A Matching Pursuit Full Search Algorithm for Image Approximations

Rosa M. Figueras i Ventura, Oscar Divorra Escoda and Pierre Vandergheynst
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Signal Processing Institute (ITS)
CH-1015 Lausanne, Switzerland
e-mail: {rosa.figueras,oscar.divorra,pierre.vandergheynst}@epfl.ch

Technical Report No. 31/2004

Abstract

There is a growing interest on adapted signal expansions for efficient sparse approximations. For this purpose, signal expansions on over-complete bases are of high interest. Several strategies exist in order to get sparse approximations of a signal as a superposition of functions from a redundant dictionary. One of these strategies is the well known Matching Pursuit (MP). MP is an algorithm where complexity depends on the accuracy of the desired approximation. This is due to its greedy iterative nature. For very large dictionaries, however, complexity depends in great manner on the size of this, i.e. at each iteration the whole dictionary has to be browsed. Sometimes, heuristic procedures need to be adopted due to the overwhelming complexity that this may represent. However, these reduce the search space and, consequently, a poorer signal approximation is retrieved. In this work, we propose a feasible approach for Full Search Matching Pursuit (FSMP) for the particular case of natural image approximations with an-isotropically refined oriented atoms (which have the purpose of exploiting image geometry). Thanks to the structure of the dictionary and its spatio-temporal localisation, several enhancements are possible to speed-up the calculation of the most critical step: the scalar product of the signal with all the functions from the dictionary.

Index Terms

Matching Pursuit, Full Search, Fast Algorithm, FFT, Sparse Approximations, Redundant Dictionaries, Anisotropic Refinement Atoms, Edges, Image Geometry, Memory Compression, Steerability, Complexity Optimization.

Contents

Web page: http://lts2www.epfl.ch
This report concerns the work done during automn 2002 for the developement of a full search Matching Pursuit algorithm using AR atoms for image approximations.
The work of Oscar Divorra Escoda was sponsored by Swiss Federal Office for Education and Technology grant number 6044.1 KTS.

I. Introduction

Efficient image representation has been, and still is, a challenge. After considering images as a set of pixels for a long time, image representation in the Transformed Domain was studied. The first tools that were used to solve the image representation problem in the transformed domain were Fourier Transforms, Windowed Fourier Transforms or tools based on the properties of the Fourier Transform like DCT (see [1]). These tools have the inconvenience of considering the signal stationary. The next step toward efficient image representation was through the use of wavelets, which did not consider the signal stationary any more. Wavelet transforms used were mainly obtained through the tensor product of 1D wavelet basis, forming 2D separable orthogonal wavelet basis [1]. The use of wavelets meant a good increase of representation efficiency for piecewise smooth signals [2]. It has been proved, however, that orthogonal separable basis do not produce the best possible expansion of piecewise-smooth 2D signals [3]. In order to be able to achieve sparser representations, correlation along contours and geometry have to be taken into account. This is not straightforward to do with orthogonal basis, due to the large variety of contour features. A sparse representation of an image may be achieved if the image is represented through a redundant dictionary. It has recently been shown that this redundant dictionary should include anisotropy and rotations, as well as translations. Intuitively, this can be seen with the fact that an image can have the same structure but zoomed (so isotropic dilations are needed), rotated (so rotations are needed) and translated (so, translations are needed as well). However, due to the fact that edges are anisotropic structures, anisotropy is as well a good property to include in the set of functions that form the redundant dictionary. Some experiments, like [4] show these needs experimentally.

The main drawback of redundant dictionaries is that finding the sparsest solution of the span of a signal on them is an NP hard problem. Greedy algorithms, under certain conditions and with certain dictionaries of functions [5], may give a sparse approximation of a signal. Hence, they must be considered as an interesting tool.

Among greedy algorithms, the simplest one might be Matching Pursuit. Matching Pursuit (MP) tries to minimize the representation error at every iteration, and it has been proved in [6] that this algorithm may achieve sparsity. MP seems to be an appropriate tool to deal with redundant dictionaries. It gives a way to solve the posed non-linear problem. In addition it provides some properties quite interesting for image processing, such as a progressive signal decomposition.

A. Image Model

In order to represent a signal efficiently, there is a need to know some features of this signal and to make some assumptions. The main assumption that it is done with images is that they can be represented (or at least approximated) as a finite sum of basis functions, with a number of terms smaller than the dimensionality of the signal:

$$f = \sum_{n=0}^{N-1} c_n g_{\gamma_n} . \tag{1}$$

This kind of decompositions are straightforward to obtain when one uses an orthonormal basis or a frame method [1]. However, frame methods do not usually provide sparse enough signal approximations. In addition, computing the dual basis is not always straightforward. When dealing with redundant dictionaries, thus, there is the need of having an algorithm that gives a sparse decomposition, and greedy algorithms seem a solution.

The sparse image model does not make any assumption about the nature of the basis functions. In order to have efficient representations, basis functions have to be adapted to the specific features of natural images, e.g. contours. Contours are often assumed to be continuous smooth functions [7], [8], [3], [9]. Taking this model and trying to approximate any 1D function with the smallest number of piecewise linear segments, one can see that there is a need to be adaptive in length, orientation and location [9]. This implies that the over-complete dictionary in use may be composed by a diverse set of an-isotropically scaled, rotated and translated versions of a mother basis function.

B. Dictionary of basis functions

The dictionary used in the scope of this work is generated by applying a set of geometrical transformations (anisotropic scaling, rotation and translation) to the following mother function:

$$g_{AR}(x,y) = K(4x^2 - 2)e^{-(x^2+y^2)} . \tag{2}$$

The generated dictionary (whose functions are called anisotropic refinement (AR) atoms [10]) has the property of giving dilation, rotation and translation covariant image decompositions. An example of this kind of atoms is given in Fig. 1.

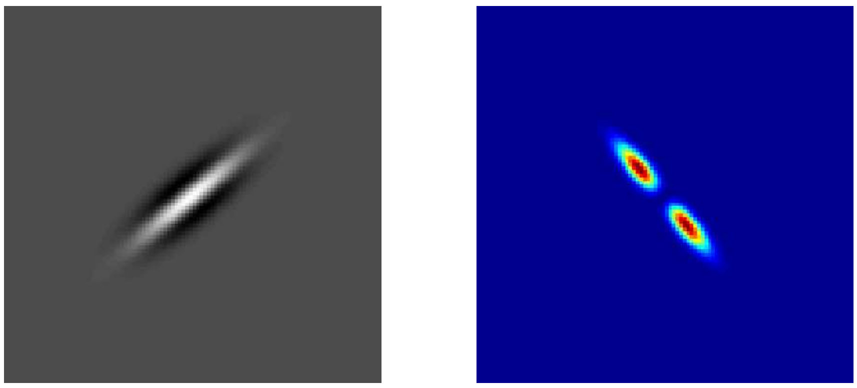

(a) Spatial domain (b) Frequency domain

Fig. 1. An example of one atom of the dictionary in the spatial domain and in the frequency domain.

