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

Philippe Jost, Pierre Vandergheynst and Pascal Frossard

Swiss Federal Institute of Technology Lausanne (EPFL)

Signal Processing Institute Technical Report

TR-ITS-2004.13

July 20th, 2004

TREE-BASED PURSUIT

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

Building good sparse approximations of functions is one of the major themes in approximation theory. When applied to signals, images or any kind of data, it allows to deal with basic building blocks that essentially synthesize all 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 in noise or even for analyzing and compressing signals. The very strong links between approximation theory and computational harmonic analysis on one hand and data processing on the other hand, resulted in fruitful crossfertilizations over the last decade.

This paper proposes to create a tree structure from an arbitrary dictionary of functions. Due to a hierarchical classification of the original data, an important part of the redundancy is intrinsically hold by the structure that represents whole bunch of highly correlated atoms by an unique element. A pursuit algorithm taking advantage of this structure is proposed. It consists in finding the best path through the tree. It presents the important advantage of being much faster than a classical Matching Pursuit. The proposed method reduces the dimensionality of the problems without losing important information for the problem at hand; it only minimally degrades the quality of approximation. The performance of the proposed algorithm are demonstrated in the context of image representation.

I. INTRODUCTION

A signal can be represented as a superposition of waveforms. In many cases, the basis waveform are orthogonal to each other like for the Fourier analysis or for wavelets. Non-orthogonal bases became popular for their ability to lead to sparser approximation. However, finding the sparsest approximation of a signal over a redundant dictionary of basis function can be a daunting task. It has been proved in [4] that the finding an optimal function expansion over a redundant dictionary is an NP-hard problem.

Mallat and Zhang introduced the Matching Pursuit algorithm [10], which greedily takes the best atoms out of the signal. A *good* representation of the signal using an overcomplete dictionary can be found without having to pay the price of solving a problem of combinatorial complexity. Since then, other methods have been proposed. The aim of the proposed methods remain the same, lower the amount of computations that have to be made. Different approaches have been proposed. Basis Pursuit (BP) [1] finds the representation in the overcomplete dictionary that minimizes the l^1 norm of the coefficients of the approximation instead of the l^0 norm which characterizes the sparsity.

Starting from existent algorithms, it is possible to introduce slight modifications or special cases to obtain efficient search algorithms. Making MP computationally efficient is mainly done through the design of specific dictionaries. Lots of efforts have been invested in finding optimal dictionaries. A dictionary is optimal regarding the purpose it should fulfill. One can think of approximation capabilities, coding efficiency or computational complexity needed during the search phase. Finding a dictionary that is jointly good in all theses fields is a daunting task. As explained in [11], the most used approach to find efficient dictionaries in terms of complexity is to reduce the space of possible dictionaries to a complexity-restricted set. However, in terms of approximation, a good dictionary is less likely to lie in this reduced set. The typical example are dictionaries based on 2D separable functions that are not able to capture curves. Regarding this simple fact, it is proposed in [11] to create a dictionary that is efficient for approximation without complexity considerations and to slightly modify it in order to make it computationally efficient. Small changes should not affect too much its approximation capabilities.

Dealing with information naturally introduces the notion of structuring data. When the amount of data at disposal grows, the structure becomes very important. Data structure where also often responsible for the emergence of new methods. The success of wavelets decompositions for images has certainly to be put in parallel with the ease of use of the associated tree structure. It has been shown that for image compression using wavelets, a tree structure fits particularly well [2]. The present article does not use a tree for coding but to structure the dictionary of redundant basis functions. The approximation rate decreases as the size of dictionary increases. On the other hand, the complexity of an algorithm like matching pursuit highly depends on the size of the dictionary. This paper will present an algorithm whose complexity only increases logarithmically.

Section II is an overview of Redundant Image Expansion. It presents the principles and the notion of sparsity of such a decomposition. Matching Pursuit is presented as an example to solve greedily the problem. It also presents the used dictionary for the examples that will take place in the results.

Section III deals with the creation of the tree structured dictionary. It presents the properties we would like to have and proposes a recursive algorithm to solve the problem.

Section IV describes the pursuit algorithm that exploits the capabilities of the previously described tree-based structure. It also investigates the complexity of the proposed algorithm and compares them to Matching Pursuit.

Section V deals with the exact sparse problem. It exhibits a minimal condition under which it is possible to recover exactly a signal that is a weighted sum of atoms from the dictionary.

Section VI presents the results and performance of the proposed algorithms for images.

II. Redundant Image Expansions

Signal expansions using redundant dictionaries is a very active domain since the introduction of the Matching Pursuit algorithm by Mallat and Zhang in 1993 [10]. They have shown that such a greedy algorithm converges exponentially in finite dimension, and thus provides a good approximation to a difficult combinatorial problem. The excellent paper of Gribonval and Nielsen [8] presents the main results in the research field during the last decade.

