Tree-Based Pursuit: Algorithm and Properties

Philippe Jost, Pierre Vandergheynst, *Member, IEEE*, and Pascal Frossard, *Senior Member, IEEE*

***Abstract*—This paper proposes a tree-based pursuit algorithm that efficiently trades off complexity and approximation performance for overcomplete signal expansions. Finding the sparsest representation of a signal using a redundant dictionary is, in general, an NP-hard problem. Even suboptimal algorithms such as Matching Pursuit remain highly complex. We propose a structuring strategy that can be applied to any redundant set of functions, and which basically groups similar atoms together. A measure of similarity based on coherence allows for representing a highly redundant subdictionary of atoms by a unique element, called molecule. When the clustering is applied recursively on atoms and then on molecules, it naturally leads to the creation of a tree structure. We then present a new pursuit algorithm that uses the structure created by clustering as a decision tree. This tree-based algorithm offers important complexity reduction with respect to Matching Pursuit, as it prunes important parts of the dictionary when traversing the tree. Recent results on incoherent dictionaries are extended to molecules, while the true highly redundant nature of the dictionary stays *hidden* by the tree structure. We then derive recovery conditions on the structured dictionary, under which tree-based pursuit is guaranteed to converge. Experimental results finally show that the gain in complexity offered by tree-based pursuit does in general not have a high penalty on the approximation performance. They show that the dimensionality of the problem is reduced thanks to the tree construction, without significant loss of information.**

***Index Terms*—Clustering, greedy algorithms, matching pursuit, redundant expansions, sparse approximation, tree search.**

I. Introduction

BUILDING good sparse approximations of functions is one of the major themes in approximation theory. When applied to signals, images, or any kind of multidimensional data, it allows to deal with basic building blocks that essentially synthesize the information at hand. It has been known since the early successes of wavelet analysis that sparse expansions very often result in efficient algorithms for characterizing signals, or even for analysis and compression. An interesting and increasingly popular way of achieving sparsity is to turn to very redundant systems. It often allows for short-length representation of signals, since the probability of finding a sparse approximation generally increases with the redundancy of the dictionary. In most cases, sparsity is measured by the ℓ_0 norm of the vector of coefficients.

Manuscript received May 12, 2005; revised January 31, 2006. The associate editor coordinating the review of the manuscript and approving it for publication was Prof. Ioan Tabus. This work was supported in part by the Swiss National Science Foundation through NCCR IM2 and under Grant PP002-68737.

The authors are with Ecole Polytechnique Fédérale de Lausanne (EPFL), Signal Processing Institute, CH-1015 Lausanne, Switzerland (e-mail: philippe.jost@epfl.ch; pierre.vandergheynst@epfl.ch; pascal.frossard@epfl.ch).

Digital Object Identifier 10.1109/TSP.2006.882080

Finding the best linear expansion using a redundant dictionary of functions is a daunting task and even a NP-hard problem [1] in the general case. Despite the difficulty to find the *best*, sparsest solution, it is possible to find sufficiently *good* representations that are nearly optimal. Suboptimal heuristics have been developed that recover the best approximations of a function in a redundant dictionary. Among the most popular algorithms that find good suboptimal solutions to the sparsest signal representation problem, we can cite Matching Pursuit [2] and Basis Pursuit [3]: both reach a solution close to optimum by relaxing some constraints of the original optimization problem. Even if specific optimizations are possible for particular classes of dictionaries, the complexity of these algorithms however remains very high in general.

Several methods have recently been proposed in order to decrease the computational complexity to find sparse signal expansions. They generally propose modifications of either the search algorithm itself, or the dictionary. Starting from existing algorithms, it is indeed possible to introduce small changes to obtain efficient search algorithms. A two-stage design is proposed in [4]–[6], where the original dictionary functions are approximated by linear combinations of very simple, elementary vectors. The search is then performed in the space of elementary vectors, hence a great reduction in computational complexity.

Approximation of functions of the dictionary or special constructions can also lead to efficient search algorithms, without an important penalty on the approximation performance [5]. Multiscale [7] or subband dictionaries [8] can be used to decrease the search complexity, where the linearity of the inner product can even be further exploited to speed up the computation, at the price of higher memory requirements. Similarly, [9] proposed to use a dictionary that is based on damped sinusoids, which can be efficiently derived using simple recursive filter banks. Since the size of the dictionary has obviously an important impact on the search complexity, several studies have also been proposed to prune the dictionary to its most meaningful elements, by vector quantization for example [10], [11]. In general, these methods however only apply to specific dictionaries.

One of the aims of this paper is to study the reduction of the computational complexity of the search for the sparsest signal expansion, for any, highly redundant, dictionary. It naturally leads to the notion of data structuring, which becomes critical when the amount of data gets very large. Dictionary functions with similar properties can be clustered together, in order to facilitate the search for the sparsest representation. Clustering is a widely used technique when the amount of data is huge and hides the underlying structures, see [12] for a survey. Clustering algorithms depend on a measure to quantify the similarity between two objects. Proper data arrangement then allows for the development of tree data structures, which can be efficiently used for search when a huge amount of data is present [13]. Tree search has been proposed in [14] in order to improve the performance of Matching Pursuit expansion. We however propose to

study tree-based pursuit from a complexity reduction perspective, as an interesting tradeoff between efficient implementation and sufficiently sparse signal approximation.

The paper is organized as follows. Section II proposes an overview of linear expansions using redundant dictionaries of functions. Section III presents a structuring method that allows to represent a subset of highly correlated atoms by a single element, called molecule. Hierarchical clustering then allows for building trees, where each node corresponds to a molecule that encompasses the characteristics of all its relative children. A tree construction method is then proposed that respects the necessary conditions for nodes at each level to be sufficiently incoherent. A tree-based pursuit algorithm is described in Section IV, which exploits the tree structure to reduce the computational complexity of the pursuit. Performance and characteristics of the algorithm are analyzed in Section V. A bound is derived, which ensures that molecules cover the same span as the initial dictionary. A minimal condition ensuring that the algorithm chooses only *good* molecules under the root node is also presented. Finally, Section VI illustrates the performance of Tree-Based Pursuit in terms of approximation and complexity, compared to Matching Pursuit.

II. Sparse Approximation Using Redundant Dictionaries

A. Sparse Approximations

For the last few years, there has been a tremendous activity in the field of sparse approximations. This is partly motivated by the potential of the related techniques for typical tasks in signal processing such as analysis, dimensionality reduction, denoising, or compression. This section provides an overview of the main recent results on sparse approximations, and practical algorithms like Matching Pursuit.

Given a d-dimensional signal f in a real vector space, the central problem faced in this paper is the following: compute a good approximation $\tilde{f}_N$ as a linear superposition of N basic elements, which are often called atoms, picked up in a huge collection of signals $\mathcal{D}$, usually referred to as a dictionary. The dictionary is said to be redundant when its cardinality $|\mathcal{D}| \gg d$. The approximant $\tilde{f}_N$ is sparse when $N \ll d$, where

$$\tilde{f}_N = \sum_{k=0}^{N-1} c_k g_k, \qquad g_k \in \mathcal{D}. \tag{1}$$

There is no particular requirement concerning the dictionary, except that it should span the signal space $\mathcal{H}$, and there is no prescription on how to compute the coefficients c_k in (1). The main advantage of this class of techniques is the complete freedom in designing the dictionary, which can then be efficiently tailored to closely match signal structures.

This problem is better studied under the form of the following constrained optimization:

$$P_0 : \text{minimize } \|c\|_0 \text{ subject to } \left\| f - \sum_{k=0}^{N-1} c_k g_{\gamma_k} \right\|_2 < \epsilon$$

where $\|c\|_0$ counts the number of nonzero entries in the sequence $\{c_k\}$ and ϵ is the maximal acceptable error. Usually, finding the solution of P_0 would be a hopeless combinatorial problem.

Simple greedy strategies such as Matching Pursuit and Orthogonal Matching Pursuit are able to recover very good approximants [15]. On the downside, these results only hold for a limited class of dictionaries: $\mathcal{D}$ has to be sufficiently *incoherent*. The coherence of a dictionary $\mathcal{D}$ is defined as

$$\mu = \sup_{\substack{i,j \in \mathcal{D} \\ i \neq j}} |\langle g_i, g_j \rangle|. \tag{2}$$

Coherence is a measure of the redundancy of the dictionary and small coherence means that $\mathcal{D}$ is not too far from an orthogonal basis (although it may be highly overcomplete). More properties of such dictionaries can be found in [16]–[18]. We will also come back to incoherent dictionaries in the course of this paper.

These results tell us that, if a sufficiently sparse solution exists in a sufficiently incoherent dictionary, it can be found by solving a problem closely connected to P_0. However, in practice, redundant dictionaries, most of the time, do not satisfy the incoherence condition while still leading to good results.