These atoms have been chosen because they can be easily an-isotropically scaled and rotated, and they have good behavior for edge representation: when an anisotropic refinement atom is on an edge, his response will be maximal when the function is exactly oriented parallel to the edge. This comes from the fact that this atoms are, in one axis, wavelet functions with two vanishing moments. The choice of two vanishing moments comes from the fact that it is a large enough number in order to ignore polynomial surfaces but small enough to avoid oscillatory effects, quite annoying to the human eye. In addition, this kind of functions are very similar to the kind of functions used in the V1 cells [4], which makes coding artifacts smooth enough not to be too annoying for the Human observer.

The atoms used in this work do not represent properly the low-pass frequency of images. Thus, this part needs to be treated apart. Before expanding the signal onto the dictionary, its low-pass frequency part is subtracted from the signal and represented by its downscaled version.

C. Purpose of this Work

In this work, we propose a feasible approach for Full Search Matching Pursuit (FSMP) in the particular case of natural image approximations with an-isotropically refined oriented atoms. The purpose of the use of this kind of atoms is to exploit image geometry. Thanks to the structure of the dictionary and its spatio-temporal localization, several enhancements are possible to speed-up the calculation of the most critical step: the scalar product of the signal with all the functions from the dictionary.

This report is structured in the following way: In Sec. II the Matching Pursuit principle is reviewed. Next, we recall in Sec. III a sub-optimal MP approach based on the genetic algorithm and developed for image approximation [11], [12], [13]. Once situated the problem, the proposed feasible Full Search MP approach is described in Sec. IV. In this, we expose the usage of the FFT to accelerate the scalar products computation as well as some comparative results with genetic algorithm based MP. The overall result is that our FSMP is not only faster than GA based MP but it is also able to recover much better approximations. Finally, conclusions are drawn in Sec. VI.

II. Matching Pursuit

A way of retrieving a signal approximation based on the model of Eq. (1) is by using the so called Matching Pursuit algorithm. Matching Pursuit was first defined by Mallat [1] and Mallat and Zhang [14] as *a greedy algorithm that decomposes any signal into a linear expansion of waveforms taken from a redundant dictionary.* These waveforms are iteratively chosen to best match the signal structures, producing a sub-optimal expansion. Vectors are selected one by one from the dictionary, while optimizing the signal approximation (minimizing error energy) at each step. Even though the expansion is linear, it gives a non-linear signal decomposition.

MP has been proved to be able to give sparse signal approximations when certain dictionary properties are kept [6]. Due to the great interest sparse signal approximations have for the image coding community, computationally efficient implementations of MP are worth being studied.

Let $\mathcal{D} = \{g_\gamma\}_{\gamma\in\Gamma}$ (where Γ is the set of valid indexes) be a dictionary of P atoms (where $P > N$, being N the dimensionality of the signal) having unit norm. This dictionary includes N linearly independent vectors that define a basis of the space $\mathbb{C}^N$ of signals of size N. Let $R^n f$ be the residual of an n term approximation of a given signal f. A Matching Pursuit is an iterative algorithm that sub-decomposes the residue $R^n f$ by projecting it on a vector of $\mathcal{D}$

that matches $R^n f$ at best. If we consider $R^0 f = f$, the first MP iteration will represent the signal as:

$$f = R^0 f = \langle f, g_{\gamma_0} \rangle g_{\gamma_0} + R^1 f \,, \tag{3}$$

where $R^1 f$ is the residual vector after approximating $R^0 f$ in the direction of g_{γ_0}. Since $R^1 f$ is orthogonal to g_{γ_0}, the module of f will be:

$$\|R^0 f\|_2^2 = |\langle R^0 f, g_{\gamma_0} \rangle|_2^2 + \|R^1 f\|_2^2 \,. \tag{4}$$

As the term that must be minimized is the error $\|R^1 f\|$,

$$\|R^1 f\|_2^2 = \|R^0 f\|^2 - |\langle R^0 f, g_{\gamma_0} \rangle|^2 \,, \tag{5}$$

the $g_\gamma \in \mathcal{D}$ to be chosen is the one that maximizes $|\langle R^0 f, g_{\gamma_0} \rangle|$, or, generalizing, $|\langle R^n f, g_{\gamma_n} \rangle|$. In some cases it is not computationally efficient to find the optimal solution, and a suboptimal solution may be computed instead:

$$|\langle R^n f, g_{\gamma_0} \rangle| \geqslant \alpha \sup_{\gamma \in \Gamma} |\langle R^n f, g_\gamma \rangle| \,, \tag{6}$$

where $\alpha \in (0, 1]$ is an optimality factor which is 1 when the optimal solution is chosen. This sub-optimality factor α depends on the searching method used to find the solution (see Sec. III for an example).

From (3), one easily sees by induction that the N term decomposition of f is given by:

$$f = \sum_{n=0}^{N-1} \langle R^n f, g_{\gamma_n} \rangle g_{\gamma_n} + R^N f \tag{7}$$

and with the same principle we can also deduce from (4) that the L^2 norm of the signal f is:

$$\|f\|^2 = \sum_{n=0}^{N-1} |\langle R^n f, g_{\gamma_n} \rangle|^2 + \|R^N f\|^2 \,, \tag{8}$$

where $\|R^N f\|$, when dealing with finite dimension signals, converges exponentially to 0 when N tends to infinity and $M_1 \times M_2$ is finite (see [15] for a proof).

Matching Pursuit decomposition efficiency is highly dependent on the dictionary adaptation to the signal to represent: The more adapted is the dictionary to the signal, the quicker the energy of the residual will decrease.

The goal of this work is to establish a feasible Full Search Matching Pursuit algorithm (i.e. an algorithm with $\alpha = 1$) for the particular case of image approximations with the dictionary of Sec. I-B.

III. Genetic Algorithm based MP: a Weak Matching Pursuit Implementation

The straight-forward computation of all scalar products required by full search Matching Pursuit is a computationally hard problem. In order to reduce the computation time, some clever optimization strategy may be taken. Due to the non-linearity and non-convexity of the problem, standard minimization algorithms, such as steepest descent, do not perform well, due to the presence of many local minimas. One way of implementing a quicker computation of a global solution is through the use of Genetic Algorithms [11], [12], [13]. Matching Pursuit, when implemented with a sub-optimal research algorithm, is called Weak Matching Pursuit (WMP). Indeed, instead of choosing the optimal solution at every iteration, a sub-optimal one is selected instead.

Genetic Algorithms may be appropriate for MP because they can converge no matter the local minima of the problem is. The disadvantage of this kind of algorithms is that the convergence degree is merely statistical. One can never know how close to the optimal solution is the one retrieved. However, greedy algorithms have been demonstrated to converge even though the solution chosen at every iteration is not the optimal [16]. Hence, the use of a Genetic Algorithm will not cause the algorithm to diverge, but it will slow down the convergence.

The use of Genetic Algorithm (GA) in the framework of MP for image approximations was proposed in [12], [17]. The GA takes the parameters of the atoms (translation, scaling and rotation) as *genes*. Each group of five parameters that define an atom is an *individual*. The group of N_{ind} atoms being evaluated in a certain moment, or the group of N_{ind} individuals alive in a certain moment is the *population*. The individuals of a *population* will have descendants through *crossover*, *mutations* and *survival of the fittest*, which will form the new *generation* of the *population*. Normally the algorithm will go on until a certain number N_g of generations is achieved or until the minimal error has been reached. A diagram of how this algorithm works is depicted in Fig. 2.

