SCHOOL OF ENGINEERING - STI
SIGNAL PROCESSING INSTITUTE
Philippe Jost, Pierre Vandergheynst and Pascal Frossard

CH-1015 LAUSANNE

Telephone: +4121 6936874
Telefax: +4121 6937600
e-mail: philippe.jost@epfl.ch

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

TREE-BASED PURSUIT: ALGORITHM AND PROPERTIES

Philippe Jost, Pierre Vandergheynst and Pascal Frossard

Swiss Federal Institute of Technology Lausanne (EPFL)

Signal Processing Institute Technical Report

TR-ITS-2005.013

May 17th, 2005

Submitted to IEEE Transactions on Signal Processing

Tree-Based Pursuit: Algorithm and Properties

Philippe Jost, Pierre Vandergheynst and Pascal Frossard
Signal Processing Institute
Swiss Federal Institute of Technology
Lausanne, Switzerland
{philippe.jost,pierre.vandergheynst,pascal.frossard}@epfl.ch

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, a NP-Hard problem. Even sub-optimal 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 sub-dictionary 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 at hand.

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 is 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 way of achieving sparsity that has gained a lot of interest recently is to turn to very redundant systems. It often allows for short-length representation of signals, since the probability of finding a good 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. A review of the most usual sparseness measures can be found in [1].

Finding the best linear expansion using a redundant dictionary of functions is however, in the general case, a daunting task. It has been shown that it is in fact a NP-Hard problem [2]. Despite the difficulty to find the *best*, sparsest solution, it is possible to find sufficiently *good* representations that are nearly optimal. Sub-optimal heuristics have been developed that recover the main components of a function in a redundant dictionary. Among the most popular algorithms that finds good suboptimal solutions to the sparsest signal representation problem, we can cite Matching Pursuit [3] and Basis Pursuit [4]: 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 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 [5], [6], [7], 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 [6]. Multiscale [8] or subband dictionaries [9] can be used to decrease 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, [10] 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 [11], [12]. 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 arbitrary highly redundant dictionary. It naturally leads to the notion of data structuring, that 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

This work has been partly supported by Swiss National Science Foundation trough NCCR IM2 and grants PP002-68737.

is huge and hides the underlying structures, see [13] 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 [14]. Tree search has been proposed in [15] 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 trade-off 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 then proposed in Section IV that 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. Section VI illustrates the performance of Tree-Based Pursuit in terms of approximation and complexity, compared to Matching Pursuit. Section VII finally concludes the paper.

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 approximation. This is partly motivated by the potential of the related techniques for typical tasks in signal processing such as analysis, dimensionality reduction, de-noising or compression. This section provides an overview of the main recent results on sparse approximation, and practical algorithms like Matching Pursuit.

Given a d dimensional signal s in a real vector space, the central problem faced in this paper is the following: compute a good approximation $\tilde{s}_N$ as a linear superposition of N basic elements picked up in a huge collection of signals or probes $\mathcal{D}$, usually referred to as a dictionary. We will sometimes deal with $\mathcal{D}$ as a big matrix of size $d \times |D|$, where $|D|$ is the cardinality of $\mathcal{D}$. In this case, the columns of this matrix are the basic signals mentioned above, which are often called atoms. The dictionary is said to be redundant when $|D| \gg d$. The approximant $\tilde{s}_N$ is sparse when $N \ll d$ and, in this paper, the error is usually measured in the mean-square sense, i.e.,

$$\tilde{s}_N = \sum_{k=0}^{N-1} c_k g_k, \quad g_k \in \mathcal{D}\,, \quad \|s - \tilde{s}_N\|_2 \leq \epsilon\,. \tag{1}$$

There is no particular requirements 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 eq. (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 :

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

where $\|c\|_0$ counts the number of nonzero entries in the sequence $\{c_k\}$. Usually, finding the solution of P_0 would be a hopeless combinatorial problem. Recently though there has been tremendous advances studying particular instances of the following relaxed version [16] of P_0 :

$$P_1 : \text{minimize } \|c\|_1 \text{ subject to } \|s - \sum_{k=0}^{K-1} c_k g_{\gamma_k}\|_2 < \epsilon\ .$$

For the particular case where $\epsilon = 0$, P_1 can be solved by a simple convex mathematical program known as Basis Pursuit [17]. The interested reader may want to check [18], [19], [20] for full account on the exact sparse representation case (i.e. $\epsilon = 0$). The technical battle for fully understanding P_0, P_1 and their connections still rages on. In the more general case, it has recently been shown that a quadratic programming algorithm known as Basis Pursuit Denoising is able to recover a solution very close to the optimal solution of P_1 under some technical hypotheses on the dictionary [21], [16]. More surprisingly, even simple greedy strategies such as Matching Pursuit and Orthogonal Matching Pursuit are able to recover very good approximants [16]. On the downside, these results hold only 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 [19], [20], [22]. We will also come back to incoherent dictionaries in the course of this paper.

So far the results obtained are not constructive. They essentially 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. In practice, given a solution computed by any algorithm, one could use the test described in [23] to check if the solution is indeed the sparsest. Incoherence is a very strict constraint imposed upon a dictionary. But this has to be understood as a mathematical artifice to tackle a difficult problem and redundant dictionaries work very well in practice. One of the most widely used algorithm for computing sparse approximations with redundant dictionaries is the greedy algorithm known as matching pursuit, which we review in the next section.

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* [3]. Assuming that all atoms in $\mathcal{D}$ have norm one, we initialize the algorithm by setting $R_0 = s$ and we first decompose the signal as

$$R_0 = \langle g_{\gamma_0}, R_0 \rangle g_{\gamma_0} + R_1 \,,$$

where g_{γ_0} is chosen so as to maximize the correlation with R_0 :

$$g_{\gamma_0} = \arg\max_{\mathcal{D}} |\langle g_{\gamma_0}, R_0 \rangle| \,.$$

We then iterate the procedure on the residual R_1 and, after M steps, build the following approximation :

$$s = \sum_{m=0}^{M-1} \langle g_{\gamma_m}, R_m \rangle g_{\gamma_m} + R_M \,,$$

where the norm of the residual (approximation error) satisifies

$$\|R_M\|^2 = \|s\|^2 - \sum_{m=0}^{M-1} |\langle g_{\gamma_m}, R_m \rangle|^2 \,.$$

The performance of greedy algorithms like 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. Other more sophisticated metrics have been proposed to provide more precise description of dictionaries and will be described later on in this paper. Similarly, the structural redundancy [24] of a dictionary provides important information about the structure of a redundant dictionary. Matching Pursuit converges exponentially fast in finite dimension [3], [2]. There exist two constants $\alpha > 0$ and $\beta > 0$ 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

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

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 eq. (1) using a redundant dictionary is of combinatorial complexity. The greedy heuristic finds a usually satisfactory solution 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 [22]. However, polynomial time still does not mean fast! Typical implementations of Matching Pursuit suffer from a high computational complexity when compared to most orthogonal transforms. In the remainder of this paper, we therefore propose to 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, that 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 that 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 labelled by the index set Γ. A sub-dictionary $\mathcal{D}_\Lambda$ is such that $\mathcal{D}_\Lambda = \{g_i\}_{i\in\Lambda}$ where $\Lambda \subset \Gamma$ and $\Lambda \neq \emptyset$. A collection of sub-dictionaries $\{\mathcal{D}_{\Lambda_i}\}$ forms a partition of the dictionary $\mathcal{D}$ if $\bigcup_i \Lambda_i = \Gamma$ and $\forall i \neq j,\ \Lambda_i \bigcap \Lambda_j = \emptyset$. 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 [22]). This is not the case for highly correlated redundant dictionaries, though. This 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 sub-dictionary $\mathcal{D}_{\Lambda_i}$ 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 sub-dictionary by :

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

A sub-dictionary will be referred to as *reducible* when $\lambda_\Lambda > 0$ and sufficiently big. In order to quantify the adequation of the molecule in representing the atoms in the sub-dictionary $\{\mathcal{D}_{\Lambda_i}\}$, a distance measure has to be defined. Let $d(g_i, g_j)$ be a measure of the distance between two atoms g_i and g_j. In this paper, we chose to use the following distance measure, derived from the simple cosine function :

$$d(g_i, g_j) = 1 - \frac{| < g_i, g_j > |^2}{\|g_i\|_2 \|g_j\|_2}. \tag{6}$$

Without loss of generality, the distance between two atoms therefore takes values between 0 and 1, where two atoms are strongly correlated if their distance is close to 0. Since moreover the atoms we consider here have unit energy, the distance $d(g_i, g_j)$ is equal to: $d(g_i, g_j) = 1 - | < g_i, g_j > |^2$. 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 $f = \sum_{i\in\Gamma} a_i g_i + \epsilon$ could be reversed. The distance measure given in eq. (6) is independent of the direction of g_i.