B. Greedy Algorithms: Matching Pursuit

Greedy algorithms iteratively construct an approximant by selecting the element of the dictionary that best matches the signal at each iteration. The pure greedy algorithm is known as *Matching Pursuit* [12]. Assuming that all atoms in $\mathcal{D}$ have norm one, we initialize the algorithm by setting $R^0 f = f$ and we first decompose the signal as

$$R^0 f = \langle g_{\gamma_0}, R^0 f \rangle g_{\gamma_0} + R^1 f$$

where g_{γ_0} is chosen so as to maximize the correlation with $R^0 f$

$$g_{\gamma_0} = \arg\max_{\mathcal{D}} \left| \langle g_{\gamma_0}, R^0 f \rangle \right|.$$

We then iterate the procedure on the residual $R^1 f$ and, after N steps, build the following approximation:

$$f = \sum_{n=0}^{N-1} \langle g_{\gamma_n}, R^n f \rangle g_{\gamma_n} + R^N f$$

where the norm of the residual (approximation error) satisfies

$$\|R^N f\|^2 = \|f\|^2 - \sum_{n=0}^{N-1} |\langle g_{\gamma_n}, R^n f \rangle|^2.$$

The performance of greedy algorithms such as Matching Pursuit are tightly linked to the structure of the dictionary. The coherence μ described above is often not sufficient to represent the properties of a dictionary, since it represents a worst case bound, and does not take into account the local structures of the dictionary. The structural redundancy [19] of a dictionary provides important information about the structure of a redundant dictionary. Matching Pursuit converges exponentially fast in finite dimension [1], [2]. There exist two constants $\alpha > 0$, reflecting the optimality of the pursuit algorithm, and $\beta > 0$, characterizing the redundancy of the dictionary, such that

$$\|R^{n+1} f\| \leq (1 - \alpha^2 \beta^2)^{1/2} \|R^n f\| \tag{3}$$

where β can be expressed as

$$\beta = \inf_{a,\|a\|=1} \sup_{g_i \in \mathcal{D}} |\langle a, g_i \rangle| . \tag{4}$$

This equation confirms that the algorithm will behave well, provided there is always an atom closely aligned with the residual. The properties of the signal, dictionary and algorithm, are tightly linked.

As already mentioned, solving the sparse approximation problem of (1) using a redundant dictionary is of combinatorial complexity. The greedy heuristic finds in general a good approximant to the problem in polynomial time. There is however no guarantee on the optimality of the solution, except in the case where sufficient conditions are set on the dictionary [18]. However, polynomial time still does not mean fast! Typical implementations of Matching Pursuit suffer from high computational complexity when compared to most orthogonal transforms. In the remainder of this paper, we propose to reduce the complexity by an efficient organization of the dictionary. We group similar atoms together, and represent them by a unique element called *molecule*. Applying clustering recursively on atoms and molecules yields a hierarchical tree structure, which can be exploited to design a search algorithm with greatly reduced complexity.

III. Structuring Redundant Dictionaries

A. From Atoms to Molecules

This section discusses clustering of a generic, redundant dictionary, which eventually leads to the creation of a tree structure. First, it describes the problem of representing a group of highly correlated dictionary atoms by a unique element. We then discuss the characteristics which are necessary for a dictionary to be efficiently clustered and organized in a tree structure.

Let the elements of the dictionary $\mathcal{D} = \{g_i\}_{i \in \Gamma}$ be labeled by the index set Γ. A subdictionary $\mathcal{D}_\Lambda$ is such that $\mathcal{D}_\Lambda = \{g_i\}_{i \in \Lambda}$, where $\Lambda \subset \Gamma$ and $\Lambda \neq \emptyset$. A collection of subdictionaries $\{\mathcal{D}_{\Lambda_i}\}$ forms a partition of the dictionary $\mathcal{D}$ if $\bigcup_i \Lambda_i = \Gamma$ and $\Lambda_i \bigcap \Lambda_j = \emptyset, \forall i \neq j$. If the atoms in $\mathcal{D}$ are sufficiently uncorrelated, a simple greedy algorithm is able to recover a sparse approximation of the signal (see, for example, [18]). This is not the case for highly correlated redundant dictionaries. It can be explained intuitively by the fact that high correlation in the dictionary can fool the pursuit and result in wrong choices. We are thus going to try to represent a highly correlated subdictionary $\mathcal{D}_{\Lambda_i}$ by a single molecule, while at the same time minimizing the correlation among molecules. This procedure should result in a set of molecules that behaves like a (quasi-) incoherent dictionary.

Let us first define the minimal coherence λ_Λ of a subdictionary by:

$$\lambda_\Lambda = \min_{i,j \in \Lambda} |\langle g_i, g_j \rangle| . \tag{5}$$

A subdictionary will be referred to as *reducible* when λ_Λ is positive and sufficiently big. In order to quantify the adequation of the molecule in representing the atoms in the subdictionary $\mathcal{D}_{\Lambda_i}$, a distance measure has to be defined. Let $d(g_i, g_j)$ be a measure of the distance between two unit energy atoms g_i and g_j. In this paper, we chose to use the following distance measure, derived from the simple cosine function:

$$d(g_i, g_j) = 1 - |\langle g_i, g_j \rangle|^2 . \tag{6}$$

Note that an atom g_i can be considered as equivalent to $-g_i$, from an approximation point of view, the sign of the weights a_i in (1) could be reversed. The distance measure given in (6) is independent of the direction of g_i as $d(g_i, -g_i) = 0$.

Most clustering algorithms represent a cluster by a centroid whose mean distance to all elements it represents is minimized. Let us define the optimal centroid or unit norm molecule m_Λ^{opt}, for a subdictionary $\mathcal{D}_\Lambda$, by

$$m_\Lambda^{\text{opt}} = \arg \min_{\substack{m \\ \|m\|=1}} \sum_{i \in \Lambda} d(m, g_i). \tag{7}$$

Using the distance measure defined in (6), the optimal centroid becomes

$$\begin{aligned} m_\Lambda^{\text{opt}} &= \arg \min_{\substack{m \\ \|m\|=1}} \sum_{i \in \Lambda} 1 - |\langle m, g_i \rangle|^2 && (8) \\ &= \arg \max_{\substack{m \\ \|m\|=1}} \sum_{i \in \Lambda} |\langle m, g_i \rangle|^2 && (9) \\ &= \arg \max_{\substack{m \\ \|m\|=1}} m^* A_\Lambda A_\Lambda^* m && (10) \end{aligned}$$

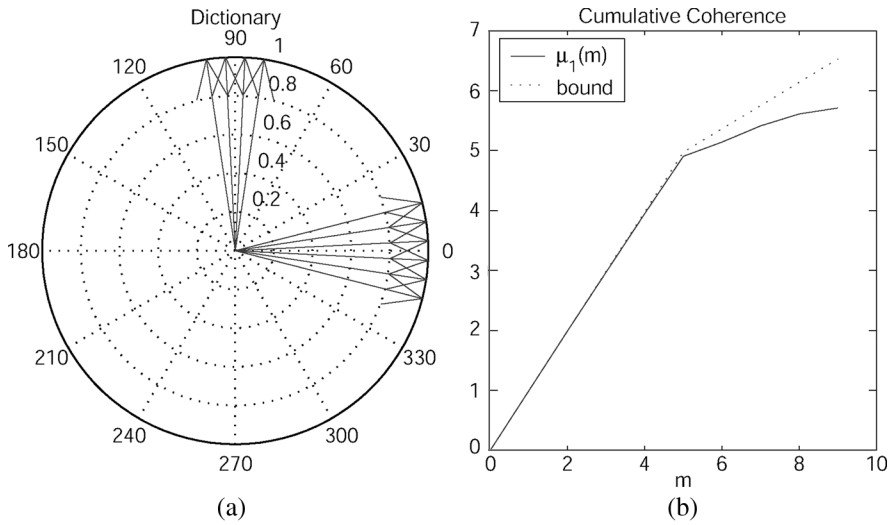
where the columns of the matrix A_Λ are the atoms of the subdictionary $\mathcal{D}_\Lambda$. The molecule m_Λ^{opt} is the eigenvector associated to the biggest eigenvalue of the matrix $A_\Lambda A_\Lambda^*$. The eigenvalues of $A_\Lambda A_\Lambda^*$ are equal to the eigenvalues of $A_\Lambda^* A_\Lambda$ (see [20, Theorem 1.3.20]). This last matrix is the Grammian of A_Λ. Later, Fig. 4 illustrates the reduction capabilities of a molecule regarding a group of similar atoms. As the matrix $A_\Lambda A_\Lambda^*$ is symmetric, the associated eigenvalues are real and the associated eigenvectors are orthogonal. The molecule m_Λ^{opt} is also equivalent to the dominant left singular vector of the matrix A_Λ [20].

B. Dictionary Characterization

In the previous subsection, we introduced the definition of molecule in order to structure the *information* at hand in a highly redundant subdictionary. We will now see how a dictionary can be partitioned into disjoint subdictionaries represented by molecules through a simple clustering procedure. Further recursive application of clustering on the set of molecules results in a hierarchical tree structure that will be used by the search algorithm.

We previously stated that representing a subdictionary by a molecule makes sense only for *reducible* subdictionaries. By extension, a dictionary $\mathcal{D}$ is said to be *reducible* if it contains a partition $\{\mathcal{D}_{\Lambda_i}\}$, such that all its subdictionaries are reducible and $|\{\mathcal{D}_{\Lambda_i}\}| \ll |\mathcal{D}|$, i.e., the number of subdictionaries is much smaller than the number of atoms in the dictionary. A special case of *reducible* dictionaries is represented by the *block incoherent* dictionaries [21]. These dictionaries are such that it is possible to find a partition having a small block coherence μ_B defined by

$$\mu_B = \max_{i \neq j} \max_{\substack{k \in \Lambda_i \\ l \in \Lambda_j}} |\langle g_k, g_l \rangle| . \tag{11}$$



(a) (b)

Fig. 1. (a) Simple *block-incoherent* dictionary made of two highly redundant parts. (b) Evolution of its cumulative coherence and the upper bound provided by (13).

If $\mathcal{D}$ is *reducible*, then the coherence μ of $\mathcal{D}$ is large; the reverse is however not necessarily true. A dictionary $\mathcal{D}$ can have a large coherence μ without being *reducible*, due to the fact that the coherence given in (2) only reflects an extreme property of the dictionary. Similarly, the quantity β defined in (4), or the structural redundancy [19], also reports an extreme property of the dictionary. For *block-incoherent* dictionaries, the structural redundancy is low and provides some *inter* subdictionaries redundancy measure. It is however closely related to the block-coherence μ_B given in (11).

