ACCELERATING MATCHING PURSUIT FOR MULTIPLE TIME-FREQUENCY DICTIONARIES

Zdeněk Průša, Nicki Holighaus and Peter Balazs *

Acoustics Research Institute
Austrian Academy of Sciences
Vienna, Austria
zprusa@kfs.oeaw.ac.at,nicki.holighaus@oeaw.ac.at,peter.balazs@oeaw.ac.at

ABSTRACT

Matching pursuit (MP) algorithms are widely used greedy methods to find K-sparse signal approximations in redundant dictionaries. We present an acceleration technique and an implementation of the matching pursuit algorithm acting on a multi-Gabor dictionary, i.e., a concatenation of several Gabor-type time-frequency dictionaries, consisting of translations and modulations of possibly different windows, time- and frequency-shift parameters. The proposed acceleration is based on pre-computing and thresholding inner products between atoms and on updating the residual directly in the coefficient domain, i.e., without the round-trip to the signal domain. Previously, coefficient-domain residual updates have been dismissed as having prohibitive memory requirements. By introducing an approximate update step, we can overcome this restriction and greatly improve the performance of matching pursuit at a modest cost in terms of approximation quality per selected atom. An implementation in C with Matlab and GNU Octave interfaces is available, outperforming the standard Matching Pursuit Toolkit (MPTK) by a factor of 3.5 to 70 in the tested conditions. Additionally, we provide experimental results illustrating the convergence of the implementation.

1. INTRODUCTION

Optimal K-sparse approximation of a signal $\mathbf{x} \in \mathbb{R}^L$ in an overcomplete dictionary of P normalized atoms (vectors) $\mathbf{D} = [\mathbf{d}_0|\mathbf{d}_1|\dots|\mathbf{d}_{P-1}] \in \mathbb{C}^{L\times P}$, $\|\mathbf{d}_p\|_2 = 1$ or, similarly, finding the minimal number of atoms that achieve a given error tolerance $\|\mathbf{x} - \mathbf{D}\mathbf{c}\|_2 \leq E$, are NP-hard problems [1]. Both problems can be (sub-optimally) solved by employing greedy *matching pursuit* (MP) algorithms. The only difference is the choice of the stopping criterion. It has been shown that the basic version of MP [2] achieves an exponential approximation rate [1, 3, 4, 5]. In practice, without imposing any structure on the dictionary, the effectiveness of MP and its variants, see, e.g., [6, 7, 8, 9], quickly deteriorates when the problem dimensionality increases; either by increasing the input signal length L or the size of the dictionary P. Even with structured dictionaries, and using fast algorithms in place of matrix operations, a naive implementation can still be prohibitively inefficient; even decomposing a short audio clip can take hours.

* This work was supported by the Austrian Science Fund (FWF): Y 551–N13 and I 3067–N30. The authors thank Bob L. Sturm for sharing his thoughts on the subject in a form of a blog *Pursuits in the Null Space*.

An overview of greedy algorithms, a class of algorithms MP falls under, can be found in [10, 11] and in the context of audio and music processing in [12, 13, 14]. Notable applications of MP algorithms in the audio domain include analysis [15], [16], coding [17, 18, 19], time scaling/pitch shifting [20] [21], source separation [22], denoising [23], partial and harmonic detection and tracking [24].

We present a method for accelerating MP-based algorithms acting on a single Gabor-type time-frequency dictionary or on a concatenation of several Gabor dictionaries with possibly different windows and parameters. The main idea of the present acceleration technique is performing the residual update in the coefficient domain while exploiting the locality of the inner products between the atoms in the dictionaries and dismissing values below a user definable threshold. The idea of performing the residual update in the coefficient domain using inner products between the atoms in fact dates back to the original work by Mallat and Zhang [2, Appendix E], but has always been dismissed as being prohibitively storage expensive, as it generally requires storing pairwise inner products between all dictionary elements. By exploiting the Gabor structure of the dictionaries and dismissing insignificant inner products, it becomes feasible to store all significant inner products in a lookup table and avoid atom synthesis and the residual reanalysis in every iteration of MP as it is usually done in practice. The size of the lookup table as well as the cost of computing it are independent of the signal length and depend only on the parameters of the Gabor dictionaries.

A freely available implementation in C (compatible with C99 and C++11), can be found in the backend library of the Matlab/GNU Octave Large Time-Frequency Analysis Toolbox (LTFAT, `http://ltfat.github.io`) [25, 26] available individually at `http://ltfat.github.io/libltfat`[1].

We compare the performance of the proposed implementation with the Matching Pursuit Toolkit (MPTK) [27], the de-facto standard implementation of several MP based algorithms. By specializing on multi-Gabor dictionaries, the presented algorithm is able to significantly outperform MPTK.

This manuscript is a shortened version of the paper [28], which provides an extended discussion of the proposed algorithm including a worst case analysis of its convergence properties and more details about the provided implementation.