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 sub-dictionary $\mathcal{D}_\Lambda$, 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 eq. (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 sub-dictionary $\mathcal{D}_\Lambda$. 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 theorem 1.3.20 of [25]). This last matrix is the Grammian of A_Λ. Fig. 5 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 [25]. This result was exhibited in [26] for the computation of the centroid for a modified k-means algorithm that considers two anti-correlated vectors, i.e., g and $-g$, as being part of the same cluster.

The computation of the optimal molecule relies on the distance measure at hand; in a different context, [11] studied the same problem with : $d(g_i, g_j) = 1 - | < g_i, g_j > |$ and derived an iterative method to compute the optimal molecule based on a weighted average update. Assuming the existence of a past version of the molecule m_Λ^k, the sub-dictionary $\mathcal{D}_\Lambda$ is divided into two parts, $\mathcal{D}_\Lambda^{(+)}$ and $\mathcal{D}_\Lambda^{(-)}$ according to the sign of the scalar product between the atoms and m_Λ^k. The new molecule is found using:

$$m_\Lambda^{k+1} = \frac{\sum_{i\in\mathcal{D}_\Lambda^{(+)}} w_i g_i - \sum_{i\in\mathcal{D}_\Lambda^{(-)}} w_i g_i}{\sum_{i\in\mathcal{D}_\Lambda} w_i}. \tag{11}$$

The positive weights w_i associated to each atoms are used to give more importance to some patterns. Due to the recursive computation of the molecule, this kind of approach fits well into a k-means algorithm.

B. Dictionary characterization

In the previous section, we introduced the definition of molecule in order to structure the *information* at hand in a highly redundant sub-dictionary. We will now see how a dictionary can be partitioned into disjoint sub-dictionaries 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 in an efficient search algorithm.

We previously stated that representing a sub-dictionary by a molecule makes sense only for *reducible* sub-dictionaries. By extension, a dictionary $\mathcal{D}$ is said to be *reducible* if it contains a partition $\{\mathcal{D}_{\Lambda_i}\}$, such that all its sub-dictionaries are reducible and $|\{\mathcal{D}_{\Lambda_i}\}| \ll |\mathcal{D}|$, i.e., the number of sub-dictionaries is much smaller that the number of atoms in the dictionary. A special case of *reducible* dictionaries is represented by the *block incoherent* dictionaries [27]. These dictionaries are such that it is possible to find a partition having a small block-coherence μ_B defined by :

$$\mu_B = \max_{i \neq j} \max_{\substack{k \in \Lambda_i \\ l \in \Lambda_j}} \mid \langle g_k, g_l \rangle \mid \ . \tag{12}$$

If $\mathcal{D}$ is *reducible*, then the coherence μ of $\mathcal{D}$ is big; the reverse is however not necessarily true. A dictionary $\mathcal{D}$ can have a big coherence μ without being *reducible*, due to the fact that the coherence given in eq. (2) only reflects an extreme property of the dictionary. Similarly, the quantity β defined in eq. (4), or the structural redundancy [24], also reports an extreme property of the dictionary. For *block incoherent* dictionaries, the structural redundancy is low and provides some *inter* sub-dictionaries redundancy measure. It is however closely related to the block-coherence μ_B given in eq. (12).

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 :

$$\mu_1(m) = \max_{|\Lambda|=m} \max_{i \notin \Lambda} \sum_{j \in \Lambda} \mid \langle g_i, g_j \rangle \mid . \tag{13}$$

A dictionary whose cumulative coherence grows slowly is said to be *quasi-incoherent* [22]. If it grows fast, it is at least possible to have one highly correlated sub-dictionary. 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 sub-dictionary, we then have :

$$\mu_1(m) \leq \begin{cases} m\mu & \text{if } m < k. \\ (k-1)\mu + (m-k+1)\mu_B & \text{if } m \geq k. \end{cases} \tag{14}$$

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 [16]. 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 sub-dictionary. 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.

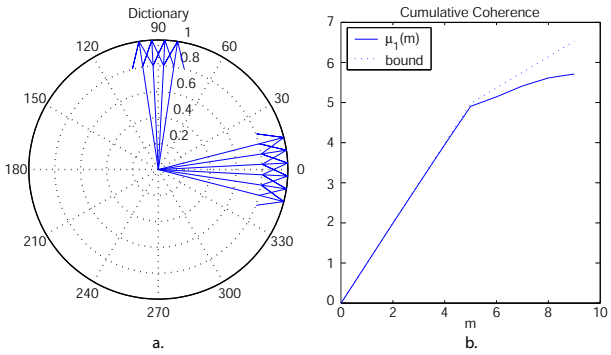

Fig. 1. Simple *block incoherent* dictionary made of two highly redundant parts (a). Evolution of its cumulative coherence and the upper bound provided by eq. (14) (b).

C. Tree-structured dictionaries

The hypothesis that the dictionary is *reducible* ensures that it is possible to partition it into *reducible* sub-dictionaries, and recursively find molecules. However, we have not yet provided a way to compute the partition of $\mathcal{D}$ in sub-dictionaries. Our ultimate goal is to have as few sub-dictionaries as possible, with atoms within each sub-dictionary that are as similar (correlated) as possible, and atoms from different sub-dictionaries 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 sub-dictionaries are seen as clusters of atoms, and the associated molecules are the centroids of each cluster. Each node t_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 eq. (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 Figure 7 for an illustration of these notations.

Our goal is to generate a tree representation of a dictionary $\mathcal{D}$ through recursive clustering, in order to eventually decrease the pursuit computational complexity. In general, two different clustering approaches can be chosen: (i) a top-down approach that tries to divide the *reducible* dictionary (or sub-dictionary) into sub-dictionaries, which 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 [26] 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. The algorithm proposed in [26] is a modified *k-means* using as distance measure $d(x, y) = \langle x, y\rangle^2$ where x and y are unit norm vectors. The correspondence between this distance and the correlation distance measure of eq. (6) is straightforward. The optimal centroid is indeed derived in the same way as the optimal molecule (see section III-A). Note however that [26] contains an additional step that explicitly identifies two anti-correlated clusters.

In this paper however, we will rather 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 top-down algorithm presented in [26] fixes a priori the number of clusters (sub-dictionaries), while the bottom-up approach presented here sets the cardinality k of each cluster. Algorithm 1 presents a sketch of the method. Initially, it creates nodes containing the atoms from a dictionary $\mathcal{D}$ and marks all these nodes as potential candidates to be grouped by adding the indexes of the corresponding nodes to a list L. The next step consists in finding a group $G \in L$ of k nodes that can be grouped. The distance measure is used to decide whether a group of nodes can be merged and a new node added to the tree. The decision algorithm considers a set Ω_G of node indexes (possibly different from G) and computes the value $d_{max} = \max_{\substack{i,j \in \Omega_G \\ i \neq j}} d(m_i, m_j)$. This value is closely related to the minimal coherence measure given in eq. (5) as $d_{max} = 1 - {\lambda_{\Omega_G}}^2$, where $\mathcal{D}_{\Omega_G}$ is a sub-dictionary made of atoms and molecules. A *reducible* sub-dictionary has been defined to have a high minimal coherence and thus, d_{max} is low. According to this definition, it makes sense to represent a sub-dictionary by a molecule if d_{max} is smaller than a fixed threshold δ. The molecules are created from the atoms or molecules listed in Ω_G. The algorithm goes on as long as it is possible to find a group of nodes fulfilling our requirements. If it is no longer possible, we create the root node of the tree; the remaining nodes in L are its children.

Algorithm 1 Tree Creation by grouping.