In general, a redundant expansion of a function f in a Hilbert space $\mathcal{H}$ is weighted sum of basis functions, also called atoms which are functions lying also in $\mathcal{H}$. The dictionary $\mathcal{D}$ is the overcomplete set of all atoms, and can be written as $\mathcal{D} = \{g_{\vec{\gamma}}\}_{\vec{\gamma}\in\Gamma}$ with $\|g_{\vec{\gamma}}\| = 1$.

For redundant dictionaries, f has many different possible representations such that $f = \sum_{n=0}^{|\mathcal{D}|-1} c_n g_{\vec{\gamma_n}}$. The best decomposition c is chosen according to a *sparsity* measure. The aim would be to find the weights vector c with minimal l^0 norm. Thus, the problem can be expressed as:

$$\min_c ||\vec{c}||_0 s.t. f = \sum_{n=0}^{|\mathcal{D}|-1} c_n g_{\vec{\gamma_n}}. \tag{1}$$

It has been proved in [4] that the finding an optimal solution for eq. 1 over a redundant dictionary is an NP-hard problem. Different algorithms have been developed in order to find less optimal solutions at non-combinatorial costs.

Matching Pursuit (MP) is a greedy algorithm that iteratively approximates the signal. It chooses $g_{\vec{\gamma_n}}$ such that the projection coefficient with the last residual is maximal. The residual signal at step n is $\mathcal{R}^n f = \mathcal{R}^{n-1} f - < \mathcal{R}^{n-1} f | g_{\vec{\gamma_n}} > g_{\vec{\gamma_n}}$. The initial residual $\mathcal{R}^0 f = f$. Thus, the function f is decomposed as follows:

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

The matching pursuit algorithm stops when the energy of the residual is lower than an acceptable limit.

In the case of redundant expansions for images, the atoms are bi-dimensional functions. They are often chosen to match features contained in the scene as edges for example. The design of a dictionary depends on the application and on the purpose to fulfill. In this paper, we used the dictionary described in [5]. The atoms of the dictionary are built from generating functions that are scaled, rotated and translated. The first generating function is a Gaussian (eq. 3) that suits well the task of capturing the low-frequency parts of the images. The second generating function (eq. 4) 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 the images and is spatially and frequencially well located.

$$g_1(x, y) = \frac{1}{\sqrt{\pi}} \exp -(x^2 + y^2). \tag{3}$$

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

The dictionary $\mathcal{D}$ is then generated by applying transformations on these generating functions: rotations (θ), scaling ($\vec{a} = (a_1, a_2)$) and translations ($\vec{b}$). The identity k of the generating function and the parameters of the transformations define the atom $g_{\vec{\gamma}}$ where $\vec{\gamma} = \{k, \theta, \vec{a}, \vec{b}\}$. The dictionary is the overcomplete set of unit energy atoms $\mathcal{D} = \{g_{\vec{\gamma}}\}_{\vec{\gamma}\in\Gamma}$ with $\|g_{\vec{\gamma}}\| = 1$.

For the implementation and the experiments, the parameters had to take discrete values. For the atoms using g_2 as generating function, the translation parameters 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 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, rotations are obviously useless in this kind of atoms.

III. Tree Structure

For algorithms like matching pursuit, overcompletness of a dictionary is intrinsically holding the fact that redundant computations are made. The aim of structuring the data is to capture the redundancy in fewer elements without loosing the approximation capabilities of highly redundant dictionaries.

The original dictionary $\mathcal{D}$ is recursively clustered and stored as a tree where a node holds a subset of $\mathcal{D}$. Let $N_{l,n}$ be a node of the tree at level l and position n and $L_{l,n}$, the list indexes of the atoms from $\mathcal{D}$ contained in the subtree spanned by $N_{l,n}$. The matrix $A_{l,n}$ holds, in its columns, the atoms listed in $L_{l,n}$. The node $N_{l,n}$ is a leaf node if $L_{l,n}$ contains only one element. The atoms contained in $\mathcal{D}$ are stored in the leaves of the tree. Each non leaf node has an associated atom $c_{l,n}$, called centroid, which represents the atoms contained in the leaves of its subtree. Figure 1 show a part of a tree and illustrates the notation that we use.