[1] Confer to `http://ltfat.github.io/libltfat/group__multidgtrealmp.html` or `http://ltfat.github.io/doc/gabor/multidgtrealmp.html`) for more information on the library and its Matlab interface.

2. PRELIMINARIES

Matrices and column vectors are denoted with capital and lowercase bold upright letters, e.g., $\mathbf{M}, \mathbf{x}$, respectively. Conjugate transpose is denoted with a star superscript, $(\mathbf{x}^*, \mathbf{M}^*)$, scalar variables with a capital or lowercase italics letter s, S. A single element of a matrix or a vector is selected using round brackets $\mathbf{M}(m,n)$, $\mathbf{x}(l)$. The index is always assumed to be applied modulo vector length (or matrix size in the respective dimension) such that $\mathbf{x}(l) = \mathbf{x}(l+kL)$ for $l = 0,\dots,L-1$ and $k \in \mathbb{Z}$. Moreover, we will use two indices and subscript for vectors such that $\mathbf{c}(m,n)_M = \mathbf{c}(m+nM)$ in order to transparently "matrixify" a vector. Sub-vectors and sub-matrices are selected by an index set denoted by a calligraphic letter e.g. $\mathbf{x}(\mathcal{P})$ and the m-th row or n-th column of a matrix $\mathbf{M}$ are selected using the notation $\mathbf{M}(m,\bullet)$ and $\mathbf{M}(\bullet,n)$, respectively. Scalar-domain functions used on matrices or vectors are applied element-wise e.g. $|\mathbf{x}|^2(l) = |\mathbf{x}(l)|^2$.

The Euclidean inner product of two vectors in $\mathbb{C}^L$ is $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{y}^*\mathbf{x} = \sum_{l=0}^{L-1} \mathbf{x}(l)\overline{\mathbf{y}(l)}$, where the overline denotes complex conjugation, such that the 2-norm of a vector is defined by $\|\mathbf{x}\|_2^2 = \langle \mathbf{x}, \mathbf{x} \rangle$.

2.1. Multi-Gabor Dictionaries

A Gabor dictionary $\mathbf{D}_{(\mathbf{g},a,M)}$ generated from a window $\mathbf{g} \in \mathbb{R}^L$, $\|\mathbf{g}\|_2 = 1$, time shift a and a number of modulations M is given as

$$\mathbf{D}_{(\mathbf{g},a,M)}(l, m+nM) = \mathbf{g}(l-na)\mathrm{e}^{\mathrm{i}2\pi m(l-na)/M} \qquad (1)$$

for $l = 0,\dots,L-1$ and $m = 0,\dots,M-1$ for each $n = 0,\dots,N-1$, where $N = L/a$ is the number of window time shifts and the overall number of atoms is $P = MN$. Hence, the redundancy of a dictionary is $P/L = M/a$. A multi-Gabor dictionary consists of W Gabor dictionaries, i.e.,

$$\left[\mathbf{D}_{(\mathbf{g}_1,a_1,M_1)} \middle| \mathbf{D}_{(\mathbf{g}_2,a_2,M_2)} \middle| \dots \mathbf{D}_{(\mathbf{g}_W,a_W,M_W)}\right] \qquad (2)$$

and we will also use a shortened notation $\mathbf{D}_w = \mathbf{D}_{(\mathbf{g}_w,a_w,M_w)}$. Generally, a_w, M_w need only be divisors of L. Due to technical reasons explained in Sec. 3.1, however, efficiency of the presented algorithm depends on the pairwise *compatibility* of a_u, a_v and M_u, M_v, implying some restrictions of the dictionary parameters. In the following, we focus on the optimal setting, in which every pair a_u, a_v is divisible by $a_{\min} = \min\{a_u, a_v\}$ and, similarly, every pair of M_u, M_v should divide $M_{\max} = \max\{M_u, M_v\}$ and each M_w/a_w should be a positive integer. While not strictly necessary, such setting is commonly used in practice and leads to the most efficient implementation.

2.2. Matching Pursuit – MP

The MP algorithm attempts to find a K-term approximation of a given signal $\mathbf{x}$ by elements from the dictionary, i.e. $\mathbf{x} \approx \mathbf{x}_K = \sum_{l=1}^{K} c_l \mathbf{d}_l$. To do so, MP iteratively selects from the normalized dictionary the element which maximizes the inner product with the current residual elements $\mathbf{r}_k = \mathbf{x} - \mathbf{x}_k$, where $\mathbf{x}_0$ is the zero vector and, with $p_{\max} = \operatorname{argmax}_p |\langle \mathbf{r}_k, \mathbf{d}_p \rangle|$, $\mathbf{x}_k = \mathbf{x}_{k-1} + \langle \mathbf{r}_k, \mathbf{d}_{p_{\max}} \rangle \mathbf{d}_{p_{\max}}$. This is equivalent to adding the *largest orthogonal projection of the current residual on a single dictionary element* to the current approximation, such that the approximation error is $E_{k+1} = \|\mathbf{r}_{k+1}\|_2^2 = \|\mathbf{r}_k\|_2^2 - |\langle \mathbf{r}_k, \mathbf{d}_{p_{\max}} \rangle|^2$, for $k \geq 0$.