The computational complexity of the GA will directly depend on the number of individuals in the population and on the number of generations allowed. The number of scalar products to compute in order to obtain an MP coefficient through the GA will be at most $N_{ind} \cdot N_g - N_g + 1$ (from one generation to the other, some individuals can be kept without change due to the randomness of mutations and crossovers). In effect, the computation of the winner is done only once, because then it is passed from one generation to the other. For a general dictionary (so, dictionary atoms

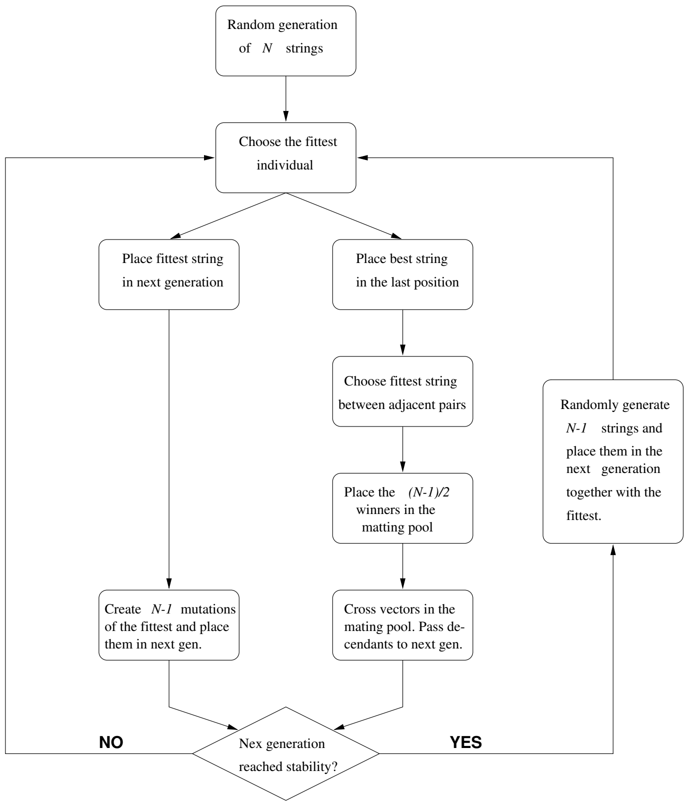

Random generation of N strings
Choose the fittest individual
Place fittest string in next generation
Place best string in the last position
Choose fittest string between adjacent pairs
Randomly generate N-1 strings and place them in the next generation together with the fittest.
Place the (N-1)/2 winners in the matting pool
Create N-1 mutations of the fittest and place them in next gen.
Cross vectors in the mating pool. Pass decendants to next gen.
NO
Nex generation reached stability?
YES

Fig. 2. Genetic Algorithm block diagram

are not necessarily bounded, and their size is taken equal to the image size), performing a scalar product will imply about N products and sums. The final computational complexity of the GA will be given by:

$$O\left(N \cdot N_{ind} \cdot N_g\right). \tag{9}$$

When compared to the "Brute Force" Full Search MP implementation (see section IV-A), this complexity is lower. However, the price to pay for this lower complexity is quite high, specially if one takes into account that GA introduces a loss of approximation fidelity which is difficult to quantize due to the randomness of the method.

IV. Full Search MP

Sub-optimal algorithms can advantageously reduce computational complexity. As seen in Sec. III, however, there is a price to be paid in approximation fidelity. This can affect negatively the performance of the application that uses the generated signal expansion. In fact, errors committed during function selection will require further expansion terms in order to be corrected. In a "Full search" case, approximations sparsity would be maximized.

A. "Brute Force" Full Search MP

According to the MP algorithm principle (see Sec. II), at every iteration, it is necessary to browse through all the functions that conform the given dictionary to perform all scalar products between dictionary atoms and the signal. Hence, the complexity of a full search MP iteration directly depends on the size of the dictionary times the complexity of a single scalar product. In a general case, and for any general dictionary, scalar products can be assumed to involve all the pixels of the image (non-separable atoms with no compact support). In this manner, and considering both contributions,

$$O\left(N_F \cdot N\right) \tag{10}$$

operations are needed to compute all the scalar products, where $N = Size_x \cdot Size_y$, as defined in Sec. II and N_F the number of dictionary functions.

In Sec. I we discussed that the use of very dense dictionaries may be of interest in image representations. The choice of atoms in MP expansions looks forward to finding the best spatio-frequential approximation of an image. The fact that precise localization of "spatio-frequential" features is required, all possible displacements for a certain shape of atom need to be taken into account. This implies that most of the functions that compose a certain dictionary are just the displaced version of a frequential model. Thus,

$$N_F = M_F \cdot Size_x \cdot Size_y = M_F \cdot N, \tag{11}$$

where M_F is the number of frequential models.

It follows from Eq. (11) that the complexity of a full search MP with a dictionary such as the described above is

$$O\left(M_F \cdot N^2\right) . \tag{12}$$

Of course, the cost of generating the functions that compose the dictionary is not taken into account in this estimation of complexity. We will assume that functions are generated once and stored in memory.

Given the dictionary example of Sec. I, we see that the dictionary generation is also an important point to take into account concerning the complexity of the whole expansion algorithm. Actually, the fact of having an analytical generating expression for the dictionary functions may lead to compute at every step the concerned atom. This, fruit of some pragmatism, from a programing point of view, may carry a very important overhead on the computations. The cost of computing a general atom of the image dimensions is *a priori* related with the size of the image. Hence, assuming that the number of operations (Δ) needed to compute a pixel is $\Delta << N$, complexity turns to be $O(N)$. Anyway, this bound can be not representative of the real complexity. Depending on the function to be computed at every pixel (see Eq. (2)), then considering $O(N \cdot \Delta)$ may be required.

In order to reduce the impact of Δ some part of the analytical atom expression may be heavy to compute, look-up tables may be defined with a sufficiently dense set of pre-computed values of the critical function (e. g. "exp()" in Eq. (2)). The corresponding value can be approximated, then, by its nearest neighbor in the look-up table. This approach can be very interesting in cases where all atoms from the dictionary are generated using the same analytical expression.

A more exhaustive solution can be to store all the functions that compose the dictionary such that no extra calculations are needed a part from the scalar products and the memory transfers. This is equivalent to dispose of a numerically defined dictionary. In here, the complexity problem of computing all atoms, is completely turned into a memory problem in terms of size and bandwidth. For a general dictionary, $O(N_F \cdot N)$ memory words would be required. This can absolutely be impractical for a very dense dictionary.

Coming back to our dictionary (Eq. (12)), the fact of having $O(N)$ functions per each of the M_F frequency models would represent a $O\left(M_F \cdot N^2\right)$ in terms of memory requirements. Thus, it would be absolutely impracticable. However, the dictionary functions are translation invariant (except for a normalization factor). Simply with a centered version for each one of the M_F models, a simple convolution would solve the scalar products for all translations of a frequency models.