The cumulative coherence is a refinement of the simple coherence measure and therefore provides much more information about the dictionary. It is defined as follows:

$$\mu_1(m) = \max_{|\Lambda|=m} \max_{i \notin \Lambda} \sum_{j \in \Lambda} |\langle g_i, g_j \rangle| . \tag{12}$$

A dictionary whose cumulative coherence grows slowly (i.e. $\mu_1(m) \ll \mu m$) is said to be *quasi-incoherent* [18]. If it grows fast, it is at least possible to have one highly correlated subdictionary. The cumulative coherence can be bounded using the coherence, $\mu_1(m) \leq m\mu$. In the special case of *block-incoherent* dictionaries, a better bound on the cumulative coherence $\mu_1(m)$ can even be proposed. Let k be the cardinality of the most populated highly correlated subdictionary, we then have

$$\mu_1(m) \leq \begin{cases} m\mu, & \text{if } m < k \\ (k-1)\mu + (m-k+1)\mu_B, & \text{if } m \geq k. \end{cases} \tag{13}$$

The cumulative coherence provides more accurate *local* information than the coherence, but is more complex to compute. Moreover, a fast growing cumulative coherence is not a sufficient condition for a dictionary to be *reducible*: it reflects the behavior of the dictionary in the region of the space of signals that is best *covered* by the dictionary [15]. For example, in the case of *block-incoherent* dictionaries, the cumulative coherence grows rapidly from $\mu_1(0)$ up to $\mu_1(k-1)$ and then grows slowly, with k being the cardinality of the most populated subdictionary. Fig. 1 presents the evolution of the cumulative coherence for a dictionary having two highly redundant parts. For $m = 5$, there is a sharp inflection of the curve as the cardinality of the most populated group of atoms is $k = 6$. To summarize, a quasi-incoherent dictionary has both small coherence, and small structural redundancy, and its cumulative coherence grows slowly. Block-incoherent dictionaries rather have a large coherence and a cumulative coherence that grows fast up to an inflexion point at $m = k - 1$, and then grows slowly. Block-incoherent dictionaries are good candidates for one-step clustering of atoms into molecules.

C. Tree-Structured Dictionaries

The hypothesis that the dictionary is *reducible* ensures that it is possible to partition it into *reducible* subdictionaries, and to recursively find molecules. However, we have not yet provided a way to compute the partition of $\mathcal{D}$ in subdictionaries. Our ultimate goal is to have as few subdictionaries as possible, with atoms within each subdictionary that are as similar (correlated) as possible, and atoms from different subdictionaries as different (uncorrelated) as possible. We propose a clustering approach that starts from an existing dictionary and endows it with a tree structure $\mathcal{T}$, with nodes $t_i \in \mathcal{T}$. The subdictionaries are seen as clusters of atoms, and the associated molecules are the centroids of each cluster. Each node $t_i = \{c_i, m_i\}$ of the tree is associated to a list c_i containing the indices of its children and to a molecule m_i representing these children through (7). A leaf node t_i is associated to an original atom from the dictionary $\mathcal{D}$, and c_i contains the index of that atom in $\mathcal{D}$. The root node of the tree is labeled t_0 and has no associated molecule. See Fig. 6 for an illustration of these notations.

In general, two different clustering approaches can be chosen: i) a top-down approach that tries to divide the *reducible* dictionary (or subdictionary) into subdictionaries, that satisfy the similarity constraints, and ii) a bottom-up approach that groups similar atoms/molecules together as long as similarity constraints are satisfied. A top-down approach using constraints on similarity has been introduced in [22] and is called *diametrical clustering*. This algorithm was developed for gene clustering to fit an observation stating that genes with anti-correlated expression patterns can be functionally similar. The same observation is true for a dictionary approach of signal decomposition: two anti-correlated atoms have the same behavior as they capture the same structure.

In this paper, we will follow a bottom-up approach, which consists in grouping nodes, starting from atoms, to create new nodes and molecules. The bottom-up approach is better appropriate to the clustering of arbitrary dictionaries, since the number of clusters does not need to be known in advance. The bottom-up approach presented here sets the cardinality k of each cluster. Algorithm 1 presents the method. Initially, it creates nodes containing the atoms from a dictionary $\mathcal{D}$. The recursive part consists in finding groups $\{G\}$ of nodes that can be merged. A set Ω_G of molecules is associated to G. We propose a *weak* and a *strong* rule. The *weak* version defines $\Omega_G = \{m_i\}_{i \in G}$, while for the *strong* decision rule, Ω_G contains the molecules associated to the leaf nodes that are the descendants of the different nodes of G (i.e., $\Omega_G \subset \mathcal{D}$). The set of nodes G is merged if $\max_{\substack{i,j \in G \\ i \neq j}} d(m_i, m_j) < \delta$, where δ is an *a priori* fixed threshold.

Algorithm 1 Tree Creation by grouping

INPUT: $\mathcal{D} = \{g_i\}$, the dictionary.

k, the cardinality of the clusters.

δ, the grouping threshold.

OUTPUT: $\mathcal{T} = \{t_i\}$, the tree.

ALGORITHM:

$c_i = \emptyset, \forall 0 < i \leq |\mathcal{D}| + 1$, leaf nodes have no children.

$m_i = g_{i-1}, \forall 0 < i \leq |\mathcal{D}| + 1$, molecules at leaf nodes.

$t_i = \{c_i, m_i\}$, leaf nodes of the tree.

$\mathcal{T} = \{t_i\}$, initial tree.

$L = \{i\}_{i=1}^{|\mathcal{D}|+1}$, list of candidates for grouping.

$G = \arg\max_{G \subset L, |G|=k} \lambda_{\Omega_G}$, subset of nodes to group.

while $1 - \lambda_{\Omega_G}^2 < \delta$ **do**

$m_{|\mathcal{T}|+1} = m_{\Omega_G}^{opt}$, create the new molecule.

$c_{|\mathcal{T}|+1} = G$, list of children.

$L = L \setminus G$, remove grouped nodes from the list.

$L = L \cup \{|\mathcal{T}| + 1\}$, add index of new node to the list.

$t_{|\mathcal{T}|+1} = \{c_{|\mathcal{T}|+1}, m_{|\mathcal{T}|+1}\}$, create new node.

$\mathcal{T} = \mathcal{T} \cup t_{|\mathcal{T}|+1}$, add the new node to the tree.

$G = \arg\max_{G \subset L, |G|=k} \lambda_{\Omega_G}$, subset of nodes to group.

end while

$m_0 = 0$, no molecule at root node.

$t_0 = \{L, m_0\}$, list of nodes at first level.

$\mathcal{T} = \mathcal{T} \cup t_0$, add root node to the tree.

Finding the best group of k nodes is still a combinatorial problem, but it can be easily solved for small values of k (our results are based on trees created with $k = 2$). The tree can be constructed off-line, without penalizing the pursuit algorithm. Fig. 2 illustrates the construction of a binary tree, for a dictionary of 12 random vectors. The most similar atoms are paired together, until the algorithm reaches level 1 with three molecules, which are too incoherent to be further clustered.

IV. Tree-Based Pursuit Algorithm

A. Tree-Based Search

In a sense, a single iteration of Matching Pursuit can be seen as a classification problem where each atom corresponds to a class of signals. Its aim becomes to successively map the residual signal to a class according to a given distance measure. When considering the greedy approximation problem as an iterative classification problem, the tree structure can be used to divide the decision into smaller steps in a manner similar to a decision tree. Matching Pursuit simply tries all possibilities to find the best class. The use of a hierarchical structure allows to discard an important part of the dictionary at each node. In the following, we describe a practical implementation of this technique, the Tree-Based Pursuit algorithm. Like Matching Pursuit, the proposed algorithm iteratively searches for a good atom to approximate a residual signal $R^n f$. Instead of testing all possible atoms from $\mathcal{D}$, Tree-Based Pursuit uses the tree structure $\mathcal{T}$ that groups similar atoms in the same subtree. The search starts at the root node and goes down through the tree until a leaf node is reached. At each node, the algorithm chooses the child whose molecule best approximates the signal (i.e., the one that leads to the highest amplitude of the scalar product with the residual).

In practice, a dictionary $\mathcal{D}$ is often built using several generating functions, which are translated to different positions in the signal space, e.g., in time or space. Let T_p be the operator that translates a generating function at position p on the support of the atom and leaves the energy unchanged. Tree-Based Pursuit is described by Algorithm 2 for dictionaries built using generating functions that can be translated at any place on the support of the signal to approximate. However, it remains valid for any kind of dictionary. For dictionaries that do not explicit the translation of atoms, the search of the optimal position is simply discarded.

Algorithm 2 Tree-Based Pursuit algorithm

INPUT: $\mathcal{T} = \{c_i, m_i\}$, the tree structured dictionary

σ, the size of the local search window

f, the signal to approximate.

OUTPUT: $\{g_n\}$, the set of chosen atoms

$\{a_n\}$, the set of corresponding projections.