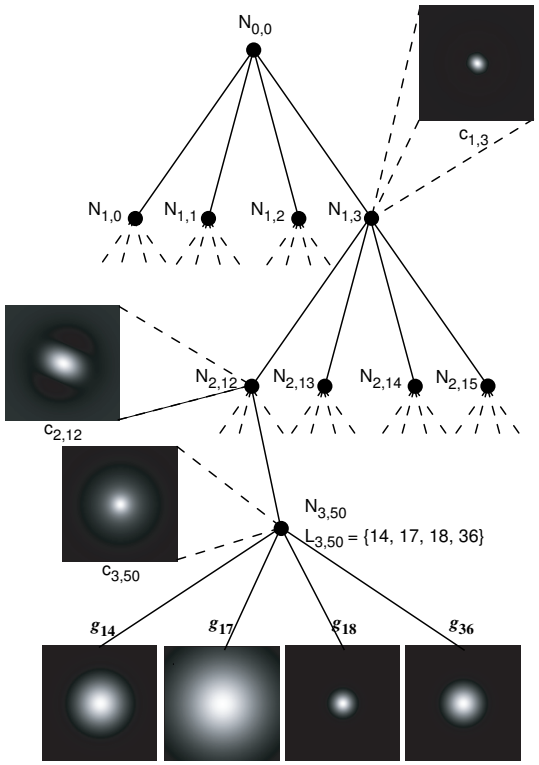

Fig. 1. Illustration of the notations used to describe the tree. Example of classification it is possible to obtain for two dimensional Gaussian atoms.

The aim of the clustering is to assign to a node of the tree a subset of atoms belonging to $\mathcal{D}$ as correlated as possible to each other. On the other hand, their corresponding centroid should be as incoherent as possible with the rest of $\mathcal{D}$ and sibling centroids. The coherence of a tree structured dictionary can be expressed in an analog manner as for a *flat* dictionary. It is the maximal absolute value of the scalar product between two centroid associated to nodes having the same parent. A tree structured dictionary is incoherent if the coherence measure 5 is small. If this value is equal is null, the centroid of sibling nodes are orthogonal.

$$\mu = \max_{l>1} \max_{k\in[0,(l-1)M]} \max_{i,j\in[kM,(k+1)M-1],i\neq j} | < c_{l,i}, c_{l,j} > |. \tag{5}$$

The function $d(x, y)$ is a measure of the distance between two vectors x and y. It takes values between 0 and 1. Two vectors are strongly correlated if their distance is near 0. A simple measure $d(x, y) = 1 - \frac{|<x,y>|}{||x||_2||y||_2}$ fits these requirements. Another measure of the distance between two atoms can be found in [13]:

$$d(x, y) = \left[1 - \left(\frac{| < x, y > |}{||x||_2||y||_2}\right)^2\right]^{1/2}. \tag{6}$$

A centroid $c_{l,n}$ represents the atoms g_i where $i \in L_{l,n}$. It has unit energy and belongs to the column span of $A_{l,n}$ noted as $\mathcal{R}(A_{l,n})$. It can be written as a linear combination of the atoms listed in $L_{l,n}$:

$$c_{l,n} = A_{l,n}C_{l,n}. \tag{7}$$

where $C_{l,n}$ is a vector containing the weights.

A centroid $c_{l,n}$ has to be such that:

$$\min_{i \in L_{l,n}} d(g_i, c_{l,n}) \geq \max_{j \in \mathcal{D} \setminus L_{l,n}} d(g_j, c_{l,n}). \tag{8}$$

In the general case, an unique element is not able to cover the whole span of the atoms contained in $L_{l,n}$ as illustrated by eq. 9.

$$\min_{x \in \mathcal{R}(A_{l,n})} c_{l,n}^* x = 0. \tag{9}$$

The quality $D_{l,n}$ of a centroid can be computed; it measures the adequation between the centroid and the elements it has to represent. A natural measure for this is the mean distance from the centroid to all the atoms.

$$D_{l,n} = \frac{1}{|L_{l,n}|} \sum_{i \in L_{l,n}} d(g_i, c_{l,n}). \tag{10}$$

The tree can be built using a top down approach. The root node $N_{0,0}$ is responsible for the whole dictionary and does not have any associated centroid. The list $L_{0,0}$ contains all the indices from $\mathcal{D}$. The creation of the tree begins by clustering the root node. It leads to the creation of M new children. The clustering algorithm is then called again on the children if the number of atoms they are responsible for is bigger than M. Starting from equation 10, it is possible to define a measure that quantifies the quality of a clustering as the mean of the local distances. However, this measure does not take into account the existing relation between the different centroids. The aim is to separate the different subspaces. Thus, let us define the quality of a clustering as the sum of $D^I(l,n)$ a measure of the intra-class coherence and $D^O(l,n)$ a measure of the inter-class coherence. Both measures take values in $[0,1]$; values near 0 mean that correlation is strong. For a given set $L_{l,n}$ of atoms, the quality $Q_{L_{l,n}}$ of a clustering is:

$$Q_{L_{l,n},\lambda} = D^I(l,n) + \lambda D^O(l,n). \tag{11}$$

A good clustering, is such that for a given positive λ, equation 11 is minimal. Assume the existence of a primitive fitting these requirements leading to the recursive part of the tree creation summarized by algorithm 1. The creation of the tree is done by using this procedure on the root node with $L_{0,0}$ the set of all indices of the atoms in $\mathcal{D}$.