B. Spatial Invariance in Scalar Product Computations and Boundary Renormalization

Consider the operations performed in one iteration of the full search MP:

$$\left|\langle R_f^n, g_{\gamma_n}\rangle\right| = \sup_{\gamma\in\Gamma}\left|\langle R_f^n, g_\gamma\rangle\right| , \tag{13}$$

and the translation invariance of each of the frequency models, then

$$\left|\langle R_f^n, g_\gamma\rangle\right| = \left|\langle R_f^n, g_{\gamma_{M_F}^{dx,dy}}\rangle\right| = \sum_\tau \sum_\lambda R_f^n(\tau,\lambda) g_{\gamma_{M_F}}(\tau - dx, \lambda - dy), \tag{14}$$

where $\gamma_{M_F} \in \Gamma_{M_F}$ and Γ_{M_F} is the sub-set of different frequency models in Γ. Γ_{M_F} is the set of all possible different shapes that the functions of the dictionary may have (independently of the spatial position). In (14), $\gamma_{M_F}^{dx,dy}$ indexes all the different functions included in the dictionary. That is, all the different frequency models Γ_{M_F} for every spatial location (dx, dy).

A common problem to be addressed in image processing is the finite nature of the signals. Images are defined on a finite domain. If this is not taken into account, then sparsity in representations may decrease.

An example is the representation of an image through the classical wavelet transform. When no considerations are taken on the boundaries (simple zero padding), there will be a large amount of highly valued wavelet coefficients on the image border. Several solutions have been studied to solve this. Common ones are periodic extension of the signal, mirror folding or the use of specific boundary wavelets [1].

An image $I(x, y)$ defined in a domain s.t. $x, y \in [0, Size_{x,y})$ can be considered as the windowed version of a infinite 2-D signal I_{inf}:

$$I(x,y) = \Pi\left(\frac{x - Size_x/2}{Size_x}, \frac{y - Size_y/2}{Size_y}\right) I_{inf}(x, y) , \tag{15}$$

where $\Pi(x, y)$ is a squared 2-D window defined for $x, y \in [0, 1]$.

In what concerns our purposes, the possible use of periodic extensions of the image would definitely have a negative impact in the number of functions needed to represent the boundary, i.e. artificial structures would be introduced. Mirroring would be a better choice and the energy spreading on the coefficients would be significantly lessen. Anyway, this approach introduces also artificial structures such as bending of oriented features (e. g. edges) on the image boundary.

A possible solution can be to consider into the dictionary an additional variant of atoms. The proposed functions would be nothing else than the truncated versions of the AR atoms (according to the image size). Once truncated, these need to be re-weighted with a re-normalization factor such that they continue to have unitary norm. In this way, coefficient expansions continue to have the same properties defined for MP signal decomposition on unitary dictionaries. The contribution of this new class of elementary functions is the ability to catch regular structures of the signal (in the same way as the windowed ones) that suddenly terminate at the image boundary.

The scalar product generation can thus be reformulated, considering the image boundaries, as:

$$\begin{gathered}
|\langle R_I^n, g_\gamma\rangle| = \left|\langle R_I^n, g_{\gamma_{M_F}^{dx,dy}}\rangle\right| = \\
\textstyle\sum_\tau \sum_\lambda R_I^n(\tau,\lambda) \dfrac{g_{\gamma_{M_F}}(\tau-dx,\lambda-dy)}{\left\| g_{\gamma_{M_F}}(\tau-dx,\lambda-dy)\cdot \Pi\left(\dfrac{\tau-dx-Size_x/2}{Size_x},\dfrac{\lambda-dy-Size_y/2}{Size_y}\right)\right\|_2}\cdot \\
\Pi\left(\frac{\tau-dx-Size_x/2}{Size_x},\frac{\lambda-dy-Size_y/2}{Size_y}\right) = \\
\sum_\tau\sum_\lambda R_f^n(\tau,\lambda)\cdot \tilde{g}_{\gamma_{M_F}}(\tau-dx,\lambda-dy),
\end{gathered} \tag{16}$$

where $\tilde{g}_\gamma$ represents the new dictionary of functions that is space invariant except for the re-normalization factor that depends on the spatial location.

One can consider to separate the calculation of scalar products into two steps. First, the non re-weighted, translation invariant part of atoms (i.e. the frequency model) would be used in the convolution. Afterward, each obtained coefficient should be re-weighted by the corresponding normalization factor in order to take into account image boundaries effect.

The computation of the normalization factors implies an additional increase of complexity. In a general dictionary, this complexity would be dependent on the total number of functions of the dictionary and the size of the signal. Considering the general situation where all dictionary functions may intersect with the boundary independently of its position, it turns out that N_F normalization factors are needed and

$$O\left(M_F\cdot N^2\right) \tag{17}$$

operations are required to compute them. This normalizing factors are used at every iteration of the MP algorithm, and do not depend on the signal under analysis. This allows to compute them once at the beginning of the expansion process and to store them for their ulterior use in the subsequent MP iterations. Eq. (11) trivially implies the need of

$$O\left(M_F\cdot N\right) \tag{18}$$

memory words, which in some cases, and depending on the dictionary size (and image size), may be feasible.

C. FFT Based Full Search MP: From Scalar Products to Spatial Convolution

In sec. IV-A, computation of the whole set of scalar products has been presented as a very computationally complex task. Therefore, the use of a fast convolution algorithm based on the Fourier transform is to be considered in order to make possible a full search strategy for large dictionaries. Then (12) is substituted by

$$O\left(M_F\cdot N\log\left(N\right)\right). \tag{19}$$

Furthermore, functions can be directly stored into memory in their transformed version (hence, a forward FFT per model and iteration can be spared). In the case of a dictionary of non-complex functions, the amount of memory needed for the Fourier-domain atom versions will not be bigger than the needed in spatial domain. However, in the transformed version of the function, the spatial zero padding will have to be taken into account. Furthermore, the mask of normalizing weights can be stored in the same way. This comes to further increases memory needs. Asymptotic memory requirements are:

$$O\left(M_F\cdot N\right) \tag{20}$$

memory words.

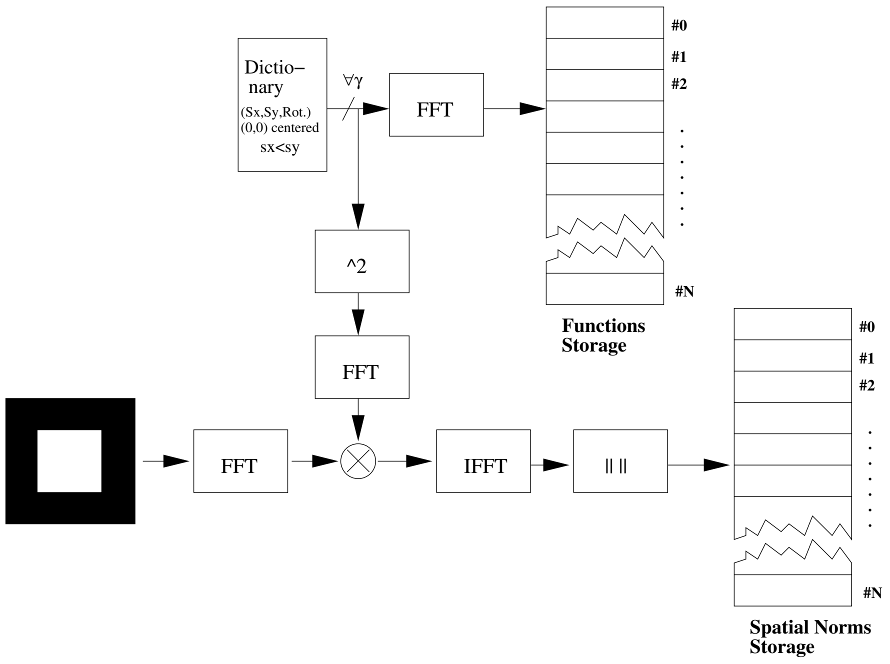


Fig. 3. Description of the setting up procedure to generate the look up tables to be used in fig. 4

Fig. 3 shows the scheme used to generate and store the dictionary atoms and weighting masks. Weighting masks are generated by convolving a centered atom by an image size pulse function of unitary amplitude.