INPUT: A dictionary $\mathcal{D}$, the desired cardinality k of clusters.
OUTPUT: A tree $\mathcal{T}$
INITIALIZATION: Create nodes t_1 up to $t_{|\mathcal{D}|}$ containing the atoms from $\mathcal{D}$. Add all indices to a list L of free nodes.
while possible to find a group G of nodes whose index are in L that can be represented by a molecule, $card(G) = k$ **do**
  create molecule
  remove k selected nodes from list L
  create new node; its children are the k selected nodes
  add index of new node to list L
end while
create root node
children of root node are the nodes whose indexes are in L

We further define a *weak* and a *strong* decision rules, which differ in the creation of the list of nodes Ω_G associated to G. The *weak* version defines $\Omega_G = G$, the set of indexes of the nodes to group, while for the *strong* decision rule, Ω_G contains the indexes of the leaf nodes that are the descendants of the different nodes of G. In the remainder of this paper we use trees built using this bottom-up strategy, with a *weak* decision rule for grouping the atoms. Finally, 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 $k = 2$), and the tree can anyway be constructed off-line, without penalizing the pursuit algorithm. Figure 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 3 molecules, which are too incoherent to be further clustered.

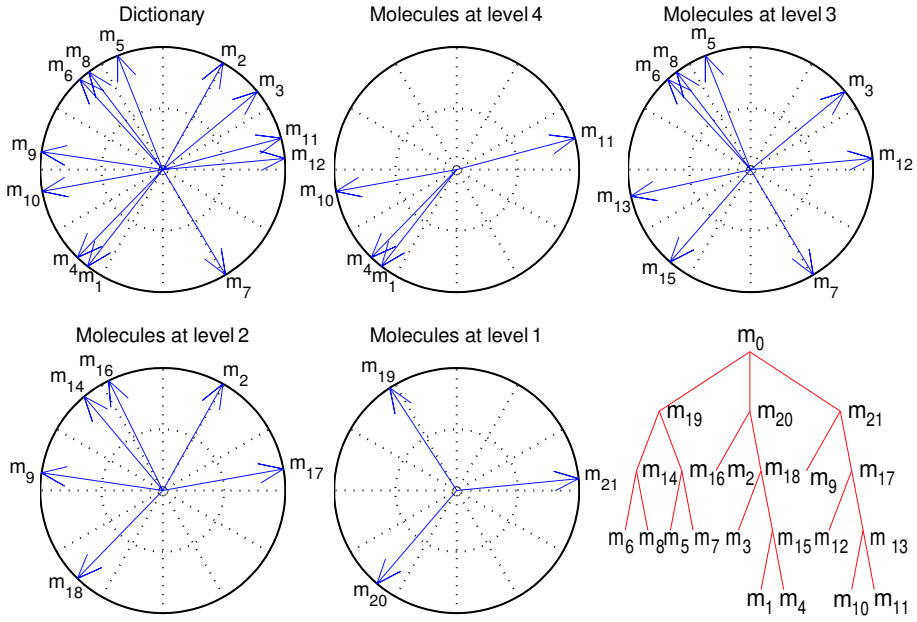

Fig. 2. Creation of a tree on top of a 2D dictionary. The upper-left part shows all atoms in the dictionary. The bottom-right part summarizes the structure of the tree. The other parts correspond to the molecules or atoms present at the different levels of the tree.

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 the hierarchical structure allows to discard an important part of the dictionary atoms 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, that are translated to different positions in the signal space, e.g., in time or space. Position parameters are conceptually part of the atom index or description. However, we chose to decouple translation from the other atom parameters, to allow for a more efficient search algorithm. Dictionary structuring does not consider atom shifts, and the tree is therefore built on atoms that are all centered on the same arbitrary reference position. Since the tree does not consider atom translations, the search algorithm itself has to deal with the position of atoms. The search algorithm has therefore to identify not only the best atom in the dictionary, but also its position in the signal space. Let $[p_{opt}, i_{opt}] = mp(f, p, \sigma, i)$ be the primitive operation that finds the atom or molecule that best approximates the signal f among the children of a node t_i of $\mathcal{T}$. The tree is shift-invariant, and the primitive *mp* searches in a window of size σ around a position p in f, and returns the index i_{opt} of the best child, and the best position p_{opt}. If the search window totally covers the function f, the primitive mp is equivalent to Matching Pursuit; in this case, we denote the search function as $[p_{opt}, i_{opt}] = mp(f, i)$.

Tree-Based Pursuit is described by Algorithm 2. At the root node, the scalar products between the residual $R^n f$ and all shifted versions of the molecules of the nodes at the first level of the tree are computed using $mp(R^n f, 0)$. This operation corresponds to an execution of Matching Pursuit using the molecules of the first tree level as dictionary. The best molecule and its associated node are found; the initial step also gives the position of the best molecule. It can also 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 search at the next node down the tree benefits from the

Algorithm 2 Tree-Based Pursuit algorithm

INPUT: A dictionary $\mathcal{D}$ and its tree representation $\mathcal{T}$, the size σ of the search window and a signal s.
OUTPUT: Atoms from $\mathcal{D}$ and projection coefficients.
INITIALIZATION: $R^0 f = f$, $n = 0$
repeat
  $[p, i] = mp(R^n f, 0)$
  while t_i is not a leaf node **do**
    $[p, i] = mp(R^n f, p, \sigma, i)$
  end while
  g_n the atom from $\mathcal{D}$ equivalent to m_i at position p.
  $a_{n+1} =< R^n f | g_n >$
  $R^{n+1} = R^n f - a_{n+1} g_n$
  $n = n + 1$
until Stop condition is met.

information about the optimal position of the molecule associated to the parent node. 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 residual function is updated and the algorithm is iterated, back to the root node, until a stopping criteria is reached. It could be a predetermined number of atoms, or a threshold on the residual energy.

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 $\lceil 1 + \log_k \frac{|\mathcal{D}|}{|c_0|} \rceil$, 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.

The proposed algorithm looks for the best child of a node, according to the adequation of the corresponding molecule with the signal. When doing so, a local search is performed at an internal node of the tree. 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. Let us first derive the complexity of both these searches. Since our atoms are centered and we have to deal with all possible translations, 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 $\mathcal{O}(|\mathcal{D}| N \log N)$ to find the best atom, where N is the size of the signal to decompose. When computed by Tree-Based Pursuit, the complexity of the search at the root node becomes $\mathcal{O}(|c_0| N \log N)$. During the traversal of the tree, only local searches are performed. It leads to a complexity of $\mathcal{O}((d-1)\sigma N)$ where d is the depth of the tree. Putting it all together, the complexity of the proposed algorithm for finding the best atom is:

$$\mathcal{O}(|c_0| N \log N + (\lceil \log_k \frac{|\mathcal{D}|}{|c_0|} \rceil) \sigma N). \tag{15}$$

The complexity of Matching Pursuit depends linearly on the size of the dictionary. A decision-tree approach to find the best atom reduces this complexity, since the divide and conquer procedure eliminates many possibilities at each level. In most cases, the second term of eq. (15) is small compared to the first one, which means that most of the complexity of Tree-Based Pursuit lies in the initial search at the root node. The complexity highly depends on the number of nodes at the first level of the tree.

The complexity of the descent through the tree depends on both the size of the search window σ, and the length of the path. The search window parameter is chosen empirically such that $\sigma \ll N$. The length of the path depends on the cardinality of the dictionary, on the number of nodes at the first level of the tree, and on the number k of children per node. This last value is also empirically chosen such that $k \ll |\mathcal{D}|$. Figure 3 shows the evolution of the complexity of the proposed algorithm, as given by eq. (15), as a function of the number of nodes c_0 at the first level under the root node. The evolution is quasi linear. It illustrates the fact that, for reasonable values of the search window σ, the descent trough the tree is negligible regarding the complexity of the initial step. The influence of the search window size is generally negligible as compared to the influence of c_0, which is usually large. The second part of the figure presents the evolution of the complexity given as a function of the size of a dictionary, for fixed number of nodes at the first level of the tree. It can be seen that the complexity of the Tree-based

Pursuit is almost unaffected by the growth of the dictionary, while the complexity of Matching Pursuit increases linearly. This confirms the weak relative importance of the second term of (15). 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.