From an analytical point of view, the *clustering* problem is related to packing lines into a given subspace of possible high dimension. This problem is widely treated in [3].

Algorithm 1 Clustering of Node $N_{l,n}$

Use clustering algorithm to get m clusters i.e. create $L_{l+1,nm+i}$ where $i \in [0, m[$
Create the centroids minimizing equation 11 for a given λ
for $i = 0$ to $m - 1$ **do**
  if $|L_{l+1,nm+i}| > m$ **then**
    Clustering of Node $N_{p(i)}$
  end if
end for

Figure 2 is a small part of a tree where the number of children per node has been fixed to 4. The correlation between siblings is obvious in this example. This figure also illustrates the representing capabilities of a parent node regarding its children.

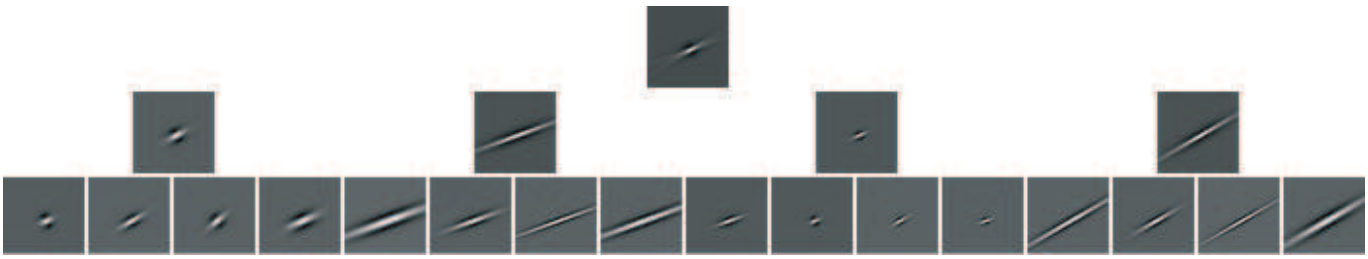

Fig. 2. Part of a tree

Depending on the technical choices made, the generation of the tree can be computationally quiet complex. As it depends just on the dictionary, it can be done once for all and stored for future reuses.

Algorithm 2 Tree-Based Pursuit algorithm

$R^0 f = f$, $n = 0$
repeat
  l=0, t=0 so that $N_{l,t}$ is set to root node
  repeat
    $I = \text{argmax}_i \mid < R^n f | c_{l+1,t*M+i} > \mid$, $i \in [0, M[$
    $l = l + 1$, $t = tM + I$
  until $N_{l,t}$ is a leaf
  $a_{n+1} =< R^n f | c_{l,t} >$
  $R^{n+1} = R^n f - a_{n+1} c_{l,t}$
  $n = n + 1$
until Enough atoms where found

IV. Tree-Based Pursuit Algorithm

Based on the tree representation of the dictionary, it is possible to design a greedy algorithm that finds the best path through the tree down to the best leaf nodes (i.e., the atoms from $\mathcal{D}$). Starting from the root node, all the scalar products between the residual and the centroids of the children are computed. The best associated node is selected and the search goes on until a leaf node is reached. This provides a fast alternative to the original Matching Pursuit method. The proposed procedure is described in Algorithm 2.

Each internal node store a representative atom for its spanned subtree. The aim of the proposed algorithm is to decide which subtree is most likely to hold atoms taking away the biggest part of the energy of the signal to approximate.

The complexity of the proposed algorithm is much lower than Matching Pursuit. For the two dimensional case, the algorithm is slightly modified. The centroid are constructed by making use of centered atoms. At the root node, all the possible scalar products between the centroids and the residual are computed using a fast Fourier transform. This gives us two precious piece of information: the best subtree and the position of the maximum. During the rest of the traversal through the tree, the possible scalar products are computed in a search window ($d_x \times d_y$) surrounding the best position found at the upper level. The step at the root node can be seen as an energy localization phase. This step is efficient due to the fact that our algorithm finds the maximal energy of the image regarding the kind of features the dictionary is able to represent. Energy localization is also used in some fast implementations of Matching Pursuit.