Fig. 4 illustrates the general procedure used to generate all scalar products from the stored Fourier atoms and spatial weighting masks. Convolutions are performed on the Fourier domain. Once available the scalar products, as reviewed in Sec. II, only the supremum search remains to be done.

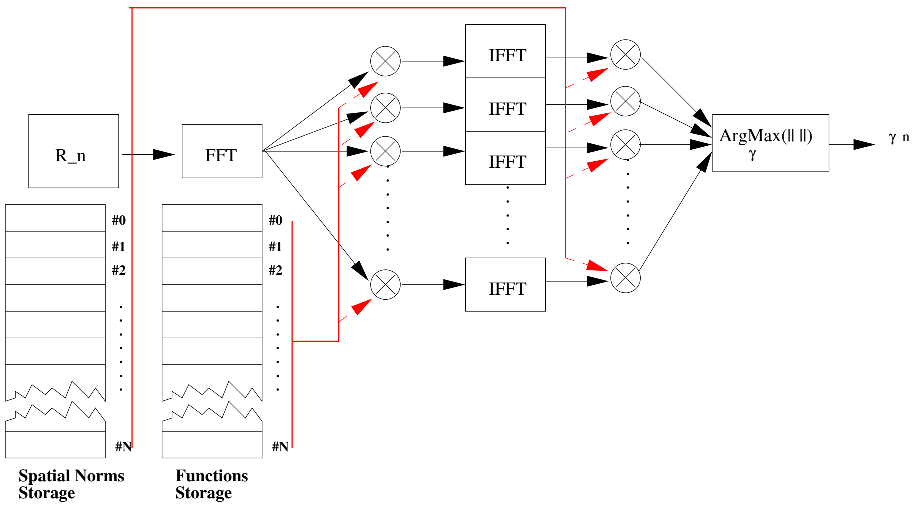


Fig. 4. Schematical description of the full search algorithm for one iteration of MP. Look up tables are used to hold the DFT of the dictionary functions and the normalizing masks.

As we show in the following sections our MP approach can benefit from several improvements in terms of computational cost and memory requirements when special care of the structure of the dictionary is also taken into account. Before, however, let us see some results of the full search algorithm compared to those obtained by the GA based approach.

D. Results: Full Search vs Genetic Algorithm Search

In this section, we compare the performance of the FSMP algorithm versus Weak-MP based on the GA. In table I, we state the relations of computation time required by each algorithm and the approximation quality achieved. The experiments have been performed on a Dual Xeon 2.2 GHz with 2GB of RAM platform. It is very relevant the improvement of 3.8 dBs registered in the piecewise-smooth like images (*lena* and *cameraman*) with a significantly smaller computational time. For the case of *baboon*, the improvement is only of 1.6 dBs due to the higher complexity of the signal (much more textured). In the three cases, approximations have been generated by recovering the first 300 terms of their MP expansions.

Image	Algorithm	Computation Time	PSNR
lena 128x128	GAMP	156’ 26.909”	26.1506 dB
	Full Search MP	126’ 29.792”	29.9076 dB
cameraman 128x128	GAMP	157’3.249”	25.0485 dB
	Full Search MP	128’13.957”	28.8270 dB
Baboon 128x128	GAMP	160’19.048”	25.1374 dB
	Full Search MP	128’9.170”	26.7932 dB

TABLE I
COMPARISON OF THE COMPUTATIONAL TIME AND THE IMAGE QUALITY OBTAINED WITH THE GENETIC ALGORITHM, USING 39 INDIVIDUALS AND 75 GENERATIONS, AND THE FULL SEARCH ALGORITHM.

More graphically, Figs. 5, 6, 7 show a visual comparison of both algorithms. In the FS case, exhaustive search at every iteration reduces the *geometric* noise introduced by the GA. GA, is not able to converge precisely at every iteration in order to select the best atom. This error appears under the form of lines in the signal. These appear very clearly in the neighborhood of relevant salient geometric structures like edges.

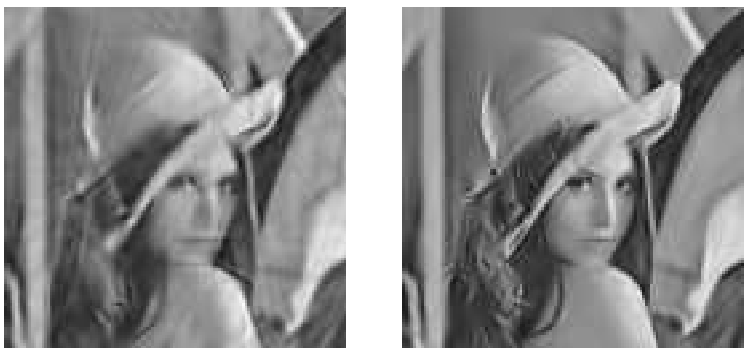

(a) Genetic Algorithm (b) Full Search Algorithm

Fig. 5. Visual comparison of lena 128x128 decomposed with MP with 300 coefficients (a) with the Genetic Algorithm and (b) with the Full Search MP.

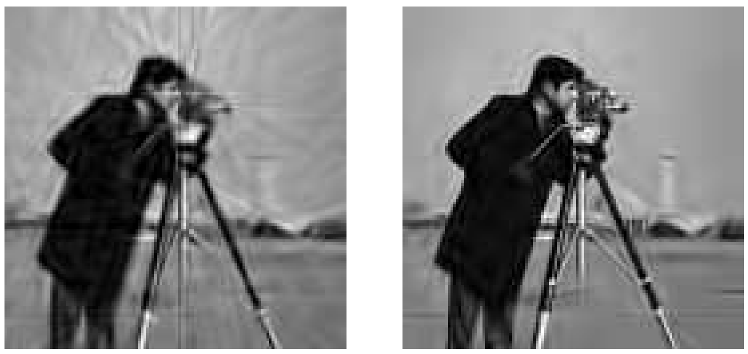
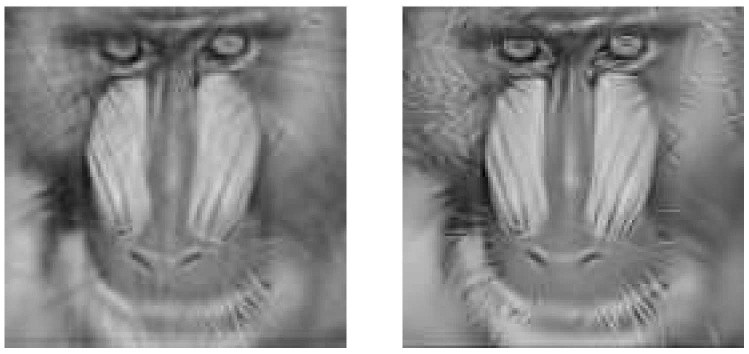

(a) Genetic Algorithm (b) Full Search Algorithm

Fig. 6. Visual comparison of cameraman 128x128 decomposed with MP with 300 coefficients (a) with the Genetic Algorithm and (b) with the Full Search MP.