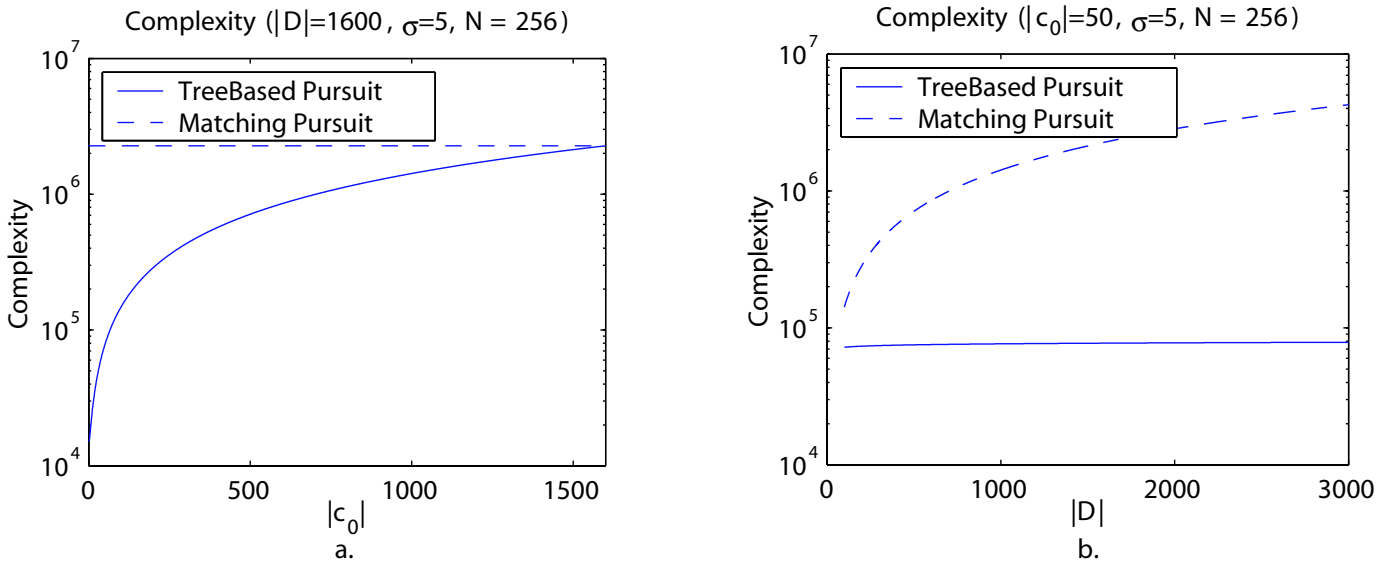

Fig. 3. Evolution of the complexity given the number of nodes at first level of the tree for a dictionary of fixed size (a) and in function of the size of the dictionary for a fixed number of nodes at the first level of the tree (b).

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 sub-dictionaries containing highly correlated atoms. As discussed in Section III-A, it makes sense in this case to define a measure of the minimal coherence of a sub-dictionary, as given in eq. (5). For an arbitrary sub-dictionary $\mathcal{D}_\Lambda$ of a dictionary $\mathcal{D}$, the minimal coherence λ_Λ is very likely to be null. This measure is strictly positive only if the most *distant* atoms of the considered sub-dictionary are correlated. As explained in Section III-A, a highly redundant sub-dictionary is the favorable case in which it is possible to represent the information at hand in the sub-dictionary by a unique element. These constraints are summarized by the following definition.

Definition 1. *A sub-dictionary $\mathcal{D}_\Lambda$ is reducible to a molecule m_Λ, which is called* representative *if*

- λ_Λ *strictly positive.*
- $\min_{k\in\Lambda} \mid \langle g_k, m_\Lambda \rangle \mid \geq \lambda_\Lambda$.
- $m_\Lambda \in span\ \{\mathcal{D}_\Lambda\}$.

In other words, the coherence between a good molecule and any atom in the sub-dictionary should be at least greater than the minimal coherence of the sub-dictionary. In section III-A, we have defined an optimality criterion for a molecule relying on the measure of a mean distance. This measure has been used for the creation of a molecule and has the advantage to define a convex set. This implies that standard optimization tools can be applied to find an optimal molecule. The adequation of a molecule regarding the sub-dictionary it represents can be defined in different ways. One possible measure consists in the minimal coherence between a sub-dictionary and its associated molecule, given by :

$$\sigma_\Lambda = \min_{i\in\Lambda} \mid \langle m_\Lambda, g_i \rangle \mid . \tag{16}$$

The definition of a representative molecule therefore implies that the minimal coherence of a molecule regarding its associated sub-dictionary is such that $\sigma_\Lambda \geq \lambda_\Lambda$. In other words, adding the molecule m_Λ to its sub-dictionary $\mathcal{D}_\Lambda$ does not change

the minimal coherence. This condition defines a subset of $\mathcal{D}$ where the molecule is allowed to exist. For example, in two dimensions, Figure 4 presents the region of admittance for a sub-dictionary of 6 atoms.

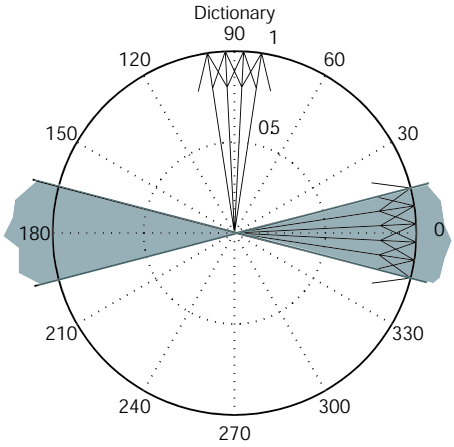

Fig. 4. A molecule must have unit norm; thus, the region of admittance is the circle in the light grey region.

B. Covering conditions

Since the search is organized along a tree structure, it has to be ensured that the re-structured dictionary is still able to cover the full space of the input signal. In particular, conditions have to be set, such that representative molecules cover the same space as the original dictionary atoms. The structural redundancy [24] can be used to define such a criteria on the dictionary construction. For example, finding a dictionary made of N vectors with good covering of the signal space can be interpreted as maximizing the structural redundancy. This quantity is however very hard to compute in practice and creating a dictionary with optimal structural redundancy is even more complex. This problem is closely related to finding an optimal covering of a projective space, i.e., a Grassmanian packing [28]. Note that Tropp has defined a measure of the covering radius of a dictionary [29], as :

$$cover(\mathcal{D}) = \max_{s \neq 0} \min_{i \in \Gamma} \sqrt{1 - (\frac{\mid \langle g_i, s \rangle \mid}{\|g_i\|_2 \|s\|_2}}). \tag{17}$$

The relation between the covering radius and the characteristic parameter β (4) of a dictionary is straightforward, i.e., the covering is minimal when β is maximal :

$$cover(\mathcal{D}) = \sqrt{1 - \beta^2}. \tag{18}$$

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 sub-dictionary. 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 sub-dictionaries* $\{\mathcal{D}_{\Lambda_i},\, i = 1, \ldots,\, K\}$ *forms a partition of* $\mathcal{D}$ *and the associated molecules are representative, then* $span\{m_{\Lambda_i,\, i=1,\ldots,\, K}\} = span\ \mathcal{D}$ *if*

$$\sigma_{\Lambda_i} > \beta + 2\sqrt{1 - \beta} - 1, \forall i. \tag{19}$$

PROOF Let $f \neq 0$ be a signal lying in the span of $\mathcal{D}$. 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., $\mid \langle f, g_0 \rangle \mid = \max_{i \in \Gamma} \mid \langle f, g_i \rangle \mid$. Suppose the atom g_0 belongs to the sub-dictionary $\mathcal{D}_{\Lambda_0}$ 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 eq. (20) as :

$$\sqrt{1 - \langle f, m_{\Lambda_0} \rangle} \leq \sqrt{1 - \mid \langle g_0, m_{\Lambda_0} \rangle \mid} + \sqrt{1 - \mid \langle f, g_0 \rangle \mid}. \tag{21}$$

We can also lower bound the last scalar product by :

$$| \langle f, g_0 \rangle | \geq \beta. \tag{22}$$

Using eqs (22) and (16), we obtain :