For the complexity analysis of our algorithm, we assume that the tree is balanced and that its mean depth is $\log_M |\mathcal{D}|$ where M is the number of children per node. The image we are looking for is of size $N = S_x \times S_y$. The FFT computes the circular scalar product; thus, the images have to be padded with zeros. Let $N^{'}$ be the amount of pixels of the padded images. Computing the inverse Fourier transform is of $\mathcal{O}(N' log N')$. Thus, our reference Matching pursuit algorithm has the following complexity for finding the best atom in the residual.

$$\mathcal{O}(|\mathcal{D}| N' \log N'). \tag{12}$$

In order to do the same task, the proposed pursuit algorithm has following complexity.

$$\mathcal{O}(M N' log N' + \log_M |\mathcal{D}| N W). \tag{13}$$

Figure 3 illustrates the fact that the proposed algorithm is less sensitive to the growth of the size of the dictionary than matching pursuit is. It is a valuable piece of information as it is known that potentially the approximation rate increases as the size of the dictionary increases.

V. Exact Recovery Condition

The proposed algorithm is designed to be used for approximation. However, it is of big interest to see under which conditions on the signal it can solve the exact sparse problem described in [12]. Let the signal s be a linear combination of atoms from the dictionary. The matrix A contains the set of atoms participating in s and b is a vector containing the weights. Let b_{min} denote the smallest element of b taken in absolute value. The signal can be written as:

$$s = Ab. \tag{14}$$

The atoms in A are linearly independent otherwise, there would exist a sparser representation of s. The exact sparse problem consists in finding a minimal condition under which the proposed algorithm recovers exactly the atoms of A and the corresponding weights.

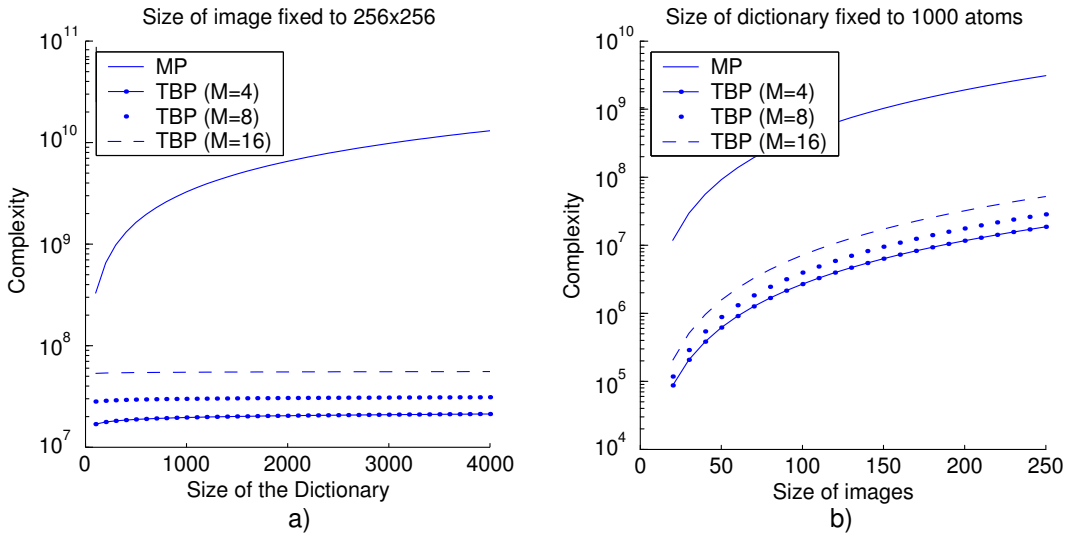

Fig. 3. Complexity comparison between Tree-Based Pursuit (TBP) and Matching Pursuit (MP). a) the size of the image to approximate has been fixed. b) the size of the dictionary has been fixed to 1000 atoms and the size of the images go from 20×20 up to 250×250.

We introduce a slight modification of equation 8 that will forbid the existence of atoms that are equidistant to different centroids. Equation 15 has to hold for all non-leaf nodes except for the root node.

$$\min_{i \in L_{l,n}} d(g_i, c_{l,n}) > \max_{j \in \mathcal{D} \setminus L_{l,n}} d(g_j, c_{l,n}). \tag{15}$$

We are looking for the worst case in which it is still possible to recover exactly the original signal s. Thus, we are interested in some extreme values and properties related to the centroids which are used to take decisions during the search. We are particularly curious about the representation capabilities of the centroids.

$$\lambda^B_{l,n} = \max_{j \in [nM,(n+1)M-1]} \max_{g_i \in \mathcal{D} \setminus L_{l+1,j}} | < c_{l+1,j}, g_i > |. \tag{16}$$

The value $\lambda^B_{l,n}$ is a measure of the correlation between the centroids of sibling nodes (children of $N_{l,n}$) and the atoms of the dictionary not contained in their respective subtree. Its value expresses the coherence between a set of centroids used by the search algorithm and the subset of the dictionary they do not represent. It is a measure of the perturbations that could be introduced by other atoms of the dictionary when looking for the best subtree based on the projections of the centroids on the residual.