The procedure is repeated until the desired approximation error is achieved or alternatively some other stopping criterion is met e.g. a sparsity or a selected inner product magnitude limits are reached. It is known that the matching pursuit (MP) algorithm and its derivatives can benefit from pre-computing inner products between the atoms in the dictionary $\mathbf{G}(k,j) = \langle \mathbf{d}_j, \mathbf{d}_k \rangle$ i.e. from pre-computing the Gram matrix $\mathbf{G} = \mathbf{D}^*\mathbf{D} \in \mathbb{C}^{P \times P}$. With $\boldsymbol{\varrho}_k = \mathbf{D}^*\mathbf{r}_k$ denoting the coefficient-domain residual, the residual update step can be written as ([29, Ch. 12])

$$\boldsymbol{\varrho}_{k+1} = \boldsymbol{\varrho}_k - \mathbf{c}(p_{\max})\mathbf{G}(\bullet, p_{\max}). \qquad (3)$$

This modification has the advantage of removing the necessity of synthesizing the residual and recomputing the inner product in the selection step. On the other hand, such approach is usually dismissed as impractical in the literature due to the high memory requirements for storing the Gram matrix. We will show that this is not the case for multi-Gabor dictionaries, see Section 3. Formally, the coefficient-domain matching pursuit algorithm is summarized in Alg. 1. The stopping criterion may contain several conditions, and the algorithm terminates if any of these conditions is met.

Input: Input signal $\mathbf{x}$, dictionary Gram matrix $\mathbf{G} = \mathbf{D}^*\mathbf{D}$
Output: Solution vector $\mathbf{c}$
Initialization: $\mathbf{c} = \mathbf{0}$, $\boldsymbol{\varrho}_0 = \mathbf{D}^*\mathbf{x}$, $E_0 = \|\mathbf{x}\|_2^2$, $k = 0$
while *Stopping criterion not met* **do**

1. **Selection:** $p_{\max} \leftarrow \operatorname{argmax}_p |\boldsymbol{\varrho}_k(p)|$
2. **Update:**
   (a) **Solution:** $\mathbf{c}(p_{\max}) \leftarrow \mathbf{c}(p_{\max}) + \boldsymbol{\varrho}_k(p_{\max})$
   (b) **Error:** $E_{k+1} \leftarrow E_k - |\boldsymbol{\varrho}_k(p_{\max})|^2$
   (c) **Residual:** $\boldsymbol{\varrho}_{k+1} \leftarrow \boldsymbol{\varrho}_k - \boldsymbol{\varrho}_k(p_{\max})\mathbf{G}(\bullet, p_{\max})$

$k \leftarrow k+1$
end

Algorithm 1: Coefficient-Domain Matching Pursuit

3. FAST APPROXIMATE COEFFICIENT-DOMAIN RESIDUAL UPDATE

The Gram matrix $\mathbf{G}$ contains the pairwise inner products of the dictionary elements. For a Gabor dictionary $\mathbf{D} = \mathbf{D}_w$, it is highly structured. It takes the form of a *twisted convolution* [30] matrix with a fixed kernel $\mathbf{h} = \mathbf{G}(\bullet, 0) = \mathbf{D}^*\mathbf{g} \in \mathbb{C}^{MN}$; a coefficient vector consisting of inner products of the window with all its possible time and frequency shifts. When combined with the time-frequency localization of the individual atoms, we see that the matrixified version $\mathbf{h}(\bullet,\bullet)_M \in \mathbb{C}^{M \times N}$ of the kernel is essentially supported around the origin, provided the window $\mathbf{g}$ is low-pass and centered at 0. Hence, the significant values of $\mathbf{h}$ can be precomputed and stored at a cost independent of the total signal length L. If $p_{\max} = k + jM$, we can represent the coefficient-domain residual update step by

$$\begin{aligned} &\boldsymbol{\varrho}_{k+1}(m-k, n-j)_M \\ &= \boldsymbol{\varrho}_k(m-k, n-j)_M - \boldsymbol{\varrho}_k(p_{\max})\mathbf{h}(m,n)_M e^{i2\pi k \frac{a}{M} n}. \end{aligned} \qquad (4)$$

Whenever $\mathbf{g}$ is a localized, low-pass window, then for any $\epsilon > 0$, there is a minimal area $\mathcal{M} \times \mathcal{N} \subset \{0,\dots,M-1\} \times \{0,\dots,N-$

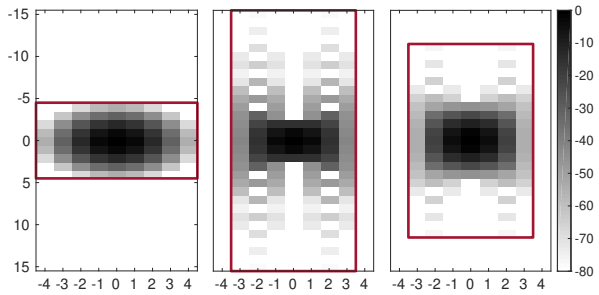

Figure 1: Examples of abs. values of truncated kernels for (left) Gaussian (9×9), (middle) Hann (31×7) and (right) Blackman (23×7) windows.