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

We would like to show that the projection of the signal f onto the molecule that is representative of the sub-dictionary $\mathcal{D}_{\Lambda_0}$ 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}. By extension to all the sub-dictionaries $\{\mathcal{D}_{\Lambda_i}\}$ at the first level of the tree, it guarantees that the signal f lies in the span of their representative molecules. 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 eq. (23), this holds whenever

$$\sqrt{1 - \sigma_{\Lambda_0}} + \sqrt{1 - \beta} < 1, \tag{24}$$

which leads to :

$$\sigma_{\Lambda_0} > \beta - 2\sqrt{1 - \beta} - 1. \tag{25}$$

If this condition given is verified, it ensures that $\langle f, m_{\Lambda_0} \rangle > 0$ whenever the signal $f \in \mathcal{D}$ has a component along $g_0 \in \mathcal{D}_{\Lambda_0}$. Furthermore, since the molecules are by construction in the span of their associated sub-dictionaries, the span of the molecules is within the span of the original dictionary $\mathcal{D}$:

$$span\ \{m_{\Lambda_i},\, i = 1, \ldots,\, K\} \subseteq span\ \mathcal{D}. \tag{26}$$

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

When 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 $f \in span\ \mathcal{D}$, we thus can lower bound the projection on the molecules ;

$$\max_{m_i \in \mathcal{D}_M} |\langle f, m_i \rangle| \geq \beta_M ||f||. \tag{27}$$

This also leads to :

$$\max_{m_i \in \mathcal{D}_M} |\langle f, m_i \rangle| \geq \beta_M \max_{i \in \Gamma} |\langle f, g_i \rangle|. \tag{28}$$

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

C. Recovery Condition

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 $\mathcal{D}$:

$$f = \sum_{\substack{i \\ g_i \in \Omega}} a_i g_i, \tag{29}$$

where Ω is a subset of indices.

Tropp [22] derived a minimal condition that guarantees that Orthogonal Matching Pursuit and Basis Pursuit recover Ω, where Ω is the smallest set such that eq. (29) holds. We now show that this recovery condition holds true for TBP 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 [22], that 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 sub-dictionaries). 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 se of atoms present in Φ_k. As it has been discussed, due to the bottom-up construction of the tree, the critical step consists in choosing the correct molecules at the first level of the tree. Assume once again that the sub-dictionaries $\{\mathcal{D}_{\Lambda_i}\}$ form a partition of the dictionary, and that each sub-dictionary 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 Lemma 1 holds true, then Tree-Based Pursuit chooses a good molecule at the first level of the tree, at iteration k, if*

$$\max_{m \in M_B} ||\Phi_k^+ m||_1 < \beta_M, \tag{30}$$

where Φ_k^+ is the Moore-Penrose pseudo-inverse of Φ_k. ♢

PROOF The proof of Theorem 1 is a simple extension of Tropp's Recovery Condition [22] and we provide it here for completeness. 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 can be exactly represented as $r_k = \Phi_k A_k$, where A_k 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 :

$$\frac{||M_B^* r_k||_\infty}{||M_G^* r_k||_\infty} < 1. \tag{31}$$

Developing further the left-hand side of the previous equation, using eq. (28), we can write :

$$\begin{aligned} \frac{||M_B^* r_k||_\infty}{||M_G^* r_k||_\infty} &\leq \frac{||M_B^* r_k||_\infty}{\beta_M ||\Phi_k^* r_k||_\infty} \\ &= \frac{||M_B^* (\Phi_k^+)^* \Phi_k^* r_k||_\infty}{\beta_M ||\Phi_k^* r_k||_\infty} \\ &\leq \frac{1}{\beta_M} ||M_B^* (\Phi_k^+)^*||_{\infty,\infty}. \end{aligned} \tag{32}$$

The matrix norm $\| \ \|_{\infty,\infty}$ is the maximum absolute row sum and the matrix norm $\| \ \|_{1,1}$ is the maximum absolute column sum. Thus, we can write that:

$$\begin{aligned} \frac{1}{\beta_M} ||M_B^* (\Phi_k^+)^*||_{\infty,\infty} &= \frac{1}{\beta_M} ||\Phi_k^+ (M_B)||_{1,1} \\ &= \frac{1}{\beta_M} \max_{m \in M_B} ||\Phi_k^+ m||_1 \end{aligned} \tag{33}$$

Combining eqs (31) and (33), eq. (30) finally leads to the conservative condition :

$$\frac{1}{\beta_M} \max_{m \in M_B} ||\Phi_k^+ m||_1 < 1. \tag{34}$$

■

One could further straightforwardly apply Tropp's estimate of (34) in terms of the cumulative coherence [22] of the set of molecules 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 which molecules are involved and not which 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. 1-D signals

This section now illustrates the Tree-Based Pursuit algorithm, and compares its performances to Matching Pursuit. We present results for both 1-D and bi-dimensional signals (i.e., images). Let us first consider a dictionary made of real Gabor functions, as in [3] :

$$g_{u,s,\xi,\phi}(t) = c_{u,s,\xi,\phi} g(\frac{t-u}{s}) \cos(2\pi\xi(t-u) + \phi), \tag{35}$$

with

$$g(t) = \frac{1}{\sqrt{s}} e^{-\pi t^2}. \tag{36}$$

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. Figure 5 presents 3 atoms of such a dictionary, and the representative molecule, which is the eigenvector associated to the biggest eigenvalue of $A_\Lambda A_\Lambda^*$, as discussed in Section III-A. Figure 6 presents the time-frequency representations of the atoms and the molecule of Figure 5. We can observe that the molecule indeed provides global information about all the atoms, and nicely summarizes the characteristics of the sub-dictionary.

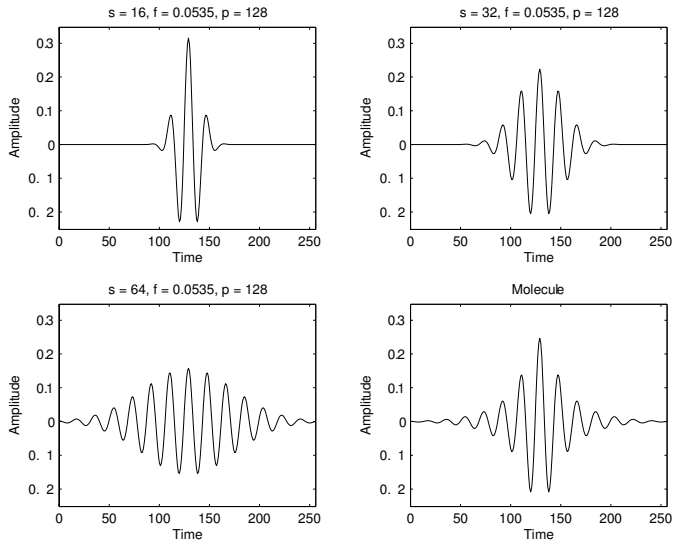

Fig. 5. Representing a group of atoms by a molecule. From top-left to bottom-left: Real Gabor atoms with same frequency f and position p but with different scales s. Bottom-right: the molecule is the eigenvector associated to the biggest eigenvalue of $A_\Lambda A_\Lambda^*$.

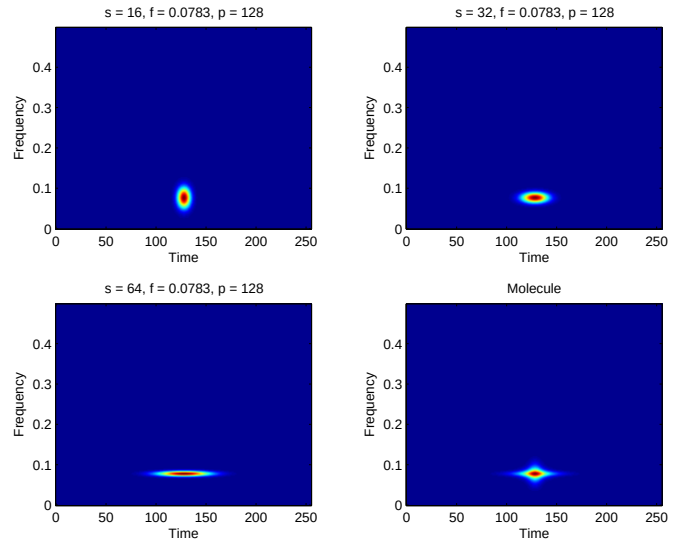

Fig. 6. Time-Frequency plane of the atoms and the molecule presented in Figure 5.