From equation 9, we know that there are cases that are not favorable; i.e. for a centroid $c_{n,l}$ there exists linear combination of atom $A_{l,n}b \neq 0$ such that $c^*_{l,n} A_{l,n} b = 0$. In general, it is not possible to cover the whole span of a collection of atoms by an unique one.

The matrix $A^I_{l,n}$ contains in its columns the atoms of A contained in the subtree of the node $N_{l,n}$. The weights corresponding to these atoms are in $b^I_{l,n}$. In an analog way, we define $A^O_{l,n}$ and $b^O_{l,n}$ for the atoms of s that are not in the subtree of $N_{l,n}$.

Theorem 1: Tree-Based Pursuit recovers exactly the signal s if at each internal node $N_{l,n}$ (except the root node) containing at least one atom participating in s, the following condition is satisfied for all matrices $K \neq 0$ having 0 outside of the diagonal and 0 or 1 on the diagonal.

$$|c_{l,n} A^I_{l,n} K b^I_{l,n}| > 2\lambda^B_{l-1,\lfloor n/M \rfloor}(||b||_1 - b_{min}). \tag{17}$$

Proof. The general idea of the proof is to find for a node $N_{l,n}$ the minimal condition that ensures that the algorithm will never choose a subtree containing no atom from the signal in its leaves.

At an arbitrary node $N_{l,n}$ in the tree during the search, we have to decide which subtree is most likely to hold a least one *good* atom of the signal $s = Ab$. To make the decision, the scalar product between s and the centroids associated to the children of the current node are computed. Let define the matrices $\phi_{l,n}$ and $\psi_{l,n}$ containing in their columns respectively the *good* and the *bad* centroids hold by the children of node $N_{l,n}$. A *good* centroid is such that the subtree of the associated node contains at least one atom participating in s with non-null associated weight.

If the signal we are trying to recover is made of only one atom from $\mathcal{D}$ then it is correctly found due to inequality 15.

If there are more than one atom participating in s then the recovery is not guaranteed as shown by equation 9. In order to be sure that the algorithm takes a *good* decision, the following inequality has to hold.

$$||\phi^*_{l,n} s||_\infty > ||\psi^*_{l,n} s||_\infty. \tag{18}$$

The number of bad centroid is k which is the number of columns of $\psi_{l,n}$. If $k = 0$ every child of the node we are considering contain atoms taking part in s. The algorithm mandatory takes a *good* decision. The interesting case is when k is not null. We first derive an upper bound to the right-hand term of equation 18.

$$\begin{aligned} ||\psi_{l,n}^* s||_\infty &= ||\psi_{l,n}^* Ab||_\infty & (19)\\ &\leq \lambda^B_{l-1,\lfloor n/M \rfloor}||b||_1. & (20)\end{aligned}$$

Let c_i be the i^{th} column of the matrix $\phi_{l,n}$. And A_i^I the atoms from A that are located in the subtree of the node corresponding to centroid c_i. In an analog way, A_i^O are the atoms that are not associated to this centroid. The vector b, containing the weights, is also divided in two parts. The signal can be expressed as:

$$Ab = A_i^I b_i^I + A_i^O b_i^O. \quad (21)$$

We now derive a lower bound to the right hand term of equation 18.

$$\begin{aligned} ||\phi_{l,n}^* s||_\infty &= ||\phi_{l,n}^* Ab||_\infty & (22)\\ &\geq \min_i |c_i^* Ab| & (23)\\ &= \min_i |c_i^* A_i^I b_i^I + c_i^* A_i^O b_i^O| & (24)\\ &\geq \min_i (|c_i^* A_i^I b_i^I| - |c_i^* A_i^O b_i^O|) & (25)\\ &\geq \min_i |c_i^* A_i^I b_i^I| - \max_i |c_i^* A_i^O b_i^O| & (26)\\ &\geq \min_i |c_i^* A_i^I b_i^I| - \lambda^B_{l-1,\lfloor n/M \rfloor}||b_i^O||_1 & (27)\\ &\geq \min_i |c_i^* A_i^I b_i^I| - \lambda^B_{l-1,\lfloor n/M \rfloor}||b||_1 + b_{min}. & (28)\end{aligned}$$

Using equations 18, 20 and 28 it is possible to derive the following condition on the signal s to ensure that the algorithm takes the right decision at the current step.