INITIALIZATION: $R^0 f = f, n = 0$

repeat

$[p_{opt}, c_{opt}] = \arg\max_{p, c \in c_o} |\langle R^n f, T_p m_c \rangle|$

while $|c_{opt}| > 1$ **do**

$[p_{opt}, c_{opt}] = \arg\max_{p, c \in c_{opt}} |\langle R^n f, T_p m_c \rangle|,$
$|p - p_{opt}| < \sigma$

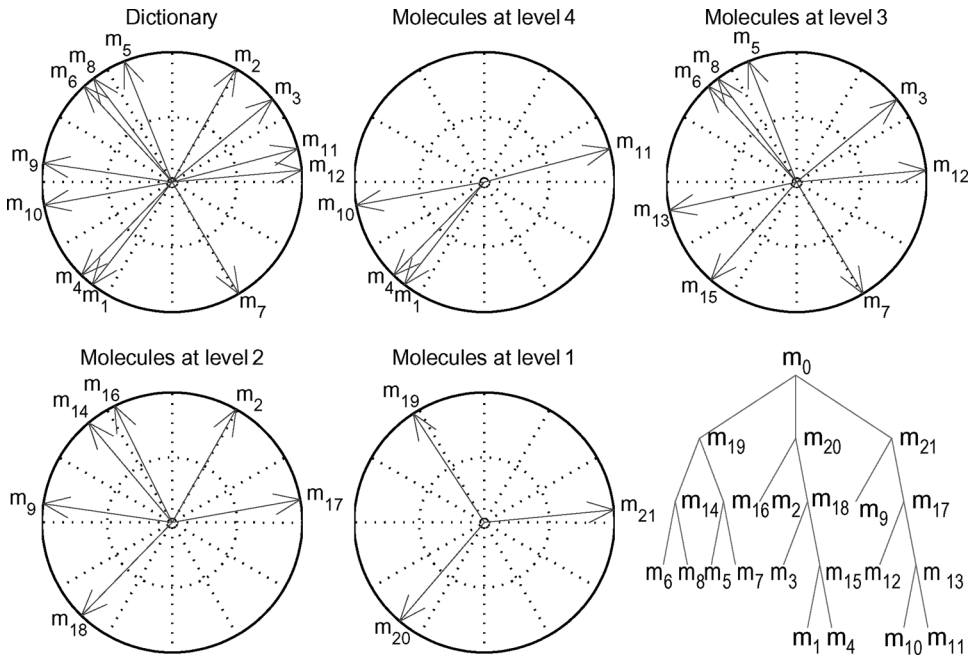


Fig. 2. Creation of a tree on top of a bidimensional (2D) dictionary. The upper-left part shows all atoms in the dictionary. The bottom-right part summarizes the structure of the tree. The other parts correspond to the molecules or atoms present at the different levels of the tree.

end while

$g_{n+1} = T_{p_{opt}} g_{c_{opt}}$

$a_{n+1} = \langle R^n f | g_{n+1} \rangle$

$R^{n+1} = R^n f - a_{n+1} g_n$

$n = n + 1$

until Stop condition is met.

At the root node, the scalar products between the residual $R^n f$ and the molecules of the nodes at the first level of the tree are computed. This step is equivalent to Matching Pursuit using the molecules of the first tree level as dictionary. When using dictionaries built using generating functions, the initial step also gives the position of the best molecule. It can be considered as an energy localization phase. Note that in our case, this localization method is particularly efficient, since molecules really represent the kind of features the dictionary is able to catch. The scalar products between the residual and the molecules of the candidate nodes are computed locally, around the position of the molecule, in a search window of size σ. The traversal is over when the algorithm reaches a leaf node. The information about the position and the node of the tree uniquely identifies an atom from the dictionary $\mathcal{D}$. The algorithm goes on until a stopping criterion is met. It could be a predetermined number of atoms, or a threshold on the residual energy.

The choice of the size of the search window σ should depend on the *shape* of the atoms. As a molecule represents its associated subtree, it is reasonable to think that the best position found for a molecule is near the best positions of its children. The search window should reflect the decay of the correlation of the molecules and translated versions of their children (i.e., the decreasing of the cross correlation).

B. Complexity Analysis

The complexity of the proposed algorithm highly depends on the structure of the tree. In order to be able to evaluate the complexity of Tree-Based Pursuit, let us first make some hypothesis about the tree. Assume that the number of children per node is a constant k, except for the root node, which has $|c_0|$ children. A tree generated by the algorithm proposed in Section III-B fulfills these constraints. Let us also suppose that the tree is balanced, meaning that the length of the longest path differs at most by 1 from the length of the shortest path. It ensures that the maximum length of the paths to the leaves is minimized. Under these assumptions, the length of the longest path is $\lceil 1+\log_k(|\mathcal{D}|/|c_0|)\rceil$, where $|c_0|$ is the number of nodes under the root node and k is the size of the groups formed during the creation of the tree.

Let us first look at the case where the dictionary is not built using translation of generating functions. At the nth iteration, Matching Pursuit would require to compute all scalar products between the atoms of $\mathcal{D}$ and a residual $R^n f$. For a d-dimensional signal, the computation of the scalar product needs d multiplications and $d-1$ additions. Thus, the complexity to find one atom is $\mathcal{O}(|\mathcal{D}|d)$. Tree-Based Pursuit reduces this complexity, since the divide-and-conquer procedure eliminates many possibilities at each level. At the root node, $|c_0|$ scalar products have to be computed. Each other node only requires the computation of k scalar product to find the best child. Thus, the overall complexity is reduced to $\mathcal{O}(|c_0|d + \lceil 1 + \log_k(|\mathcal{D}|/|c_0|)\rceil kd)$.

For dictionaries built using translation of generating functions, the algorithm has not only to find the best node but also the corresponding position in the signal. At the root node, a full search is done, which is equivalent to Matching Pursuit using the reduced dictionary made of the molecules of the nodes that are located at the first level of the tree. During the rest of the traversal of the tree, a local search in a window of size σ is performed. Let $\tilde{\sigma}$ be the maximal amount of possible positions to test in a window of size σ.

A commonly used and smart implementation of Matching Pursuit consists in using a fast Fourier transform to compute all scalar products with shifted atoms. Such an implementation has a complexity of $\mathcal{O}(|\mathcal{D}|d\log d)$ to find the best atom. Each local search has a complexity of $\mathcal{O}(\tilde{\sigma}d)$. Putting it all together, the complexity of the proposed algorithm for finding the best atom is

$$\mathcal{O}\left(|c_0|d\log d+\left(\left\lceil\log_k\frac{|\mathcal{D}|}{|c_0|}\right\rceil\right)\tilde{\sigma}d\right). \tag{14}$$

For reasonable values of the search window σ, the descent trough the tree is negligible regarding the complexity of the initial step. The complexity of Tree-Based Pursuit is less affected by the growth of the dictionary, while the complexity of Matching Pursuit increases linearly. However, it has to be noticed that the approximation rate of the Tree-Based Pursuit algorithm decreases when the number of children of the root becomes smaller relatively to the size of the dictionary, as discussed in the next section.

V. Consistency Analysis

A. From Redundant to Block-Incoherent Dictionaries

Most theoretical results in the field of sparse approximations rely on (quasi-) incoherent dictionaries. Only little work has been done on highly redundant dictionaries despite their interesting properties for approximation and compression. Interestingly, endowing the dictionary $\mathcal{D}$ with a tree structure can also be thought of as a way to artificially lower the coherence. During the creation of the tree, our clustering algorithm minimizes the coherence among molecules. Thus, even for highly correlated dictionaries, the theoretical results relying on small coherence most probably remain valid at the granularity level of the molecules. In this section, we build upon this idea and analyze the theoretical approximation performance of the algorithm.

The creation of molecules relies on having subdictionaries containing highly correlated atoms. The following definition summarizes the constraints ensuring the favorable cases.

Definition 1: A subdictionary $\mathcal{D}_\Lambda$ is reducible to a molecule m_Λ, which is called *representative* if

- its minimal coherence λ_Λ is strictly positive;
- $\min_{k\in\Lambda}|\langle g_k, m_\Lambda\rangle|\geq\lambda_\Lambda$;
- $m_\Lambda\in\operatorname{span}\{\mathcal{D}_\Lambda\}$.

In Section III-A, we have defined an optimality criterion for a molecule relying on the measure of a mean distance that defines a convex set. This implies that standard optimization tools can be applied to find an optimal molecule. Let us now measure the adequation of a molecule regarding the subdictionary it represents by

$$\sigma_\Lambda=\min_{i\in\Lambda}|\langle m_\Lambda, g_i\rangle|. \tag{15}$$

The definition of a representative molecule therefore implies that the minimal coherence of a molecule regarding its associated subdictionary is such that $\sigma_\Lambda\geq\lambda_\Lambda$. In other words, adding the molecule m_Λ to its subdictionary $\mathcal{D}_\Lambda$ does not change the minimal coherence. This condition defines a subspace of $\operatorname{span}\{\mathcal{D}\}$ where the molecule is allowed to exist.

B. Covering Conditions

Since the search is organized along a tree structure, it has to be ensured that the restructured dictionary is still able to cover the full space of the input signal. Tropp has defined a measure of the covering radius of a dictionary [23] as

$$\operatorname{cover}(\mathcal{D})=\max_{s\neq0}\min_{i\in\Gamma}\sqrt{1-\left(\frac{|\langle g_i,s\rangle|}{||g_i||_2||s||_2}\right)}. \tag{16}$$

The relation between the covering radius and the structural redundancy β of a dictionary given in (4) is straightforward. The covering is minimal when β is maximal

$$\operatorname{cover}(\mathcal{D})=\sqrt{1-\beta^2}. \tag{17}$$

We now set the conditions that are necessary for the clustered dictionary to fully cover the signal space. In particular, it is necessary that the molecules at the first level under the root node cover the signal space. Note that such a requirement is naturally met at other levels of the tree: by the bottom-up construction, each molecule is indeed representative of the related subdictionary. The following lemma states a minimal condition on the molecules to ensure that a signal f, which can be represented using atoms from $\mathcal{D}$, can also be represented using only molecules. More precisely, it provides a minimal condition, given the parameter β of $\mathcal{D}$, to ensure that the molecules at the first level of the tree cover the same span as the dictionary itself.