In our experiments, we used a dictionary built on real Gabor atoms with size 256, where the phase ϕ is set to zero in eq. (35). 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, without taking into account all possible shifts, which are not considered in the tree construction. The translation parameters are however computed by the search algorithm. Figure 7 shows a part of an example tree 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

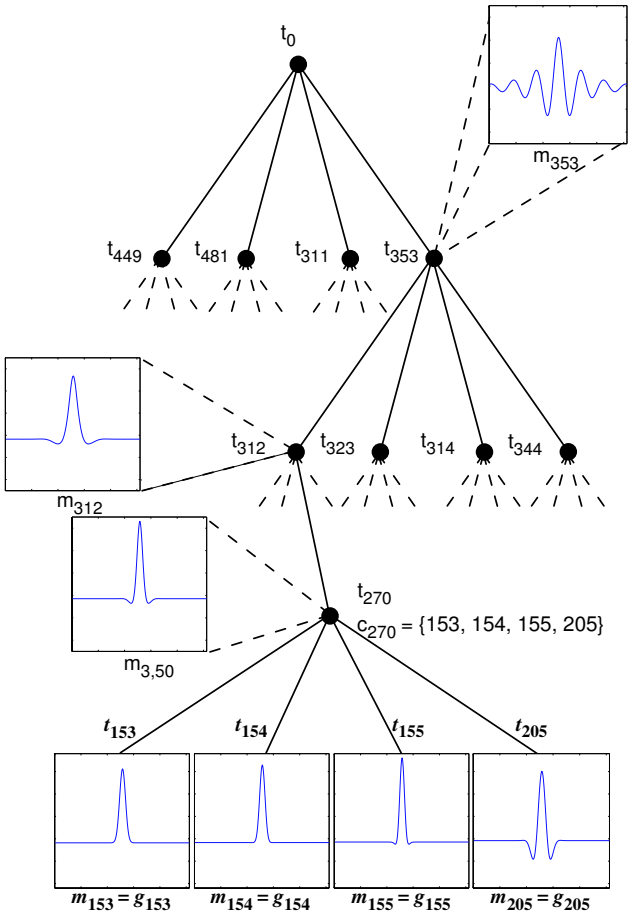

Fig. 7. Tree structure on top of a multiscale Gabor dictionary.

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 case, 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 expected depth would be 3, 5 and 8 in the order of increasing values for δ. Figure 8 presents the molecules at the first level of the tree created with a $\delta = 0.99$, while Figure 9 illustrates the corresponding time-frequency planes.

Once the dictionary has been structured in a tree representation, the search algorithm can still adapt different settings, to trade off computational complexity and approximation rate. In particular, the size σ of the local search window plays an important role in the performance of the Tree-based Pursuit algorithm. We compare results for search windows of size $\sigma = 5$, and $\sigma = 11$. Figure 10 represents distribution of the shift values, computed by the search algorithm descending along the tree, relative to the position determined in the upper-level molecules. Part (a) of Figure 10 presents the shift values, independently from the depth at which the Tree-Based Pursuit is making a position refinement. In most cases, the displacement is very small: almost 90% of them are catched by a search window of size $\sigma = 5$. Part (b-d) of Figure 10 presents the shift values at the first level of the tree (i.e., right after the initial stage has determined the correct subtree and position). In a majority of the cases, there is only a shift of 1. It is due to the fact that the Fourier transforms of the molecules at the first level are centered by a multiplication with the Fourier transform of a unit impulse located in the center of the representative molecule. As the size of the signal is even, there is a shift of 1. Due to the successive transforms, numerical imprecisions may occur, and a local search has to be performed. The shift is corrected by this refinement step. Parts (c) and (d) of Figure 10 show a similar behavior of the local search at the next levels of the tree. In general, trees with a large value of the threshold δ have correction shift values that are more uniformly distributed over the whole search window. Finally, Figure 10 illustrates the fact that the energy localization is quite efficient and that the molecules at the first level of the tree represent well the features contained

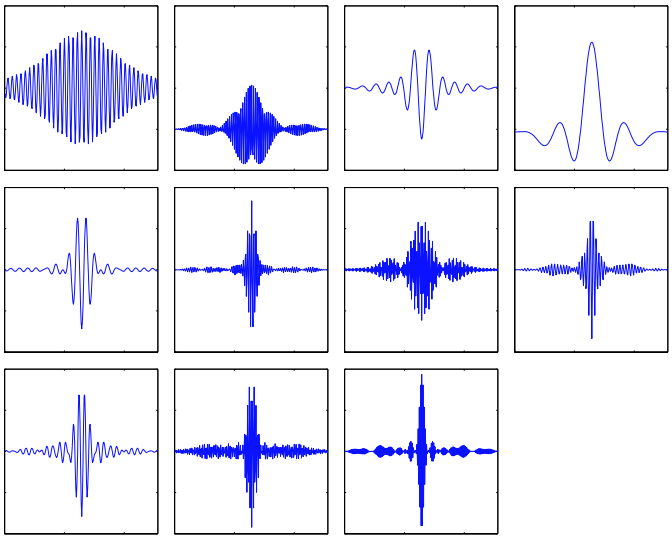

Fig. 8. Molecules associated to the nodes located at the first level of the tree created with a grouping threshold $\delta = 0.99$.

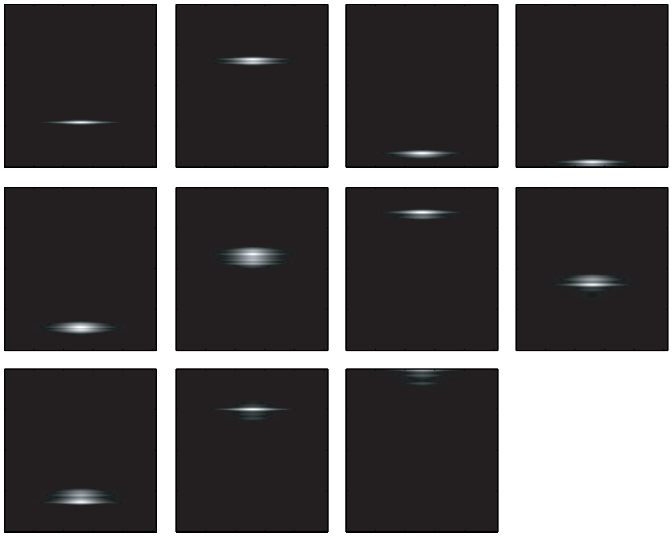

Fig. 9. Time-frequency distributions corresponding to the molecules exhibited by figure 8.

in their sub-dictionaries.

We can now compare the approximation performance, and the computational complexity of Tree-based Pursuit, as opposed to Matching Pursuit. Part (a) of Figure 11 compares the mean square 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. This fact is particularly well illustrated in Figure 11 (b), where the approximation rate of Tree-based Pursuit with $\delta = 0.18$ is close to Matching Pursuit. On the other hand, the computation time obviously also 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 Figure 12. Indeed, if we compute a linear approximation of the Tree-based Pursuit computation time curve as a function of $|c_0|$, in a mean square 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, the cost of the traversal of the tree can be considered as negligible regarding the initial step.

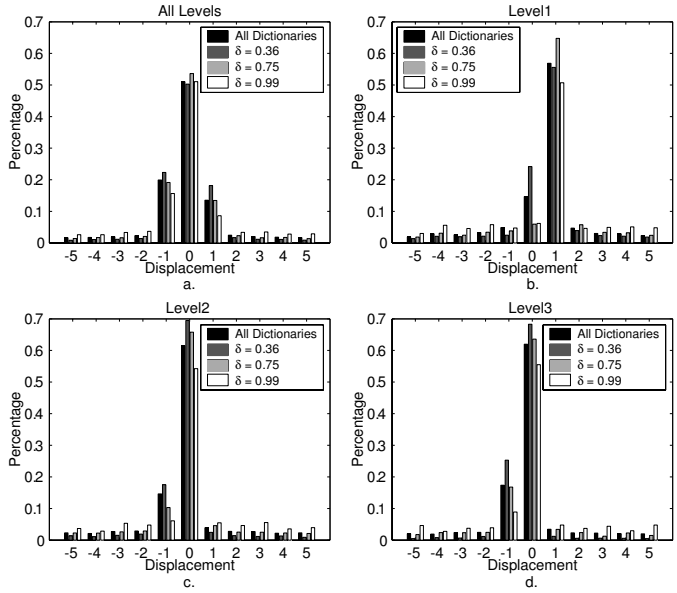

Fig. 10. Displacement of the optimal position during the execution of Tree-Based Pursuit, independently of the level (a) and for first (b), second (b) and third level (c) of the tree.