Finally, to give an overview of the convergence through MP iterations, a comparison between FS and GA is given in Fig. 8. In the experiments, the image lena has used. As can be seen, the FS algorithm outperforms through all



(a) Genetic Algorithm (b) Full Search Algorithm

Fig. 7. Visual comparison of baboon 128x128 decomposed with MP with 300 coefficients (a) with the Genetic Algorithm and (b) with the Full Search MP.

iterations the weak greedy algorithm that uses the GA. We clearly see that it is worth the effort of looking for Full Search implementations of MP, specially considering the visual improvement it brings.

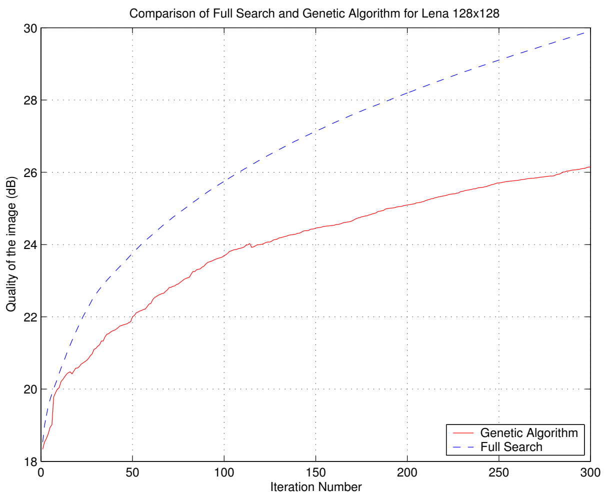


Fig. 8. Evolution of the PSNR with the iteration number for the Genetic and the Full Search Matching Pursuit algorithms. The image used in this case is the gray-scale 128x128 lena.

V. Exploiting the dictionary features

Complexity savings can be even higher if some specific features of the dictionary are taken into account. In this section, we explain how computational and memory complexity can be reduced. Several results are presented to illustrate the gains and the limits of the methods used.

A. *Exploiting Spatio-Temporal Energy Localization: Compact Support and Atoms Approximation*

In terms of computational complexity, it is very interesting in our full search MP approach to deal with compact support basis functions. We can define two sorts of compactness: compactness in frequency and compactness in space. Natural image decompositions often use dictionaries where functions have localized support in space, e.g. the basis functions used for classic dyadic wavelet decompositions [1]. In what concerns the dictionary used in this work, even if functions do not have, strictly, compact support, their energy is mostly compacted in a very localized fashion. Moreover, the fact that atoms envelop is Gaussian shaped implies that energy is localized in both domains: space and frequency. This, as seen in the following, allows for some approximations of the dictionary that will significantly reduce the usage of memory and will contribute to decrease the number of multiplications at every full search MP iteration without introducing any distortion nor affecting the rate of convergence of the MP algorithm.

A first advantage of spatially localized dictionaries is that one can use the so called M-fold MP strategy [18]. This consist in recovering M atoms at each MP iteration. Indeed, distant atoms in space that have a localized support, are often incoherent among them. Some heuristics can be based on this and contribute speeds-up by a factor M.

Thanks to the incoherence among spatially distant atoms, further savings are possible. Indeed, for a given MP iteration, ones an atom has been selected, all the scalar products of those atoms that are highly incoherent with the selected one do not need to be recomputed. This brings a possible way of parallelizing MP. Scalar products of different influence zones may be computed in parallel. This strategy, in some cases, can also be coupled with the M-fold approach leading to an even better general performance.

Our full search solution is based on the massive use of the FFT to reduce the complexity of computing the scalar products of all the translations of a given atom. For this purpose, all centered Fourier atom versions are stored into memory such that they need to be computed only once. The fact that an atom and the normalizing modulus masks (as described in Sec. IV) may have few significant non-zero samples in an image makes possible to optimize the storage by efficiently indicating where those are. Moreover, the run-length description of their positions can also be used to reduce the number of multiplications involved in the algorithm.