Lemma 1: If the collection of subdictionaries $\{\mathcal{D}_{\Lambda_i}, i=1,\ldots,K\}$ forms a partition of $\mathcal{D}$ and the associated molecules are representative, then $\operatorname{span}\{m_{\Lambda_i,i=1,\ldots,K}\}=\operatorname{span}\mathcal{D}$ if

$$\sigma_{\Lambda_i}>\beta+2\sqrt{1-\beta}-1,\quad\forall i. \tag{18}$$

Proof: Since the molecules are by construction in the span of their associated subdictionaries, the span of the molecules is within the span of the original dictionary $\mathcal{D}$

$$\operatorname{span}\{m_{\Lambda_i},\ i=1,\ldots,K\}\subseteq\operatorname{span}\mathcal{D}. \tag{19}$$

In order to ensure that the span of the molecules covers the span of the dictionary, it remains to show that the orthogonal complement of $\operatorname{span}\{m_{\Lambda_i}, i=1,\ldots,K\}$ in $\operatorname{span}\mathcal{D}$ is actually empty.

Let $f\neq0$ be a signal lying in the span of the dictionary $\mathcal{D}$. Without loss of generality, let f be a unit norm signal. In addition, let the atom $g_0\in\mathcal{D}$ carry the best one-term approximation of the signal, i.e., $|\langle f,g_0\rangle|=\max_{i\in\Gamma}|\langle f,g_i\rangle|$. Suppose the atom g_0 belongs to the subdictionary $\mathcal{D}_{\Lambda_0}$ which is represented by the molecule m_{Λ_0}. The distance between f and m_{Λ_0} can be bounded by

$$||f-m_{\Lambda_0}||_2\leq||g_0-m_{\Lambda_0}||_2+||f-g_0||_2. \tag{20}$$

Without loss of generality, assume that $\langle f, g_0 \rangle > 0$ and $\langle m_{\Lambda_0}, g_0 \rangle > 0$, by construction of the clustered dictionary. Recall that the direction of an atom does not have any impact in terms of approximation rate, so that we can assume positive correlation values. Since all vectors have unit norm, it is possible to rewrite (20) as

$$\sqrt{1 - \langle f, m_{\Lambda_0} \rangle} \leq \sqrt{1 - |\langle g_0, m_{\Lambda_0} \rangle|} + \sqrt{1 - |\langle f, g_0 \rangle|}. \quad (21)$$

We can also bound the last scalar product by

$$|\langle f, g_0 \rangle| \geq \beta. \quad (22)$$

Using (15) and (22), we obtain

$$\sqrt{1 - \langle f, m_{\Lambda_0} \rangle} \leq \sqrt{1 - \sigma_{\Lambda_0}} + \sqrt{1 - \beta}. \quad (23)$$

We would like to show that the projection of the signal f onto the molecule that is representative of the subdictionary $\mathcal{D}_{\Lambda_0}$ is never null. In other words, we would like to ensure that, if the best one-term approximation of f lies within $\mathcal{D}_{\Lambda_0}$, then the signal f is never orthogonal to the molecule m_{Λ_0}. Imposing that m_{Λ_0} is not orthogonal to f is equivalent to require that $\sqrt{1 - \langle f, m_{\Lambda_0} \rangle} \neq 1$. Using (23), this holds whenever

$$\sqrt{1 - \sigma_{\Lambda_0}} + \sqrt{1 - \beta} < 1 \quad (24)$$

which leads to

$$\sigma_{\Lambda_0} > \beta + 2\sqrt{1 - \beta} - 1. \quad (25)$$

If this condition is verified, it ensures that $\langle f, m_{\Lambda_0} \rangle > 0$ whenever the signal $f \in \mathcal{D}$ has a component along $g_0 \in \mathcal{D}_{\Lambda_0}$.

If the condition given in (25) is true for all subdictionaries of the first level of the tree (that form a partition of $\mathcal{D}$), then $\nexists f \in \operatorname{span} \mathcal{D}$ such that $\langle f, m_{\Lambda_i} \rangle = 0, \forall i$. Hence, $\operatorname{span} \mathcal{D} = \operatorname{span}\{m_{\Lambda_i}, i = 1, \ldots, K\}$. ■

When the formula of Lemma 1 holds, we can treat the set of molecules as a genuine dictionary. Let $\{\mathcal{D}_{\Lambda_i}\}$ form a partition of $\mathcal{D}$ and let $\mathcal{D}_M = \{m_{\Lambda_i}\}$ be the dictionary made of the molecules. This dictionary has an associated characteristic parameter β_M. For any signal $f \in \operatorname{span} \mathcal{D}$, we thus can lower-bound the projection on the molecules

$$\max_{m_i \in \mathcal{D}_M} |\langle f, m_i \rangle| \geq \beta_M \|f\|. \quad (26)$$

This also leads to

$$\max_{m_i \in \mathcal{D}_M} |\langle f, m_i \rangle| \geq \beta_M \max_{i \in \Gamma} |\langle f, g_i \rangle|. \quad (27)$$

Obviously, $\beta_M \leq \beta$. It would also be interesting to characterize the (cumulative) coherence of the dictionary. In the next subsection, we show that Tree-Based Pursuit benefits from representative molecules and is able to identify the signal at the granularity level of its representative subdictionaries.

C. Recovery Conditions

In the previous section, we have set the conditions for the tree-structured dictionary to cover the span of the original dictionary $\mathcal{D}$. We now derive a condition for the search algorithm to choose consistent molecules given a signal f, that is a linear combination of vectors in $\mathcal{D}$. Let the signal f have an exact representation using atoms from the dictionary $\mathcal{D}$

$$f = \sum_{\substack{i \\ g_i \in \Omega}} a_i g_i \quad (28)$$

where Ω is a subset of indices.

Tropp [18] derived a minimal condition that guarantees that Orthogonal Matching Pursuit and Basis Pursuit recover Ω, where Ω is the smallest set such that (28) holds. We now show that this recovery condition holds true for the Tree-Based Pursiut at the level of representative molecules of a very redundant dictionary. Let Φ be a matrix whose columns contain the atoms that are in Ω. The signal can be written as $f = \Phi A$, where the vector A contains the weights a_i relative to atoms in Ω.

Let f_k be the approximation of f after k iterations of Tree-Based Pursuit. We write $f_k = \Psi_k A_k$, where Ψ_k contains the atoms found by Tree-Based Pursuit and A_k the corresponding weights. Since we do not impose any restriction on the cumulative coherence of the dictionary, we cannot directly apply the results developed in [18], which typically use the cumulative coherence for an estimation of the exact recovery condition. We do not necessarily intend to recover exactly the atoms in Φ, but we rather want to ensure that the atoms found by Tree-Based Pursuit are close to the optimal ones (and, in particular, in the same subdictionaries). We focus on the decision taken by Tree-Based Pursuit at the root of the tree and want to guarantee that it never chooses a node that does not contain at least one atom from Ω in its subtree.

If after k iterations of Tree-Based Pursuit, the decisions at the root node are always *correct*, no atom from Ψ_k is located in a subtree that does not contain an atom from Ω. Let Φ_k be a matrix containing the distinct atoms from Φ and Ψ_k. Similarly, the index set Ω_k is the set of atoms present in Φ_k. As has been discussed, due to the bottom-up construction of the tree, the critical step consists of choosing the correct molecules at the first level of the tree. Assume once again that the subdictionaries $\{\mathcal{D}_{\Lambda_i}\}$ form a partition of the dictionary, and that each subdictionary is reduced to a molecule m_{Λ_i}. We say that m_{Λ_i} is a *good* molecule if it represents at least one atom participating in f. The matrix M_G contains all *good* molecules in its columns. Similarly, M_B contains the *bad* molecules of the first tree level in its columns. The following theorem states the necessary conditions for the Tree-Based Pursuit algorithm to choose the correct molecule at the first level of the tree.

Theorem 1: If the hypothesis of Lemma 1 holds true, then Tree-Based Pursuit chooses a *good* molecule at the first level of the tree, at iteration k, if

$$\max_{m \in M_B} \left\| \Phi_k^+ m \right\|_1 < \beta_M \quad (29)$$

where Φ_k^+ is the Moore–Penrose pseudoinverse of Φ_k. ♢

Proof: The proof of Theorem 1 is an extension of Tropp's Recovery Condition [18] and we provide here only the differences to the proof given in [18]. Assume that at each iteration $i < k$, Tree-Based Pursuit has chosen a good molecule at the first level of the tree. It has to be noted that the atoms in Ω all belong to subtrees of nodes associated to *good* molecules. Under the assumption that we have chosen only *good* molecules, the atoms in Ω_k also belong to subtrees of nodes associated to *good* molecules. The residual signal $r_k = f - f_k$ can be exactly represented as $r_k = \Phi_k A_k^R$, where A_k^R contains appropriate weights. The vectors $M_B^* r_k$ and $M_G^* r_k$ list all possible scalar products of the residual r_k with, respectively, the *bad* and *good* molecules (M^* stands for the adjoint of M). The aim is to find a condition that ensures that the current step also recovers a good molecule. A *good* molecule is, therefore, chosen by the search algorithm if

$$\frac{\left\|M_B^* r_k\right\|_\infty}{\left\|M_G^* r_k\right\|_\infty} < 1. \tag{30}$$

If the hypothesis of Lemma 1 holds true, then the lower bound on the projection on the molecules given in (27) can be used to further develop the left-hand side of the previous equation. We can write

$$\frac{\left\|M_B^* r_k\right\|_\infty}{\left\|M_G^* r_k\right\|_\infty} \leq \frac{\left\|M_B^* r_k\right\|_\infty}{\beta_M \left\|\Phi_k^* r_k\right\|_\infty}. \tag{31}$$