$1\}$, symmetric around the origin, such that $\mathbf{h}(m,n)_M < \epsilon$ for all $(m,n) \notin \mathcal{M} \times \mathcal{N}$. Clearly, the size of $\mathcal{M} \times \mathcal{N}$ depends on the shape of the window $\mathbf{g}$ and on the the time and frequency parameters a, M. Provided that L is at least twice as long as the essential support of $\mathbf{g}$ it is, however, independent of the signal length L. Values of $\mathbf{h}$ outside this area are zero or at least negligible, if ϵ is small and can thus be discarded. The choice of the threshold ϵ must be considered a trade-off between accuracy and efficiency.

The kernel size directly determines the number of complex multiplications and additions required to perform the entire residual update step and, obviously, also the memory requirements to store the kernel. Examples of abs. values of kernels for several windows using time shift $a = 512$ and $M = 2048$ frequency bins are depicted in Fig. 1. The values are in dB relative to the maximum element with 0 dB. Values below -80 dB were cropped and are not used in the residual update. The threshold selection is a trade-off between faster updates (higher threshold) or less requirement for resets (lower threshold). The idea of truncating the kernel $\mathbf{h}$ originates from Le Roux et al. [31] who used it for replacing the operation of the (cross) Gram matrix in an iterative scheme dealing with magnitude-only reconstruction. When inspecting formula (4), it is obvious that for a fixed frequency position k the modulation by $2\pi ka/M$ radians is performed on all rows of the kernel independently. Moreover, the modulation frequencies are $\mathrm{lcm}(a, M)/a$ periodic in k and, therefore, all unique complex exponentials can be tabulated and stored. In the best case when M is integer divisible by a, the memory requirements are equal to storing M/a additional rows of the kernel. The cost of applying the modulation during the residual update step is one complex multiplication per kernel column.

For a multi-Gabor dictionary, we must not only store the kernel of every individual dictionary, but the cross-kernels between dictionaries as well, which can be done similarly.

3.1. Pre-computing Cross-Kernels Between Dictionaries

The Gram matrix of a multi-Gabor dictionary consists of Gram matrices of individual dictionaries $\mathbf{D}_w$ and cross-Gram matrices [32] between the dictionaries. Denoting a cross-Gram matrix as $\mathbf{G}_{w,v} = \mathbf{D}_w^* \mathbf{D}_v$ the overall Gram matrix is a block matrix with the following structure

$$\begin{bmatrix} \mathbf{G}_{1,1} & \mathbf{G}_{1,2} & \dots & \mathbf{G}_{1,W} \\ \mathbf{G}_{2,1} & \mathbf{G}_{2,2} & \dots & \mathbf{G}_{2,W} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{G}_{W,1} & \mathbf{G}_{W,2} & \dots & \mathbf{G}_{W,W} \end{bmatrix}. \tag{5}$$

A cross-Gram matrix $\mathbf{G}_{w,v}$ shares the same twisted convolution structure with the regular Gram matrix with kernel $\mathbf{h}_{w,v} = \mathbf{D}_w^* \mathbf{g}_v$ only if the time-frequency shifts are equal i.e. $a_w = a_v$ and $M_w = M_v$. In the case the parameters differ, the direct twisted convolution structure is lost. The structure can be recovered on a finer "common" time-frequency grid given by the time step $\gcd(a_w, a_v)$ and the number of frequency bins $\mathrm{lcm}(M_w, M_v)$. The most efficient case is achieved when a_w and a_v are divisible by $a_{\text{min}} = \min\{a_w, a_v\}$ and M_w and M_v both divide $M_{\text{max}} = \max\{M_w, M_v\}$ resulting to a common grid given by a_{min} and M_{max}. In the residual update step of the inner products of the residual with the w-th dictionary, the modulated kernel is subsampled by ratios a_w/a_{min} and M_{max}/M_w in horizontal and vertical directions respectively. To illustrate, consider a multi-Gabor dictionary consisting of two Gabor dictionaries $\mathbf{D}_1 = \mathbf{D}_{(\mathbf{g}_1, a_1, M_1)}$ and $\mathbf{D}_2 = \mathbf{D}_{(\mathbf{g}_2, a_2, M_2)}$ with $a_1 = 4a_{\text{min}}$, $M_1 = 8a_{\text{min}}$ and $a_2 = a_{\text{min}}$, $M_2 = 2a_{\text{min}}$. Both cross-kernels $\mathbf{h}_{1,2}$ and $\mathbf{h}_{2,1}$ are computed with $a_{\text{min}} = a_2$ and $M_{\text{max}} = M_1$. The example in Fig. 2 depicts an update of inner products of the residual with both dictionaries on the common grid. The steps of the coefficient residual update in the coefficient domain are summarized in Alg. 2.