$$\min_i |c_i^* A_i^I b_i^I| > 2\lambda^B_{l-1,\lfloor n/M \rfloor}||b||_1 - \lambda^B_{l-1,\lfloor n/M \rfloor} b_{min}. \quad (29)$$

If for the current signal s inequality 29 holds at each node that the search algorithm explores, then it will eventually find a correct atom of s. If the subtree of node $N_{l,n}$ contains more than on atom of A, then the pursuit algorithm will eventually pass again in $N_{l,n}$. As it is not possible to know in which order the atoms are recovered, inequality 29 has to be true for all possible subsets of A_i. It introduces the matrix K of equation 17.

If the condition proposed there is satisfied, the algorithm will make the right choice at the current step. If there are more than one correct atoms in the currently explored subtree, and that the algorithm works *properly* then, we are going to pass again in this node. The current condition is a sufficient for making the good choice once. Thus, to be sure that at the current node the decision will always be correct, the condition has to be satisfied for all possible combinations of weighed atoms in s in the subtree of the children of the current node.

If $\lambda^B_{l-1,\lfloor n/M \rfloor} = 0$, meaning that the different centroids are orthogonal to the rest of the dictionary, then the condition implies that there should not be a combination of vectors of $A_i^I K b_i$ orthogonal to the centroid c_i. This is closely related to the fact that in general it is not possible to represent the whole span of a bunch of vectors by an unique element (eq. 9).

VI. Results

The definition of the tree creation presented in section III does not present the technical choices made for the experiments. The used clustering algorithm was the *k-means* [9] [6]. The *k-means* algorithm tries to iterativerly maximize $Q_{L_{l,n},\lambda}$. The computation is over when the gain is less than a fixed ϵ; ϵ can be made arbitrarily small. This algorithm iteratively updates the centroid that are computed as a simple sign weighted sum described by algorithm 3. The λ in equation 11 has been fixed to 0 implying that only intra-class distance is used.

The tree is built using centered versions of the atoms which implies the translation parameter to be constant. Figure 4 shows the centroids at the first level of the tree. They are used by the algorithm to compute all possible scalar products and to make the initial choice of the subtree and the area to explore. The figure illustrates the fact that the initial centroid represents the different shape of atoms at our disposal in $\mathcal{D}$. The second line shows them in the frequency domain. Again, each of them represents a special behavior.

The test image is Lena (size 128x128). The reference search strategy is a Matching Pursuit that computes all the possible scalar products by making multiplications in the Fourier domain. It is the reference matching pursuit we used in section IV for the complexity comparison.

Algorithm 3 Centroid update for a group G of atoms

$c = 0$
for all $g_i \in G$ **do**
  if $\langle c|g_i\rangle \geq 0$ **then**
    $c = c + g_i$
  else
    $c = c - g_i$
  end if
end for
Normalize c to have unit energy.

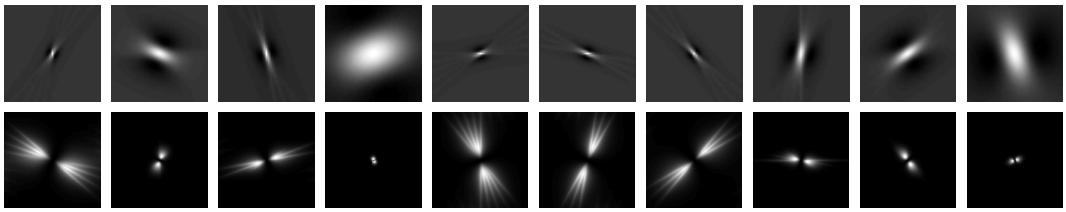

Fig. 4. Centroids at the first level of the tree. The first line shows the atoms in the spatial domain (inner third), the second shows them in the frequency domain (logarithm of the scale).

The same Matching Pursuit was used in our algorithm at the first level of the tree. If the number of children per nodes is $|\mathcal{D}|$, the proposed algorithm and the reference Matching Pursuit would do the same. The computation time and the results would also be the same. The presented results where obtained with a number of children per node under 20 (less than 1% of the size of the dictionary).

Most of the complexity of the search procedure remains in the first M full searches to execute in order to locate the best subtree at the first level. The total amount of time needed to finish highly depends on the choice of M. On the other hand, if this value is too small, the atoms will not be perfectly found. It comes from the fact that there are not enough centroid to represent the diversity of atoms contained in the original dictionary $\mathcal{D}$. As illustrated by figure 5 the reconstruction error rapidly decreases a M increases.