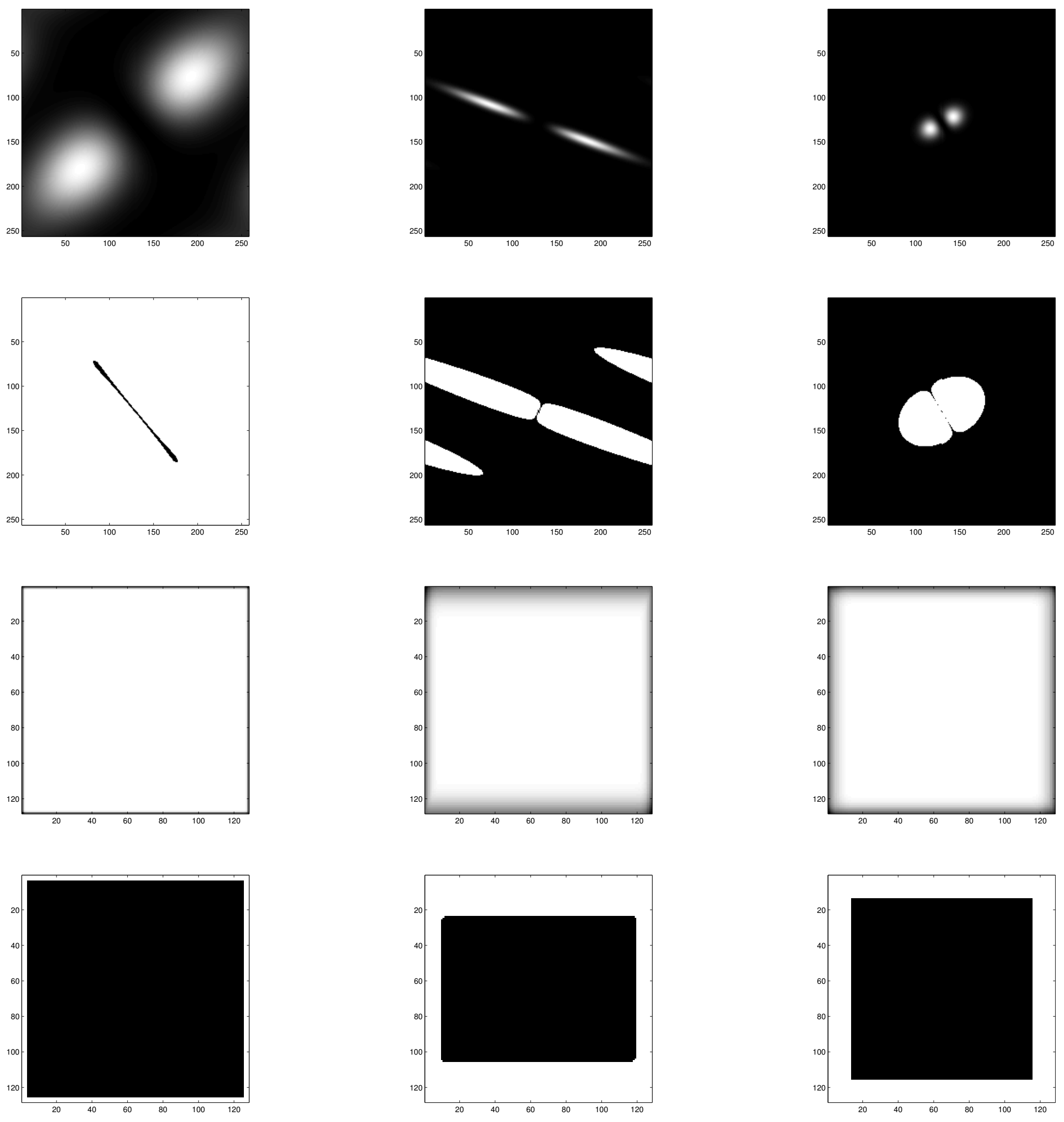


Fig. 9. First row: frequency modulus of three selected dictionary atoms. Second row: Respective support of the significant values that need to be stored in memory (values with modulus greater than 0.001). Third row: Spatial renormalization maps that correspond to the atoms of the first row. Fourth row: Binary mask that determines where values different from 1.0 with a significant difference (greater than 0.001) are located.

1) Memory Compression: Fig. 9 illustrates the kind of information contained in the maps that represent the frequency domain form of atoms (see the first row), and the normalizing factor maps (see the third row). In the second and fourth row of Fig. 9 can be respectively seen, for the Fourier domain atoms and normalization masks, the areas of each map where relevant information is cumulated. Concerning the Fourier domain version of the atoms, in this examples one sees in the second column, in white, the support where the modulus of the Fourier transform is greater than a given threshold (10^{-3} in this example). In the fourth column, the areas in white are the spatial locations where the renormalization factor differs from 1 by more than a certain threshold - 10^{-3} in the presented example - (i.e. factors smaller than 1 are used to re-normalize scalar products of atoms that significantly cross the image border). In both cases, only the original values corresponding to the white *footprints* are required to be stored, the rest can be approximated by zero for the Fourier atoms case and one for the re-normalizing mask case.

What makes feasible the compression in both cases is that zeros and ones cumulate in consecutive memory positions. Hence, nothing easier than using basic run-length coding [19] to store into memory more efficiently the default values (zero or one) for every kind of map. Of course, better techniques using quad-trees or prune-join trees (like in image compression techniques [20]) could be used, however this is out of the scope of this work.

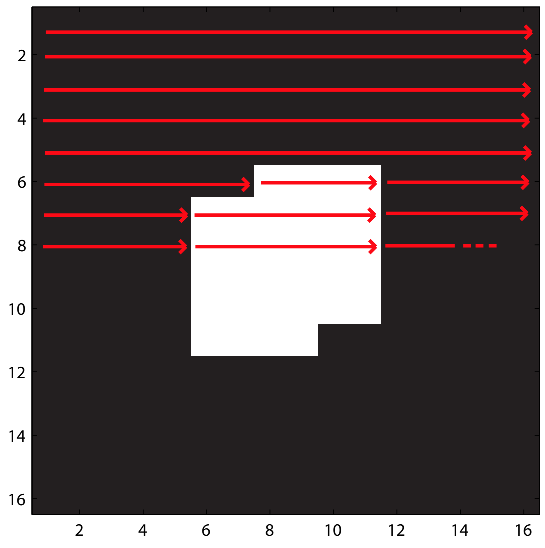


Fig. 10. Every Fourier domain atom is stored in a run-length fashion. All values considered insignificant are set to zero. All consecutive zeros (considering a line wise scanning of images) are efficiently stored using a single integer. Significant values are stored one by one together with an integer number that specifies how many significant values are consecutively aligned.

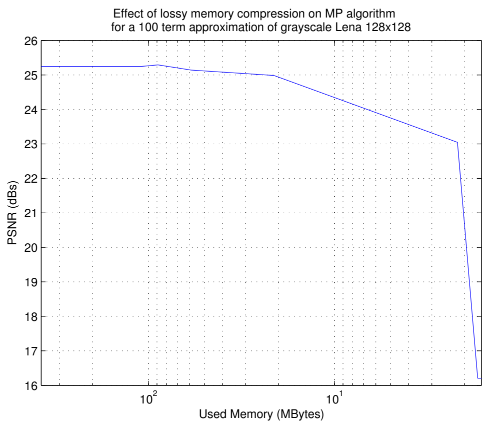
Fig. 10 depicts the raster scanning used for the storage of a simple example. In a top-down fashion, the signal is scanned line by line. Two lists of values are kept: one contains the list of elements contained in every run of significant pixels and the other contains how long is each run of values (significant and insignificant).

2) Examples: In the following, the effect of approximating the stored Fourier domain atoms and normalizing maps is analyzed. We compare in 11 and 12 how memory compression affects the quality of the approximation of lena (gray-scale 128x128 pixels) when using the FS algorithm. The degree of memory compression is varied by means of changing the threshold used to decide which region has significant values. The experiments are done by ranging the threshold from zero up to 100.

As can be appreciated in the figures, the dictionary can be approximated such that up to more than 75% of memory is saved, without any distortion increase. Fig. 12 shows this fact visually, a very high threshold has to be set before important distortions are perceived.

B. Steerability of Atoms and Complexity Benefits

Steerability is an interesting tool when dealing with orientable functions. This is based on the principle that certain classes of kernels can be tuned according to a set of geometric parameters, at the same time that can be decomposed in a linear combination of few functions sampled from that parameter space. For example, consider the kind of atoms used in this work, when their both axes have the same scaling, it is possible to generate all the possible orientations of these from just 3 basic functions with different orientations [21]. It is common to use the orientations of 0, $\frac{\pi}{3}$ and



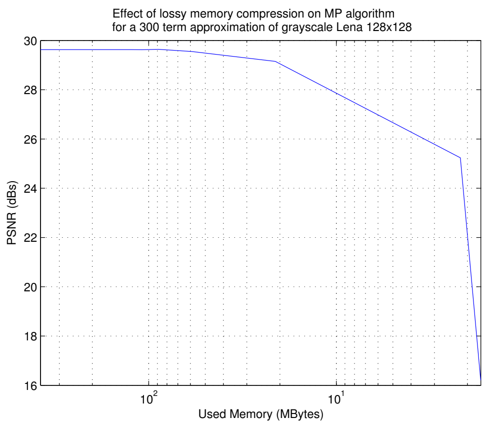


Fig. 11. Approximation PSNR vs memory used of the FFT based full search algorithm for lena 128x128. Up to 75% in memory savings can be achieved without loss in approximation convergence. In the left the approximation is done with 100 terms. In the right, 300 terms are used.