The last steps of the proof are analog to Tropp's recovery condition [18] proof, which finally leads to the conservative condition

$$\frac{1}{\beta_M} \max_{m \in M_B} \left\|\Phi_k^+ m\right\|_1 < 1. \tag{32}$$

One could further apply Tropp's estimate of (32) in terms of the cumulative coherence [18] of the set of molecules, in order to obtain a condition that would depend on the set of molecules only (and not on the unknown optimal set M_B). This estimate requires the set of molecules to be quasi-incoherent. Note that this is very likely to be the case here, but it would even be better to actually prove how μ_1 behaves as we climb up the granularity level of the tree. Finally, note that the recovery condition itself holds at a coarser level than in previous works: Tree-Based Pursuit recovers only molecules that are involved and not the individual atoms. On the other hand, this allows to shift the incoherence constraint to the molecules and work with a possibly highly correlated dictionary. ■

VI. Experimental Results

A. Unidimensional Signals

This section now illustrates the Tree-Based Pursuit algorithm, and compares its performances to Matching Pursuit. We present results for both unidimensional and bidimensional signals (i.e., images). Often, in practice, the dictionaries are built using shift-invariant generating functions that can be translated at any place in the support of the signal. In order to illustrate the generality of the method, let us first consider a dictionary $\mathcal{D}$ made of random

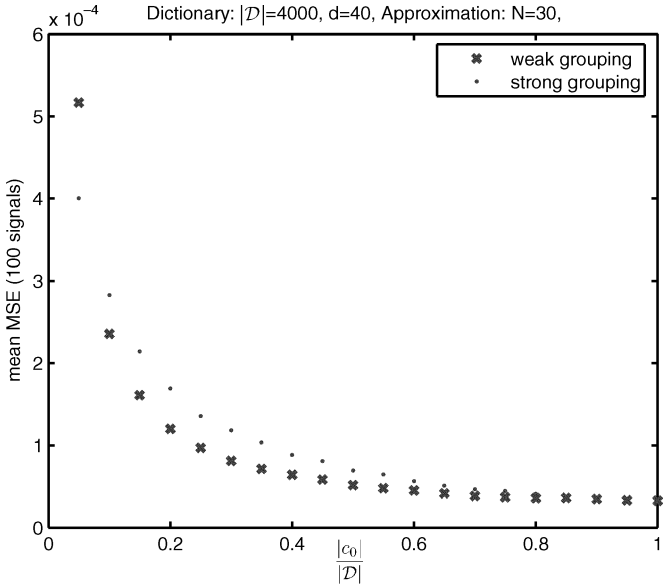


Fig. 3. Mean approximation error of random signals using a random dictionary.

atoms. It represents the worst case for the construction of the tree. The size of the dictionary is $|\mathcal{D}| = 4000$. The atoms are real vectors of size $d = 40$. Thus, the dictionary is 100 times redundant. Each sample has uniform probability between 0 and 1 and the atoms are normalized to have unit energy. The complexity of Tree-Based Pursuit mostly depends on the number of molecules under the root node. Thus, 20 trees have been generated for different numbers of nodes at the first level of the tree, $|c_0|$ ranges from 200 up 4000 by steps of 200. Two sets of trees have been generated depending on the decision rule to create the molecules (*weak* or *strong*).

In order to evaluate the performances of Tree-Based Pursuit, 100 test signals have been randomly generated using the same procedure as for the dictionary. Each individual signal has been approximated using $N = 30$ atoms from the dictionary $\mathcal{D}$. Fig. 3 illustrates the mean approximation error achieved. It has to be noticed that for $|c_0|/|\mathcal{D}| = 1$, Tree-Based Pursuit is equivalent to Matching Pursuit. The ratio $|c_0|/|\mathcal{D}|$ reflects approximately the complexity of Tree-Based Pursuit regarding Matching Pursuit. The error decreases as the number of nodes at the first level of the tree increases.

Random dictionaries are seldom used to approximate signals. Consider now a dictionary made of real Gabor functions, as in [2]

$$g_{u,s,\xi,\phi}(t) = c_{u,s,\xi,\phi}\, g\left(\frac{t-u}{s}\right) \cos\left(2\pi\xi(t-u) + \phi\right) \tag{33}$$

with

$$g(t) = \frac{1}{\sqrt{s}} e^{-\pi t^2}. \tag{34}$$

The normalizing constant $c_{u,s,\xi,\phi}$ is such that the corresponding atom is of unit energy. The parameter u is the position, s is the scale, ξ represents the frequency, and ϕ is the phase. Fig. 4 presents three atoms of such a dictionary, and the representative molecule, which is the eigenvector associated to the biggest

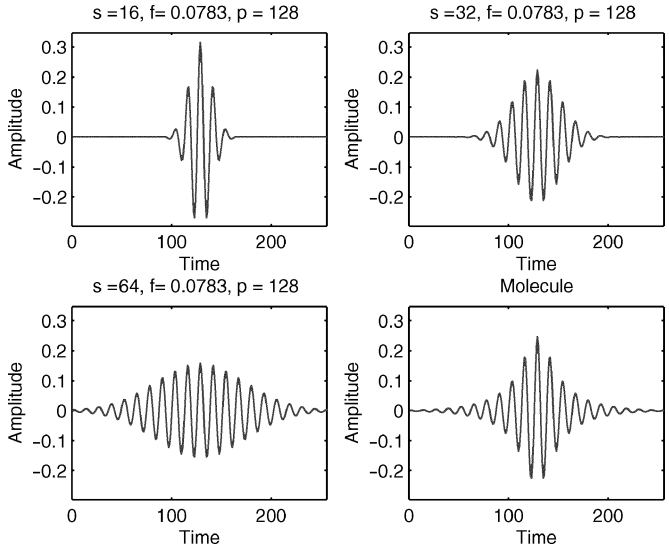


Fig. 4. Representing a group of atoms by a molecule. From top-left to bottom-left: Real Gabor atoms with same frequency f and position p but with different scales s. Bottom-right: the molecule is the eigenvector associated to the biggest eigenvalue of $A_\Lambda A_\Lambda^*$.

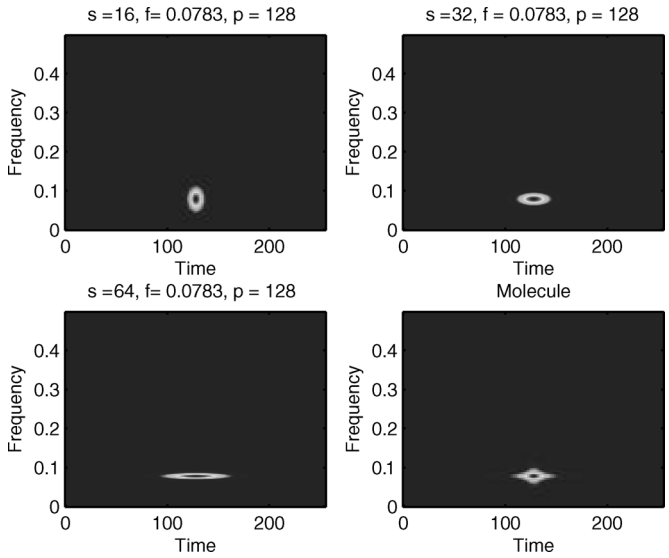


Fig. 5. Time–frequency plane of the atoms and the molecule presented in Fig. 4.

eigenvalue of $A_\Lambda A_\Lambda^*$, as discussed in Section III-A. Fig. 5 shows the time–frequency representations of the atoms and the molecule of Fig. 4. We can observe that the molecule indeed provides global information about all the atoms, and nicely summarizes the characteristics of the subdictionary.

In our experiments, we used a dictionary built on real Gabor atoms with size 256, where the phase ϕ is set to zero in (33). We used 200 different frequencies uniformly spread over the interval of normalized frequencies $[0\,0.5]$ and the scales are dyadic. The overall size of the dictionary is 1600 (i.e., 1600 times redundant), without taking into account all possible shifts, which are not considered during construction of the trees. The translation parameters are, however, computed by the search algorithm. Fig. 6 shows a part of an example tree

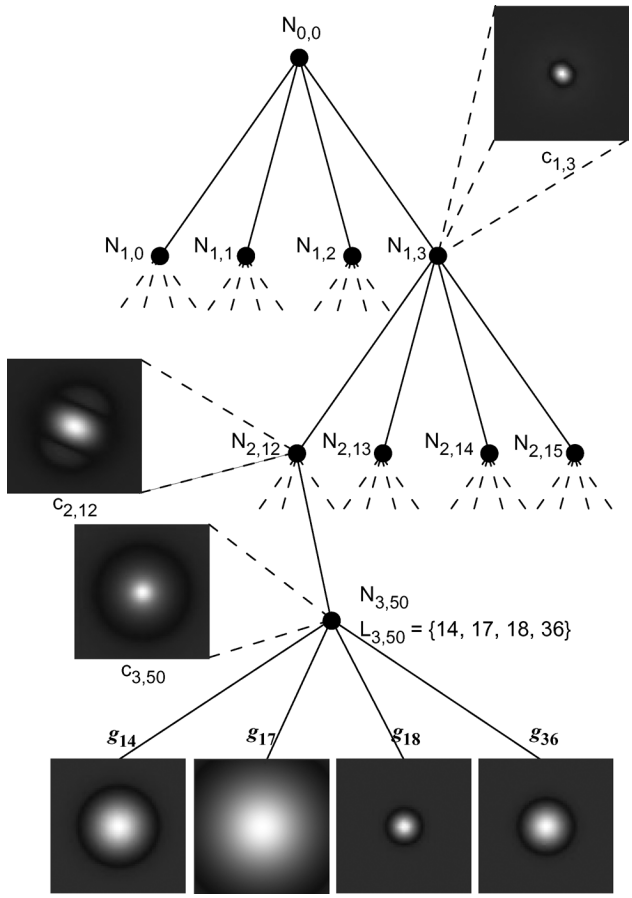


Fig. 6. Tree structure on top of a multiscale Gabor dictionary.

built on the multiscale Gabor dictionary, where we only use *centered* versions of the atoms.