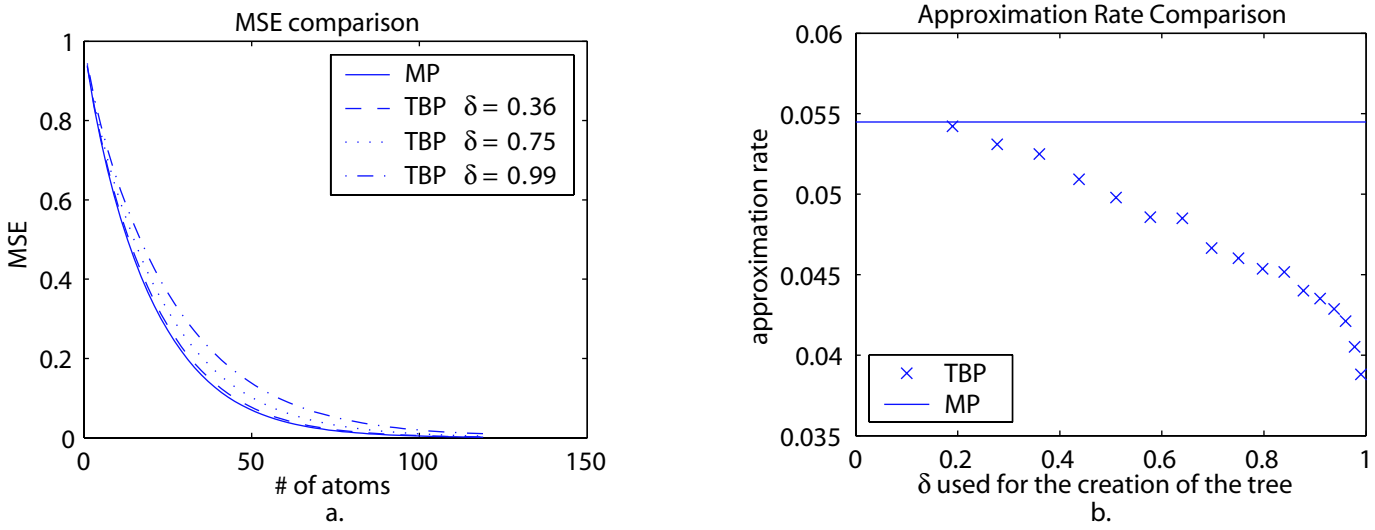

Fig. 11. Comparison of the error produced by the proposed algorithm when using different bounds for the grouping (a). The grouping parameter δ influences the approximation rate (b).

B. Extension to multi-dimensional signals

This section 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 1-D signals. We use a dictionary that is built on gaussian generating functions that are scaled, rotated and translated [30]. The first generating function is a Gaussian as given by eq. (37), that suits well the task of capturing the low-frequency parts of natural images. The second generating function, given in eq. (38) is made of a Gaussian in one direction and its second derivative in the other direction. It has a good ability to capture edges in images and is spatially and frequencially well located.

$$g_1(x,y) = \frac{1}{\sqrt{\pi}} \exp -(x^2+y^2). \tag{37}$$

$$g_2(x,y) = \frac{2}{\sqrt{3\pi}}(4x^2-2) \exp -(x^2+y^2). \tag{38}$$

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 $\frac{\pi}{18}$. The scaling parameters are uniformly

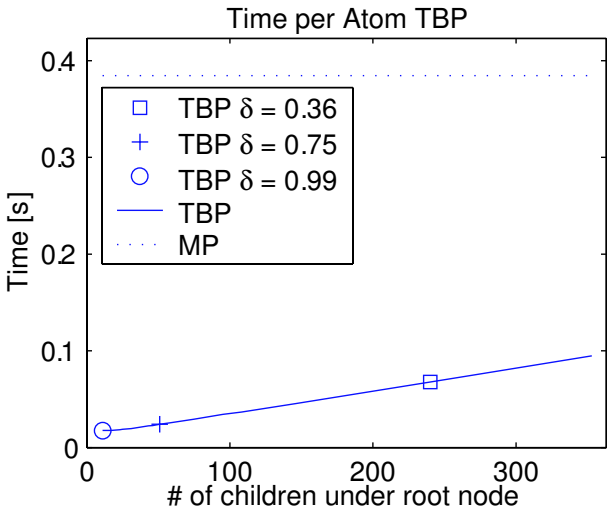

Fig. 12. Comparison of the complexity between Matching Pursuit and Tree-Based Pursuit, when using different values for δ during the creation of the tree structure.

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 $\frac{1}{32}$ to $\frac{1}{4}$ of the size of the image on a logarithmic scale with a resolution of one third of octave. Due to isotropy, rotation is obviously useless for this kind of atoms.

The tree structured dictionaries have been generated here using a top-down approach. The trees have been constructed using a *k-means* algorithm with different values for the number of children per node. Figure 13 represents a bottom part of the tree structure, built on the 2-dimensional gaussian dictionary, where the bottom components represent dictionary atoms. Figure 14 show the molecules at the first level of the tree.

Fig. 13. Bottom part of the tree-structured dictionary, built on gaussian 2-dimensional atoms.

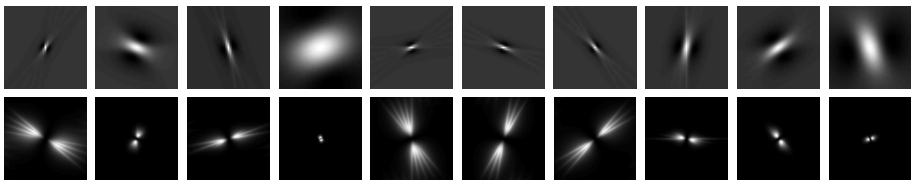

Fig. 14. Centroids at the first level of the tree built on 2-dimensional gaussian atoms. The first row presents the atoms in the spatial domain, the second one shows their frequency representation.

Figures 15 presents a comparison of the computational complexity of Tree-based Pursuit and Matching Pursuit, for different dictionary and image sizes (the Lena image has been used in these experiments). It can be seen that the number of children per node in the clustering algorithm clearly influences the computational complexity. However, approximation quality also depends on the number of children per node, as shown in Figure 16, which presents the quality of the approximation for 500 atoms. Interestingly, the computational time also varies linearly with the number of children per node, similarly to the behavior observed in the 1d signal case.

VII. Conclusions

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 set of functions. The molecules