3.2. Keeping Track Of The Maximum

Performing the full search for the maximum inner product in each selection step is highly inefficient. The authors of MPTK [27] proposed to store positions of maxima for each window time position and organize them in a partial hierarchical tournament-style tree. Such tree contains at each level maxima from pairs from one level below. Since the residual update affects only a limited number of neighboring time positions, a bottom-up tree update becomes more efficient than a full search. In the present method, we additionally exploit localization in frequency, which allows us to keep track of maxima for individual window time positions (across frequency bins) in a similar manner. See Fig. 3 for a depiction of the tournament tree structure and for an example of a worst-case bottom-up update of a 3–level tree across time-frames.

4. COMPARISON WITH MPTK

In order to showcase the efficiency of the proposed algorithm and its implementation, in this section, we present a comparison with MPTK (version 0.7.0), which is considered to be the fastest implementation available. To our knowledge there is no previous implementation of coefficient-domain MP that is able to decompose signals of the size considered here. We measure the duration of the matching pursuit decomposition only. From MPTK, we used the modified `mpd` utility tool. The testing signal was the first channel from the file no. 39 from the SQAM database [33], which is a 137 seconds long piano recording sampled at 44.1 kHz totaling $6 \cdot 10^6$ samples. Both implementations link the same FFTW library [34] version 3.3 and were compiled using the GCC (g++) compiler (version 7.2.0) with the `-Ofast` optimization flag enabled. The creation of the FFTW plans and the computation of the kernels was

Input: $c_{\max}, w_{\max}, m_{\max}, n_{\max}, \langle \mathbf{d}_{\max}, \overline{\mathbf{d}}_{\max} \rangle, a_1, \ldots, a_W, M_1, \ldots, M_W, \mathbf{h}_{w,v}\ (w, v = 1, \ldots, W)$
Output: Inner products to be updated $\mathbf{c}_1^{\mathbf{r}_k}, \ldots, \mathbf{c}_W^{\mathbf{r}_k}$
for $w = 1, \ldots, W$ **do**
  $\mathbf{c}_w^{\mathbf{r}_k} \leftarrow$`singleDictionaryResidualUpdate(`$m_{\max}$, w, $\mathbf{c}_w^{\mathbf{r}_k}$`)`
  if $|\langle \mathbf{d}_{\max}, \overline{\mathbf{d}}_{\max} \rangle| > 0$ **then**
    $m_{\text{conj}} = M_{w_{\max}} - m_{\max}$
    $\mathbf{c}_w^{\mathbf{r}_k} \leftarrow$`singleDictionaryResidualUpdate(`$m_{\text{conj}}$, w, $\mathbf{c}_w^{\mathbf{r}_k}$`)`
  end
end

Function $\mathbf{c}_w \leftarrow$`singleDictionaryResidualUpdate(`$m_{\max}$, w, $\mathbf{c}_w$`)`**:**
  $a_{\text{rat}} \leftarrow a_w / a_{w_{\max}}$, $M_{\text{rat}} \leftarrow M_{w_{\max}} / M_w$
  $a_{\text{step}} = a_{\text{rat}}$ or 1 if $a_{\text{rat}} < 1$, $M_{\text{step}} = M_{\text{rat}}$ or 1 if $M_{\text{rat}} < 1$
  `/* Determine index sets */`
  Define $\mathcal{M}$, horizontal index set with stride M_{step} in cross kernel $\mathbf{h}_{w_{\max},w}$ taking into the account the misalignment of the grids.
  Define $\mathcal{N}$, vertical index set in a similar way using stride a_{step}.
  Define $\mathcal{I}$, residual coefficient vector index set covered by the kernel.
  `/* Update the residual (as in (4)) using truncated, subsampled and modulated cross-kernel: */`
  $\mathbf{c}_w(\mathcal{I}) = \mathbf{c}_w(\mathcal{I}) - c_{\max} \mathbf{h}^{(m_{\max})}_{w_{\max},w}(\mathcal{M}, \mathcal{N})$
return

Algorithm 2: Approximate coefficient-domain MP residual update

excluded from the measurements. The specification of the PC the timing was performed on was Intel® Core™ i5-4570 3.20 GHz, 16 GB RAM running Ubuntu 16.04. The timing was done on an idle machine using the high-precision timer from the C++11 standard library `chrono`. The data type was double precision floating point. In the decomposition we performed $1.8 \cdot 10^5$ iterations each. By fixing the number of iterations instead of a desired approximation estimate, we ensure that execution time is roughly independent of the considered signal. Table 1 shows a comparison of execution times, in seconds, for a single Gabor dictionary with the Blackman window, numbers of bins $M_w = 512, \ldots, 8192$ (additionally also 16384 for the proposed implementation) and various hop sizes a (and therefore redundancies). The length of the window was always equal to M_w, as required by MPTK. Additionally, a comparison of execution times using two multi-dictionaries is shown, each of which consists of five Gabor dictionaries with $M_w = 512, \ldots, 8192$ (at redundancies 4 and 8 per dictionary).