We now compare the performance of the Tree-Based Pursuit algorithm, for different tree constructions, with Matching Pursuit. The reference Matching Pursuit computes all possible convolutions in the frequency domain by using a fast Fourier transform. Tree-Based Pursuit uses the same Matching Pursuit implementation at the initial step for the first level of the tree. This technical choice makes it possible to compare the complexity of both algorithms.

Numerous tree structured dictionaries have been generated for different values of the distance threshold δ, using the grouping strategy given in Algorithm 1, with a *weak* decision rule for clustering of the atoms. We selected three different trees, with δ values $[0.36\,0.75\,0.99]$. This corresponds, respectively, to minimal values of $[0.8\,0.5\,0.1]$ of the scalar product between two molecules to form a cluster. The value of δ determines $|c_0|$, the number of nodes at the first level of the tree. In these particular cases, the trees respectively present 240, 51, and 11 nodes at the first level under the root node. Moreover, under the assumption that the trees are balanced, their average depth would be $3, 5$, and 8 in the order of increasing values for δ. Fig. 7 presents the molecules at the first level of the tree created with $\delta = 0.99$, while Fig. 8 illustrates the corresponding time–frequency diagrams.

We can now compare the approximation performance, and the computational complexity of Tree-Based Pursuit, as opposed to

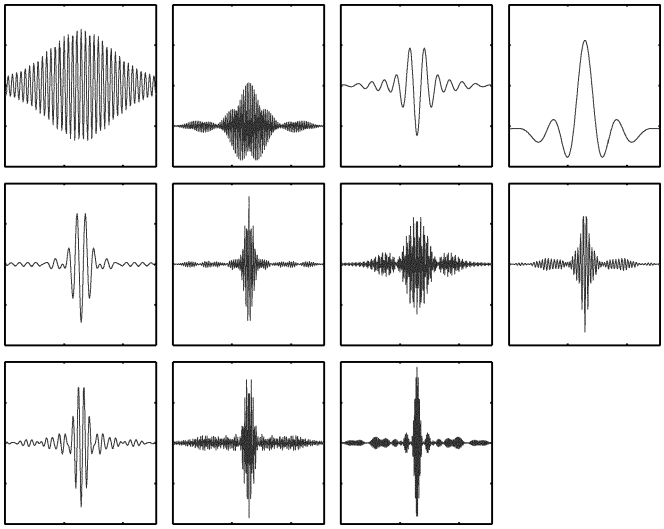


Fig. 7. Molecules associated to the nodes located at the first level of the tree created with a grouping threshold $\delta = 0.99$.

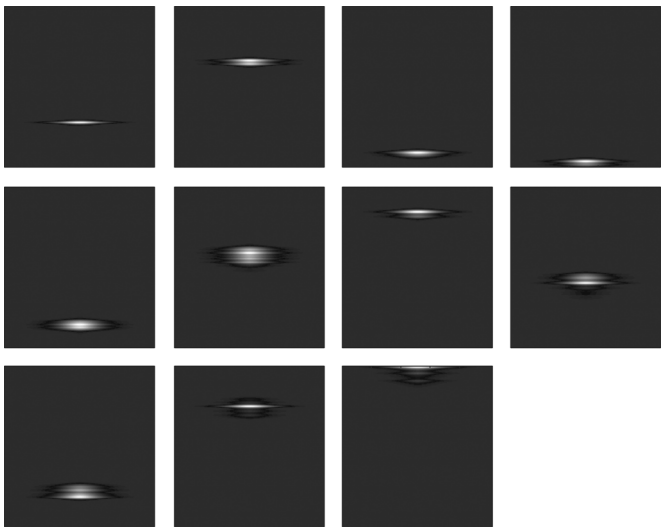


Fig. 8. Time–frequency distributions corresponding to the molecules exhibited by Fig. 7.

Matching Pursuit. The size of the search window σ is 5. Fig. 9(a) compares the mean-squared error obtained with Matching and Tree-Based Pursuit using the different trees defined above. The results have been averaged over 100 zero-mean random signals with Gaussian distribution of unit variance. When using trees created with small values for δ, the results are very close to the reference Matching Pursuit. Recall that small values of δ impose very strict constraints on the clustering of atoms and molecules, which may result in a large number of molecules at the first level of the tree. The computation time depends on δ, since computation time depends on the amount of nodes at the first level of the tree $|c_0|$, as discussed in Section IV. Experimental results confirm the complexity analysis in Fig. 9(b). Indeed, if we compute a linear approximation of the Tree-Based Pursuit computation time as a function of $|c_0|$, in a mean-squared sense, it intersects the Matching Pursuit computation time around $|c_0| = 1618$ (the dictionary contains 1600 atoms). This shows that most of the complexity of Tree-Based Pursuit lies in the full search at the first level of the tree; after this initialization step, the cost of the traversal of the tree can be considered as negligible regarding the initial search.

B. Extension to Multidimensional Signals

This subsection extends the analysis of the Tree-Based Pursuit algorithm to images. Reduction of the complexity in the case of multidimensional signal is even more crucial than for unidimensional signals. We use a dictionary that is built on Gaussian generating functions that are scaled, rotated, and translated [24]. The first generating function is a Gaussian as given by (35), which suits well the task of capturing the low-frequency parts of natural images. The second generating function, given by (36), is made of a Gaussian in one direction and its second derivative in the other direction. It has good ability to capture edges in images and is spatially and frequencially well located.

$$g_1(x,y) = \frac{1}{\sqrt{\pi}} \exp -(x^2+y^2) \tag{35}$$

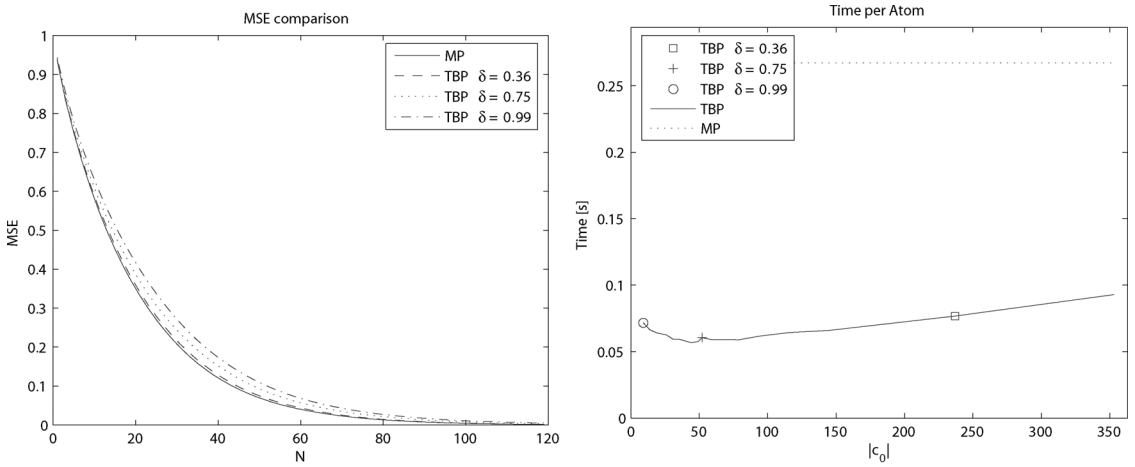
$$g_2(x,y) = \frac{2}{\sqrt{3\pi}}(4x^2-2) \exp -(x^2+y^2). \tag{36}$$

In our experiments, the atoms using g_2 as generating function have translation parameters that take any positive integer value smaller than the size of the image. The rotation parameter varies by increments of $\pi/18$. The scaling parameters are uniformly distributed on a logarithmic scale from one up to an eighth of the size of the image, with a resolution of one third of octave. The scaling along the second derivative part is always smaller. For the pure Gaussian atoms, the translation parameters can take the same values, the scaling is isotropic and varies from $1/32$ to $1/4$ of the size of the image on a logarithmic scale with a resolution of one third of an octave. Due to isotropy, rotation is obviously useless for this kind of atoms.

The dictionary is made of 514 generating functions that can be placed anywhere on the support of the signal (i.e., the dictionary is 514 times redundant). Algorithm 1 was used with a *strong* decision rule to create the molecules; 20 different trees have been created with different values of δ, chosen to create trees having values for $|c_0|$ ranging from $0.05|\mathcal{D}|$ up to $\mathcal{D}$ by steps of $0.05|\mathcal{D}|$.

