-scheme analysis*, IEEE Transactions on Signal Processing **42**:942–945, April 1994.

[16] M. Zibulski and Y. Zeevi *Multi-window Gabor-type transform for signal representation and analysis.* In SPIE Proc. Wavelet Applications in Signal and Image Processing I II, pages 116-127, San Diego, CA, July 1995.

Department of Mathematics, Physics and Computer Science, Ryerson Polytechnic University, Toronto, Ontario M5B 2K3, Canada.

E-mail address: ferrando@acs.ryerson.ca

E-mail address: lkolasa@acs.ryerson.ca

E-mail address: natasak@home.com