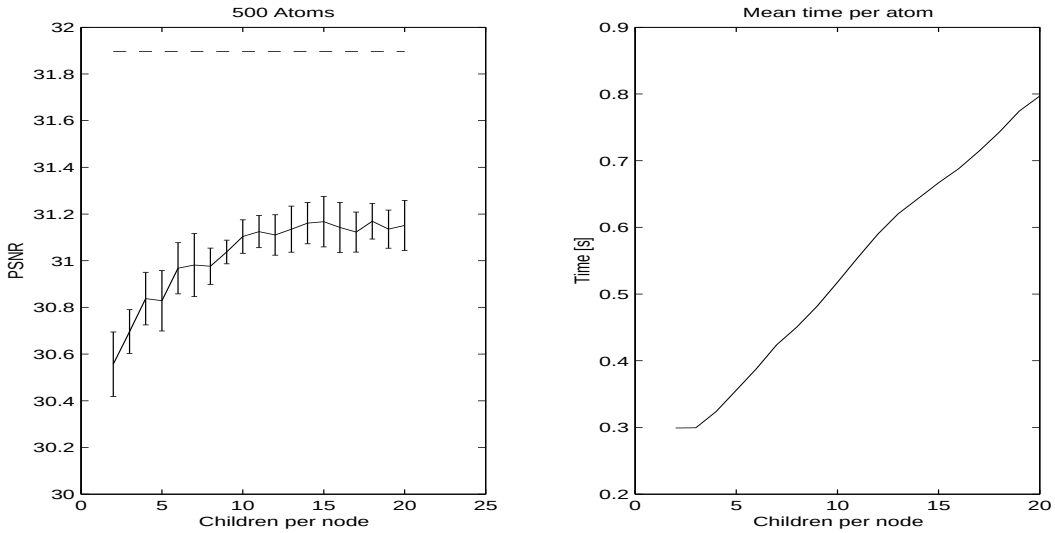

Fig. 5. Reconstruction error given the number of nodes for 500 atoms. Dashed line shows the results with a full search algorithm. Mean time to find an atom.

Figure 5 also shows the amount of time per atom given the number of children per node in the tree. On can see that the it is quasi linear on the range we are studying. To compare, our reference full-search needed about 47 seconds per atom. Thus, the proposed algorithm is about 50-150 times faster.

In the case of real block-orthogonal dictionaries, the results of our search strategy and the full-search would be the same. Even in the non-favorable case proposed as example here, it performs well.

VII. Conclusion and Future work

This paper presented the potential of the Tree-Based Pursuit algorithm as an efficient method to limit the complexity of overcomplete signal expansions. It is shown to decrease the complexity of the signal decomposition with respect to Matching Pursuit, while the quality of the resulting approximation is kept quite satisfactory, depending however on the clustering algorithm and distance measures used for the creation of the tree. Future work will investigate more efficient clustering algorithms, taking into account inter-cluster distance, since the *k-means* clustering algorithm only considers intra-cluster distances. The representing capabilities of the centroid and the impact on the Pursuit algorithm will also be investigated. A tree is a natural manner to encode information, we will try to exploit these intrinsic properties by coding the path instead of the atoms indices. We will also extend this to encode a centroid instead of atoms if it is worth regarding a rate-distortion criterion.

References

[1] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1999.

[2] Albert Cohen, Wolfgang Dahmen, Ingrid Daubechies, and Ronald DeVore. Tree approximation and optimal encoding. *Appl. Comput. Harmon. Anal.*, 11(2):192–226, 2001.

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

[4] Geoff Davis, Stphane Mallat, and Marco Avellaneda. Adaptive greedy approximations. *Journal of Constructive Approximation*, 13:57–98, 1997.

[5] Vandergheynst P Figueras i Ventura R and Frossard P. Highly flexible image coding using non-linear representations. Technical Report TR-ITS-2003.002, 1015 Ecublens, June 2003.

[6] E. Forgey. Cluster analysis of multivariate data: Efficiency vs. interpretability of classification. *Biometrics*, 21:768, 1965.

[7] Frossard P. and Vandergheynst P. and Figueras i Ventura R. M. High flexibility scalable image coding. In *Proceedings of VCIP 2003*, pages 281–296, July 2003.

[8] R. Gribonval and M. Nielsen. Approximation with highly redundant dictionaries. In *Wavelets: Applications in Signal and Image Processing, Proc. SPIE'03*, 2003.

[9] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkley Symp. Math. Statistics and Probabiliy*, volume 1, pages 281–296, 1967.

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

[11] R. Neff and A. Zakhor. Dictionary approximation for matching pursuit video coding. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, volume 2, pages 828–831, Sept. 2000.

[12] J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. ICES Report 03-04, The University of Texas at Austin, February 2003.

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