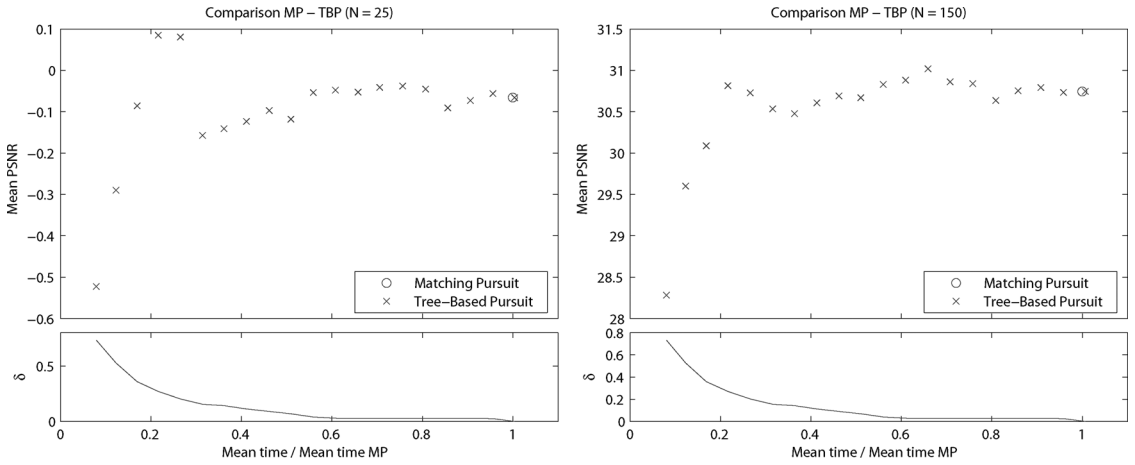
Fig. 10 presents a comparison of the approximation error of Tree-Based Pursuit and Matching Pursuit averaged over 120 natural images of size 16×16. The mean time to find one atom has also been computed. As explained in Section IV, the computational time mostly depends on the number of nodes at the first level of the tree. Fig. 10 shows that for a computational time of about 20% of the reference Matching Pursuit, the errors are comparable. When using only a few atoms, it also happens that Tree-Based Pursuit performs better. The used dictionary represents a favorable case for the creation of a tree structure as the atoms corresponding to edges in the same direction can be efficiently represented by a unique element without loosing the edge detection ability. This dictionary has a flavor of the ideal case represented by block incoherent dictionaries.

The lower parts of the results presented in Fig. 10 present the value of δ used for the creation of the corresponding trees. Tree-Based Pursuit is equivalent to Matching Pursuit when using



(a) (b)

Fig. 9. (a) Comparison of the error produced by the proposed algorithm when using different bounds for the grouping. (b) Comparison of the complexity between Matching Pursuit and Tree-Based Pursuit, when using different values for δ during the creation of the tree structure.



(a) (b)

Fig. 10. Performance of Tree-Based Pursuit compared to Matching Pursuit for different trees. The x axis corresponds to the mean time per atom divided by the mean time per atom for Matching Pursuit. The lower part of the figures presents the value of δ used during the creation of the tree. (a) PSNR achieved using 25 atoms. (b) PSNR achieved using 150 atoms.

trees created with a value of $\delta = 0$. However, due to a more complex data structure to handle, the computational time is slightly higher.

VII. Conclusion

This paper has presented a generic algorithm to reduce the computational complexity of pursuit algorithms. Hierarchical clustering of dictionary atoms in molecules has been proposed, as an efficient structuring of large sets of functions. The molecules represent a subdictionary of highly correlated atoms and are used to create a tree structure from an arbitrary highly redundant dictionary. A tree-based pursuit algorithm is then proposed, which exploits the tree structure, resulting in a computational complexity that is significantly lower than the classic pure greedy algorithm. We experimentally showed that the reduction in complexity does not imply a large penalty in approximation rate. It is shown also that Tree-Based Pursuit recovers coarse structures of the signal, even for highly redundant dictionaries, thanks to the hierarchical clustering into sufficiently incoherent dictionaries of molecules. Finally, practical applications are often based on highly redundant dictionaries, whose properties are however poorly studied. On the other hand, the class of *incoherent* dictionaries has been widely studied, but is rarely used in practical applications. Our study tries to bridge that gap, by demonstrating that, from a molecular point of view, it is possible to apply the approximation results for *incoherent* dictionaries to highly redundant dictionaries.

References

[1] G. Davis, S. Mallat, and M. Avellaneda, "Adaptive greedy approximations," *J. Construct. Approx.*, vol. 13, pp. 57–98, 1997.

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

[3] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Scientific Comput.*, vol. 20, no. 1, pp. 33–61, 1999.

[4] D. Redmill, D. Bull, and P. Czerepinki, "Video coding using a fast non-separable matching pursuits algorithm," in *Proc. IEEE Int. Conf. Image Processing*, Oct. 1998, vol. 1, pp. 769–773.

[5] R. Neff and A. Zakhor, "Matching pursuit video coding, i. Dictionary approximation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 1, pp. 13–26, Jan. 2002.
[6] K.-P. Cheung and Y.-H. Chan, "A fast two-stage algorithm for realizing matching pursuit," in *Proc. IEEE Int. Confe. Image Processing*, Oct. 2001, vol. 2, pp. 431–434.
[7] R. Gribonval, "Fast matching pursuit with a multiscale dictionary of gaussian chirps," *IEEE Trans. Signal Process.*, vol. 49, no. 5, pp. 994–1001, May 2001.
[8] C. de Vleeschwouwer and B. Macq, "Subband dictionaries for low-cost matching pursuits of video residues," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 7, pp. 984–993, Oct. , 199.
[9] M. Goodwin and M. Vetterli, "Matching pursuit and atomic signal models based on recursive filter banks," *IEEE Trans. Signal Process.*, vol. 47, no. 7, pp. 1890–1902, Jul. 1999.
[10] P. Schmid-Saugeon and A. Zakhor, "Dictionary design for matching pursuit and application to motion-compensated video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 6, pp. 880–886, Jun. 2004.
[11] Y.-T. Chou, W.-L. Hwang, and C.-L. Huang, "Gain-shape optimized dictionary for matching pursuit video coding," *Signal Process.*, vol. 83, pp. 1937–1943, Sep. 2003.
[12] D. Jiang, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: a survey," *IEEE Trans. Knowledge and Data Engi.*, vol. 16, no. 11, pp. 1370–1386, Nov. 2004.
[13] S. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Systems, Man Cybern.*, vol. 21, no. 3, pp. 660–674, May/Jun. 1991.
[14] S. Cotter and B. Rao, "Application of tree-based searches to matching pursuit," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, May 2001, vol. 6, pp. 3933–3936.
[15] J. A. Tropp, "Topics in sparse approximation," in *Computational and Applied Mathematics*. Austin, TX: Univ. Texas-Austin Press, Aug. 2004.
[16] M. Elad and A. Bruckstein, "A generalized uncertainty principle and sparse representations in pairs of bases," *IEEE Trans. Inf. Theory*, vol. 48, no. 9, pp. 2558–2567, Sep. 2002.
[17] R. Gribonval and M. Nielsen, "Sparse representations in unions of bases," *IEEE Trans. Inf. Theory*, vol. 49, no. 12, pp. 3320–3325, De. 2003.
[18] J. A. Tropp, "Greed is good: algorithmic results for sparse approximation," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.
[19] P. Frossard and P. Vandergheynst, "Redundancy in non-orthogonal transforms," in *Proc. IEEE Int. Symp. Information Theory*, Washington, DC, Jun. 2001, p. 196.
[20] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1985.
[21] L. Peotta, P. Jost, P. Vandergheynst, and P. Frossard, Sparse Approximation With Block Incoherent Dictionaries EPFL, 1015 Ecublens, TR-ITS 2003.007, Dec. 2003.
[22] I. S. Dhillon, E. M. Marcotte, and U. Roshan, "Diametrical clustering for identifying anti-correlated gene clusters," *Bioinformatics*, vol. 19, no. 13, pp. 1612–1619, 2003.
[23] J. A. Tropp, Just Relax: Convex Programming Methods for Subset Selection and Sparse Approximation Univ. Texas at Austin, ICES Rep. 04-04, Feb. 2004.
[24] R. Figueras i Ventura, P. Vandergheynst, and P. Frossard, "Low rate and flexible image coding with redundant representations," *IEEE Trans. Image Process.*, vol. 15, no. 3, pp. 726–739, Mar. 2006.

Philippe Jost received the M.S. degree in communication systems from the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, in 2002.

He is now working toward the Ph.D. degree at the Signal Processing Laboratory, Swiss Federal Institute of Technology (EPFL), Lausanne, where his research focuses on sparse representations.

Pierre Vandergheynst (M'01) received the M.S. degree in physics and the Ph.D. degree in mathematical physics from the Université Catholique de Louvain, Louvain, Belgium, in 1995 and 1998, respectively.

From 1998 to 2001, he was a Postdoctoral Researcher with the Signal Processing Laboratory, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland. He is now Assistant Professor at EPFL where his research focuses on harmonic analysis, sparse representations, and mathematical image processing with emphasis on higher dimensional and complex data processing.

Pascal Frossard (S'96–M'01–SM'04) received the M.S. and Ph.D. degrees, both in electrical engineering, from the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, in 1997 and 2000, respectively.

From 1998 to 2000, he worked with the Signal Processing Laboratory, EPFL, as a Research and Teaching Assistant under a grant from Hewlett-Packard. Between 2001 and 2003, he was a member of the research staff at the IBM T. J. Watson Research Center, Yorktown Heights, NY, where he worked on media compression and streaming technologies. Since April 2003, he has been an Assistant Professor at EPFL. His research interests include image representation and coding, nonlinear representations, visual information analysis, joint source and channel coding, multimedia communications, and multimedia content distribution.

Dr. Frossard has been the General Chair of IEEE ICME 2002 (Lausanne, Switzerland), and member of the organizing or technical program committees of numerous conferences. He has served as Guest Editor of special issues on Streaming Media (IEEE TRANSACTIONS ON MULTIMEDIA), on Media and Communication Applications on General Purpose Processors: Hardware and Software Issues (Journal of VLSI SPSS), and on Image and Video Coding Beyond Standards (Journal of Signal Processing). He is an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA (2004-) and of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (2006-), and he served as a member of the Editorial Board of the *EURASIP Journal of Signal Processing* (2003–2005). Since 2004, he has served as Vice-Chair of the IEEE Multimedia Communications Technical Committee, as a member of the IEEE Multimedia Signal Processing Technical Committee, and of the IEEE Multimedia Systems and Applications Technical Committee. He received the Swiss NSF Professorship Award in 2003, and the IBM Faculty Award in 2005.