In the tested setting, the proposed implementation clearly outperforms MPTK in terms of computational speed. The memory requirements are however notably higher since the residual is stored in the coefficient domain and, additionally, the pre-computed kernels and the modulation factors must be stored as well.

Bins (M)	512	1024	2048	4096	8192	16384
	$a = M/4$					
MPTK	3.96	8.40	17.0	36.4	75.7	–
Proposed	0.92	1.00	1.02	1.03	1.03	1.08
	$a = M/8$					
MPTK	6.73	15.1	30.2	61.3	147	–
Proposed	1.70	1.90	1.95	2.10	2.20	2.21
	$a = M/16$					
MPTK	13.2	28.3	56.8	119	274	–
Proposed	3.20	3.50	4.00	4.50	4.60	5.08
	$a = M/32$					
MPTK	23.2	52.6	110	233	530	–
Proposed	6.35	7.40	7.90	8.20	9.60	10.7

Multi-Gabor	$a = M/4$	$a = M/8$
MPTK	142	285
Proposed	10.6	20.9

Table 1: Execution time in seconds for MPTK and the proposed method on Gabor and Multi-Gabor dictionaries (180k selection steps). The Multi-Gabor dictionaries are a concatenation of the dictionaries with $M = 512, \ldots, 8192$ at redundancies $a = M/4$ and $a = M/8$, respectively.

4.1. Convergence in practice

Truncating the Gramian kernel introduces a small, but accumulating error in the computation of the coefficient-domain residual. Hence, we compare the residual norm achieved after k selection steps by MPTK and the proposed method for various values of k and the truncation threshold ϵ. The experiment was performed using a concatenation of 3 Gabor dictionaries with Blackman window, $M_w = 512, 1024, 2048$ and $a_w = M_w/4$, as used in Section 4 and considering the audio test signal used previously. For other audio signals, we observed similar behavior. Generally, our implementation of the proposed method is set to terminate when $\sum_{l=1}^{k} \boldsymbol{\varrho}_{l-1}(p_l)^2 > \|\mathbf{x}\|_2^2$, i.e. when the energy of the approximation coefficients exceeds the signal norm. This condition serves as a cheap indicator that further selection steps are expected to harm the approximation quality. We also state the final approximation error and number k of selection steps performed before termination. In all experiments, approximation quality of the proposed method follows MPTK closely until shortly before termination.

5. CONCLUSION

We have presented an accelerated MP algorithm alongside a reference implementation suitable for multi-Gabor dictionaries. Due to the structure of the Gram matrix of the multi-Gabor dictionary, the

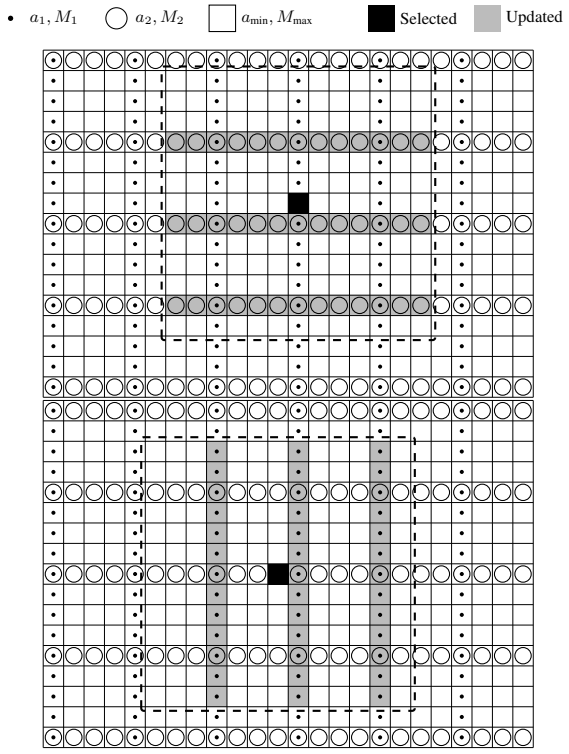

Figure 2: Illustration of the residual update between dictionaries using cross kernels. The left figure shows the case when a coefficient from the dictionary 1 was selected in the selection step of MP and inner products with dictionary 2 are being updated and vice versa right. The dashed line square depicts the area covered by a cross-kernel with respect to the common grid $a_{\min}, M_{\max}$.

coefficient domain residual update step becomes very fast while the memory requirements for storing the inner products between the atoms remain constant with increasing signal length. Moreover, the time and frequency locality of the residual update in turn allows faster search for the maximum in the next iteration. Benchmarks show that, depending on the dictionary, our implementation is 3.5–70 times faster than the standard MPTK implementation while achieving similar approximation quality, unless a simple termination condition is reached. In the single dictionary case, the most notable feature is that the execution time is virtually independent of the number of bins M when the redundancy M/a is fixed. Moreover, as it turned out, MPTK could not handle dictionaries with the number of bins higher than 8192. The tested code does not use explicit optimization techniques like exploiting the SIMD operations or parallelization of the code, leaving the possibility for future improvements. Finally, the presented algorithm does not, as is, achieve arbitrarily small energy of the (true) residual, but rather an approximation quality depending on the chosen threshold ϵ. However, preliminary experiments suggest that a simple reset procedure is sufficient to restore convergence, i.e., $\|r_k\|_2^2 \to 0$. Such resets will be considered in the upcoming, extended manuscript [28].