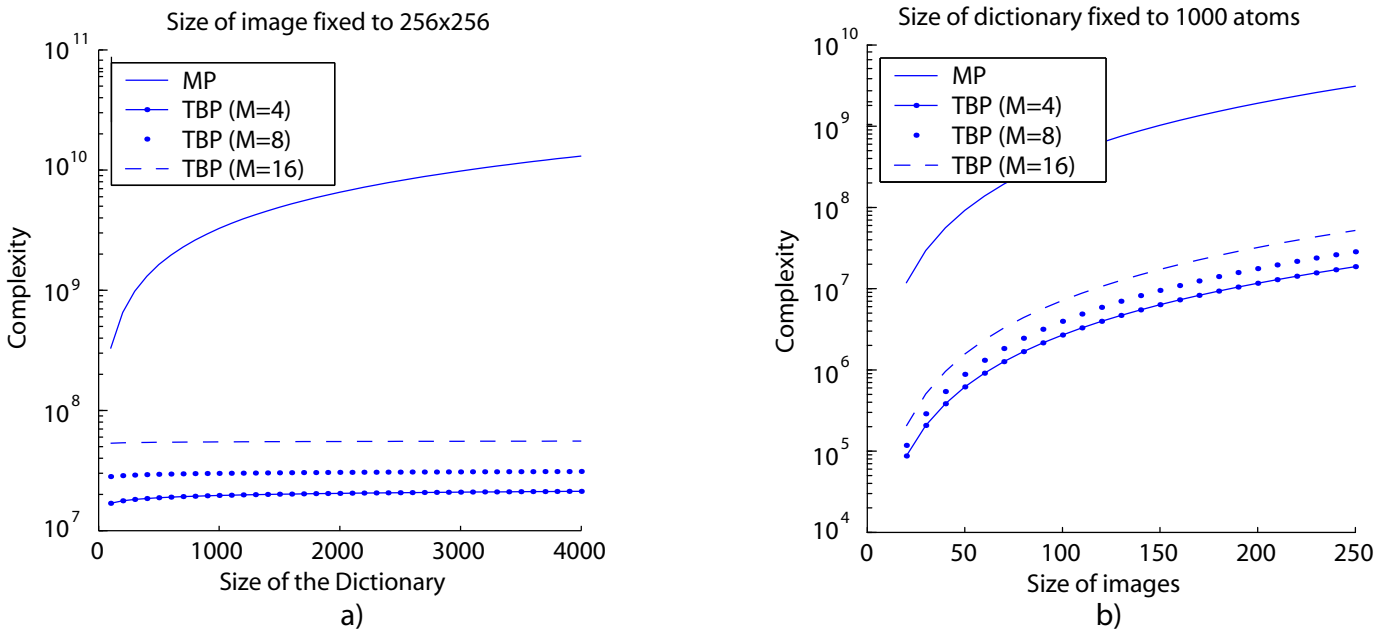

Fig. 15. Complexity comparison between Tree-based Pursuit, and Matching Pursuit, versus the size of the dictionary, and the size of the image to represent. The parameter M is the number of children per node used for the creation of different trees.

represent a sub-dictionary 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 dictionary, 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] J. Karvanen and A. Cichocki, "Measuring sparseness of noisy signals," in *Proceedings of 4th International Symposium on Independent Component Analysis and Blind Signal Separation (ICA2003)*, Apr. 2003, pp. 125–130.

[2] G. Davis, S. Mallat, and M. Avellaneda, "Adaptive greedy approximations," *Journal of Constructive Approximation*, vol. 13, pp. 57–98, 1997.

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

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

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

[6] R. Neff and A. Zakhor, "Matching pursuit video coding .i. dictionary approximation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 1, pp. 13–26, Jan 2002.

[7] K.-P. Cheung and Y.-H. Chan, "A fast two-stage algorithm for realizing matching pursuit," in *IEEE International Conference on Image Processing*, vol. 2, Oct. 2001, pp. 431 – 434.

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

[9] C. de Vleeschwouwer and B. Macq, "Subband dictionaries for low-cost matching pursuits of video residues," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 7, pp. 984–993, October 199.

[10] M. Goodwin and M. Vetterli, "Matching pursuit and atomic signal models based on recursive filter banks," *IEEE Transactions on Signal Processing*, vol. 47, no. 7, pp. 1890–1902, Jul 1999.

[11] P. Schmid-Saugeon and A. Zakhor, "Dictionary design for matching pursuit and application to motion-compensated video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 880 – 886, June 2004.

[12] Y.-T. Chou, W.-L. Hwang, and C.-L. Huang, "Gain-shape optimized dictionary for matching pursuit video coding," *Signal Processing*, vol. 83, pp. 1937–1943, September 2003.

[13] D. Jiang, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: a survey," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, no. 11, pp. 1370– 1386, Nov. 2004.

[14] S. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, no. 3, pp. 660 – 674, May-June 1991.

[15] S. Cotter and B. Rao, "Application of tree-based searches to matching pursuit," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 6, May 2001, pp. 3933 – 3936.

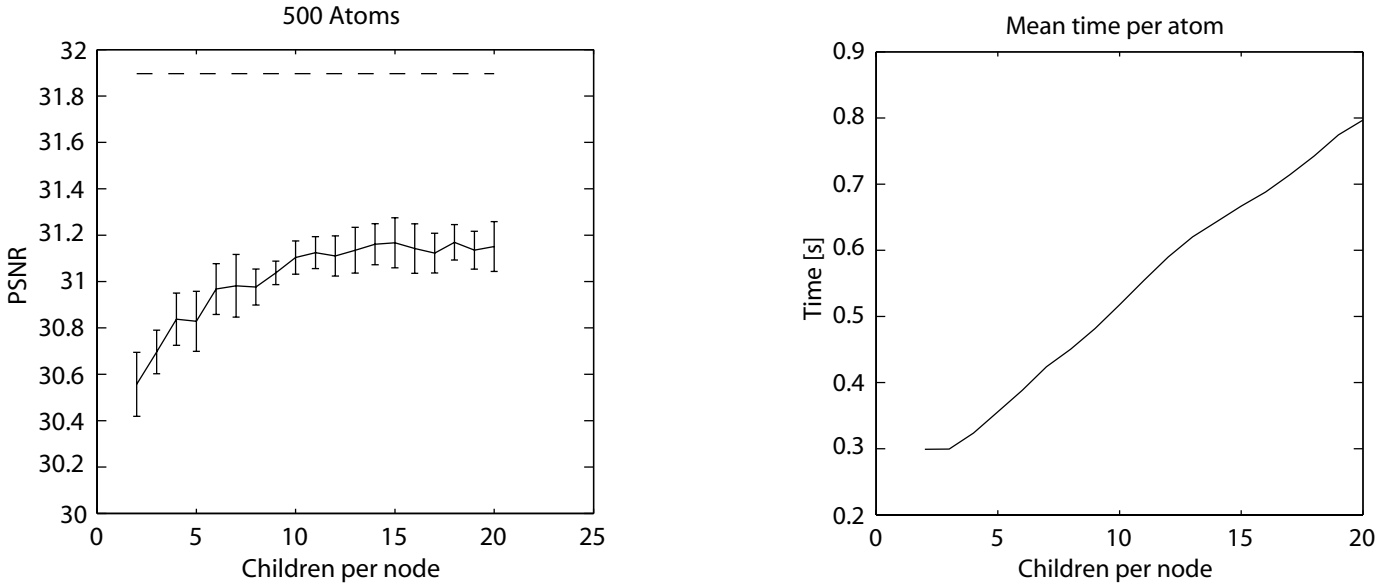

Fig. 16. Performance of Tree-based Pursuit as a function of the number of children per node used for the *k-means* algorithm. Part (a), the dashed line shows the results obtained with Matching Pursuit. Part (b) shows the mean computation time per atom for Tree-Based Pursuit; Matching Pursuit needed 47 seconds per atom.

[16] J. A. Tropp, "Topics in sparse approximation," Computational and Applied Mathematics, UT-Austin, August 2004.

[17] S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Scientific Comp.*, vol. 20, pp. 33–61, 1999.

[18] D. Donoho and X. Huo, "Uncertainty principles and ideal atomic decompositions," *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2845–2862, November 2001.

[19] M. Elad and A. Bruckstein, "A generalized uncertainty principle and sparse representations in pairs of bases," *IEEE Transactions on Information Theory*, vol. 48, no. 9, pp. 2558–2567, September 2002.

[20] R. Gribonval and M. Nielsen, "Sparse representations in unions of bases," *IEEE Trans. Inf. Th.*, vol. 49, no. 12, pp. 3320–3325, December 2003.

[21] D. Donoho, M. Elad, and V. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," 2004, working draft.

[22] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, October 2004.

[23] R. Gribonval, R. Figueras i Ventura, and P. Vandergheynst, "A simple test to check the optimality of a sparse signal approximation," *EURASIP Signal Processing Journal, Special Issue on Sparse Approximations in Signal and Image Processing [Accepted]*, 2005.

[24] P. Frossard and P. Vandergheynst, "Redundancy in non-orthogonal transforms," in *IEEE International Symposium on Information Theory*, June 2001, p. 196.

[25] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1985.

[26] 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.

[27] L. Peotta, P. Jost, P. Vandergheynst, and P. Frossard, "Sparse approximation with block incoherent dictionaries," EPFL, 1015 Ecublens, TR - ITS 2003.007, December 2003.

[28] J. Conway, R. H. Hardin, and N. J. A. Sloane, "Packing lines, planes, etc., packings in grassmannian spaces," *Experimental Mathematics*, vol. 5, pp. 139–159, 1996.

[29] J. A. Tropp, "Just relax: Convex programming methods for subset selection and sparse approximation," The University of Texas at Austin, ICES Report 04-04, February 2004.

[30] R. Figueras i Ventura, P. Vandergheynst, and P. Frossard, "Low rate and flexible image coding with redundant representations [accepted]," *IEEE Transactions on Image Processing*, February 2005.