6. REFERENCES

[1] G. Davis, S. Mallat, and M. Avellaneda, "Adaptive greedy approximations," *Constructive Approximation*, vol. 13, no. 1, pp. 57–98, 1997.

[2] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Tran. Signal Processing*, vol. 41, no. 12, pp. 3397–3415, Dec 1993.

[3] R. A. DeVore and V. N. Temlyakov, "Some remarks on greedy algorithms," *Advances in Computational Mathematics*, vol. 5, no. 1, pp. 173–187, Dec 1996.

[4] R. Gribonval, R. M. Figueras i Ventura, and P. Vandergheynst, "A simple test to check the optimality of a sparse signal approximation," *Signal Processing*, vol. 86, no. 3, pp. 496 – 510, 2006.

[5] R. Gribonval and P. Vandergheynst, "On the exponential convergence of matching pursuits in quasi-incoherent dictionaries," *IEEE Tran. Information Theory*, vol. 52, no. 1, pp. 255–261, Jan 2006.

[6] G. Rath and C. Guillemot, "A complementary matching pursuit algorithm for sparse approximation," in *Proc. 16th European Signal Processing Conference (EUSIPCO)*, Aug 2008, pp. 1–5.

[7] B. L. Sturm and M. G. Christensen, "Cyclic matching pursuits with multiscale time-frequency dictionaries," in *Conf. Record of the 44th Asilomar Conference on Signals, Systems and Computers*, Nov 2010, pp. 581–585.

[8] T. Blumensath and M. E. Davies, "Gradient pursuit for non-linear sparse signal modelling," in *Proc. European Signal Processing Conference (EUSIPCO)*. IEEE, 2008, pp. 1–5.

[9] L. Rebollo-Neira, M. Rozložník, and P. Sasmal, "Analysis of a low memory implementation of the orthogonal matching pursuit greedy strategy," *CoRR*, vol. abs/1609.00053v2, 2017.

[10] M. Yaghoobi and M. E. Davies, "Fast and scalable: A survey on sparse approximation methods," Tech. Rep., The University of Edinburgh, 2009.

[11] I. Rish and G. Grabarnik, *Sparse modeling: Theory, algorithms, and applications*, CRC Press, 2015.

[12] B. L. Sturm, *Sparse approximation and atomic decomposition: Considering atom Interactions in evaluating and building signal representations*, Ph.D. thesis, University of California, 2009.

[13] M. D. Plumbley, T. Blumensath, L. Daudet, R. Gribonval, and M. E. Davies, "Sparse representations in audio and music: From coding to source separation," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 995–1005, June 2010.

[14] D. Zantalis, *Guided matching pursuit and its application to sound source separation*, Ph.D. thesis, University of York, 2016.

[15] R. Gribonval, P. Depalle, X. Rodet, E. Bacry, and S. Mallat, "Sound signals decomposition using a high resolution matching pursuit," in *Proc. Int. Computer Music Conf. (ICMC'96)*, Aug. 1996, pp. 293–296.

[16] R. Gribonval, "Fast matching pursuit with a multiscale dictionary of Gaussian chirps," *IEEE Tran. Signal Processing*, vol. 49, no. 5, pp. 994–1001, May 2001.

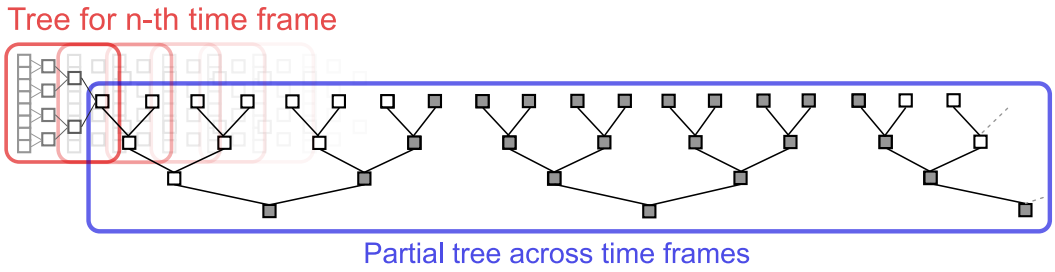

Figure 3: Schematic of the tournament tree structure for keeping track of maxima. A red border indicates individual tournament trees across frequency bins for each time frame. The blue border indicates the partial tournament tree across time frames. An example of the worst-case 10 element bottom-up update of the tree across frames is shown in gray.

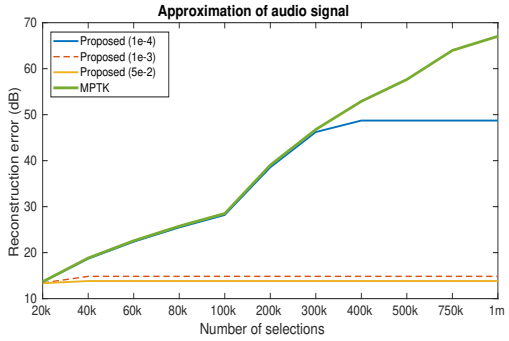

Figure 4: Approximation errors achieved by MPTK and the proposed method. The proposed method was evaluated for various threshold values ϵ. The proposed method terminated after 368235 ($\epsilon = 10^{-4}$), 35115 ($\epsilon = 10^{-3}$) and 29138 ($\epsilon = 5 \cdot 10^{-3}$) steps, achieving an approximation quality of 48.72, 14.84 and 13.83 dB, respectively.

[17] B. L. Sturm and J. D. Gibson, "Matching pursuit decompositions of non-noisy speech signals using several dictionaries," in *Proc. IEEE Int. Conf. on Acoustics Speech and Signal Processing*, May 2006, vol. 3, pp. III–III.

[18] E. Ravelli, G. Richard, and L. Daudet, "Union of MDCT bases for audio coding," *IEEE Tran. Audio, Speech, and Language Processing*, vol. 16, no. 8, pp. 1361–1372, Nov 2008.

[19] G. Chardon, T. Necciari, and P. Balazs, "Perceptual matching pursuit with Gabor dictionaries and time-frequency masking," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 3102–3106.

[20] O. Derrien, "Time-scaling of audio signals with muti-scale Gabor analysis," in *Proc. Int. Conf. Digital Audio Effects (DAFx–07)*, Bordeaux, France, Sept. 2007.

[21] B. L. Sturm, L. Daudet, and C. Roads, "Pitch-shifting audio signals using sparse atomic approximations," in *Proc. 1st ACM Workshop on Audio and Music Computing Multimedia*, New York, NY, USA, 2006, AMCMM '06, pp. 45–52, ACM.

[22] R. Gribonval, "Sparse decomposition of stereo signals with matching pursuit and application to blind separation of more than two sources from a stereo mixture," in *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, May 2002, vol. 3, pp. III–3057–III–3060.

[23] G. Bhattacharya and P. Depalle, "Sparse denoising of audio by greedy time-frequency shrinkage," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 2898–2902.

[24] P. Leveau and L. Daudet, "Multi-resolution partial tracking with modified matching pursuit," in *Proc. 14th European Signal Processing Conference*, Sept 2006, pp. 1–4.

[25] P. L. Søndergaard, B. Torrésani, and P. Balazs, "The linear time frequency analysis toolbox," *International Journal of Wavelets, Multiresolution Analysis and Information Processing*, vol. 10, no. 4, pp. 1–27, 2012.

[26] Z. Průša, P. L. Søndergaard, N. Holighaus, Ch. Wiesmeyr, and P. Balazs, "The large time-frequency analysis toolbox 2.0," in *Sound, Music, and Motion*, LNCS, pp. 419–442. Springer International Publishing, 2014.

[27] S. Krstulović and R. Gribonval, "MPTK: Matching pursuit made tractable," in *Proc. Int. Conf. on Acoustics Speech and Signal Processing ICASSP 2006*, May 2006, vol. 3, pp. III–496–III–499.

[28] Z. Průša, Nicki Holighaus, and Peter Balazs, "Fast Matching Pursuit with Multi-Gabor Dictionaries," *submitted to ACM Transactions on Mathematical Software*, 2020.

[29] Stphane Mallat, *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*, Academic Press, 3rd edition, 2008.

[30] K. Gröchenig, *Foundations of time-frequency analysis*, Applied and numerical harmonic analysis. Birkhäuser, Boston, Basel, Berlin, 2001.

[31] J. Le Roux, H. Kameoka, N. Ono, and S. Sagayama, "Fast signal reconstruction from magnitude STFT spectrogram based on spectrogram consistency," in *Proc. 13th Int. Conf. on Digital Audio Effects (DAFx-10)*, Sept. 2010, pp. 397–403.

[32] P. Balazs, "Frames and finite dimensionality: Frame transformation, classification and algorithms," *Applied Mathematical Sciences*, vol. 2, no. 41–44, pp. 2131–2144, 2008.

[33] The European Broadcasting Union, "Tech 3253: Sound quality assessment material recordings for subjective tests," Tech. Rep., Geneva, Sept. 2008.

[34] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proc. of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005, Special issue on "Program Generation, Optimization, and Platform Adaptation".