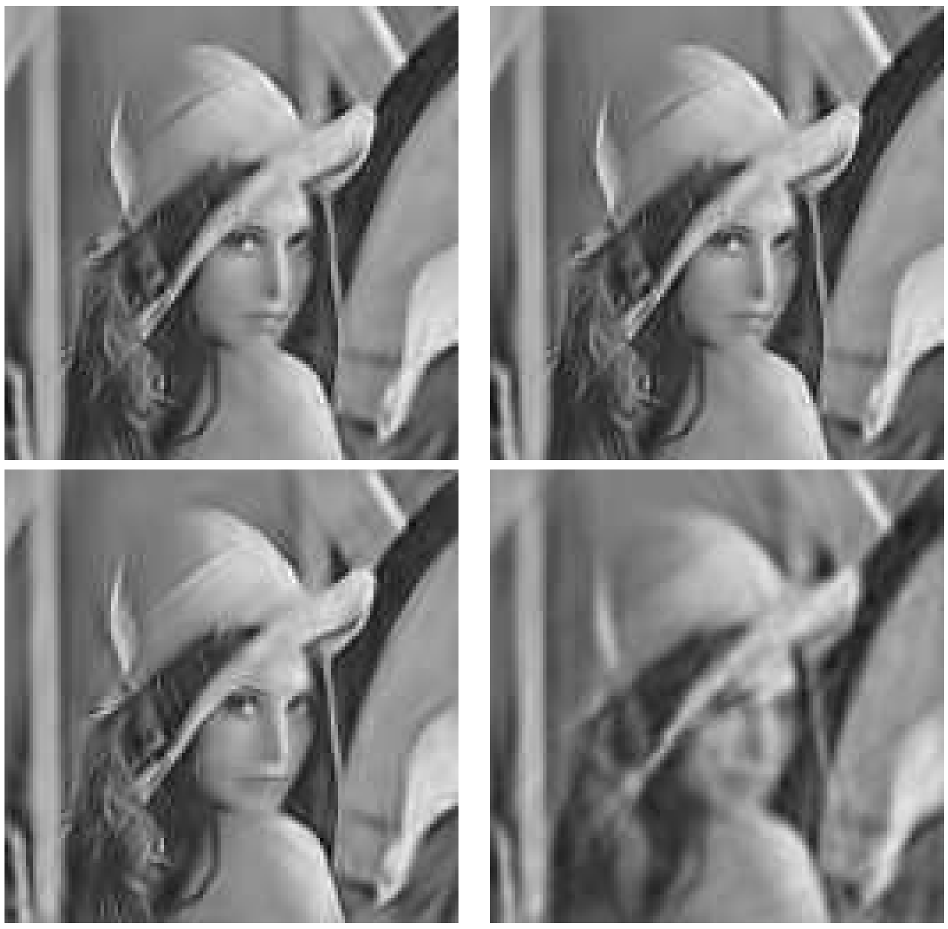


Fig. 12. Visual comparison of the different approximation of lena 128x128 for the amounts of memory of: (from left to right and then up/down) 377 MB (threshold 0), 122 MB (threshold 10^{-4}), 22 MB (threshold 1) and 2.2 MB (threshold 10).

$\frac{2\pi}{3}$ (see Fig. 13). However, the concept of steerability can be extended also to scaling transformations. In some cases, the steered functions are only approximations of the ideal ones. An extensive review of steerability can be found in [21], [22], [23], [24], [25]. In terms of computational advantages, the use of steerability would contribute significantly

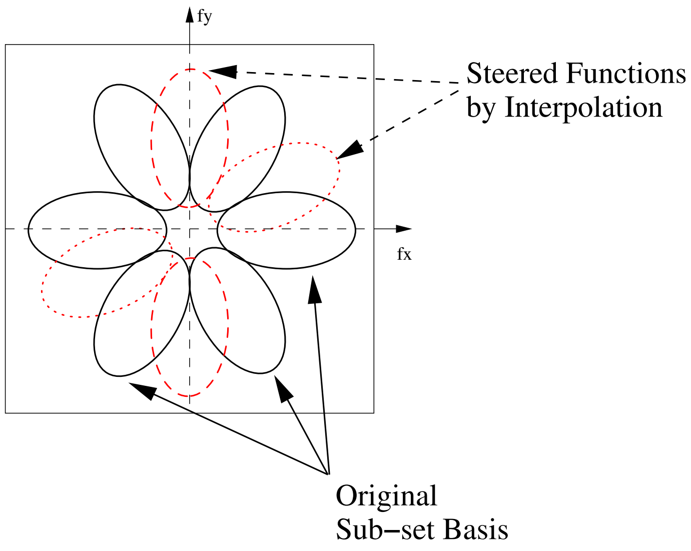


Fig. 13. In some cases, a whole set of functions from the dictionary can be generated from the linear combination of a subset of these. This is the case, for example, for isotropically scaled Gaussian second derivatives [21], only three real filters (drawn in continuous line) are needed to generate all the rest of orientations.

to reduce the number of necessary inverse Fourier transforms. To give an example, let us take again the isotropically scaled second derivative of a Gaussian. Consider we desire to compute the scalar products of all the translated versions of N_θ different orientations of our steerable kernel. For that purpose, if we store into memory a Fourier domain version of the function for each one of the N_θ orientations, as described in Sec. IV, a total number of:

$$O(N) \cdot N_\theta + O(N \log N) \cdot N_\theta \tag{21}$$

operations will be necessary (where N is the size of the data). In (21), the first term corresponds to the product in the transformed domain for filtering, and the second indicates the complexity associated to all the inverse Fourier transforms. If steerability is used instead, and all the needed orientations are obtained by linear combination of a basic set composed by three functions, the number of necessary operations turns into:

$$O(N) \cdot 3 + O(N \log N) \cdot 3 + O(N) \cdot 5 \cdot (N_\theta - 3), \tag{22}$$

where the first two terms correspond to the complexity required by the Fourier domain filtering step, and the last term corresponds to the steerability based computation of the remaining orientations. The 5 factor is due to the three products and two additions of the linear combination used to steer the filters. Fig. 14 illustrates the potential savings of using steerability in the computation of the scalar products for the simple example of isotropically scaled anisotropic second derivatives of a Gaussian. Even for small N, the use of steerability is computationally advantageous. Moreover, the larger N is, the more resources are spared. According to the particular example given by (21) and (22) for $N_\theta = 36$, when $N \to \infty$ the spare factor converges to 12. Moreover, in terms of memory usage, the amount of needed storage falls down of a factor $\frac{N_\theta}{3}$. This, reduces memory consumption as well as the time required to transfer the data from memory to the processor. In this section we state the advantages of introducing the steerability principle in the analysis step of each MP iteration. Nevertheless, its practical implementation and integration is out of the scope of this work and will be left for future work.

VI. Conclusions

In this work, a strategy has been presented to allows the implementation of full search matching pursuit for very dense dictionaries with spatially invariant atoms. This has revealed to be of capital importance for the efficient approximation of images if compared to suboptimal approaches like genetic algorithm based ones. In effect, the technique presented in here has allowed the investigation and implementation of a very interesting flexible low bit rate image coding scheme [26], [27], [28].

The use of the FFT allows to quickly calculate all scalar products of the displaced versions of a template thanks to the *Convolution Theorem*. Moreover, we have seen how some dictionaries allow the introduction of relevant enhancements to reduce complexity and memory usage. Spatio-frequential energy localization of dictionary functions is of key importance

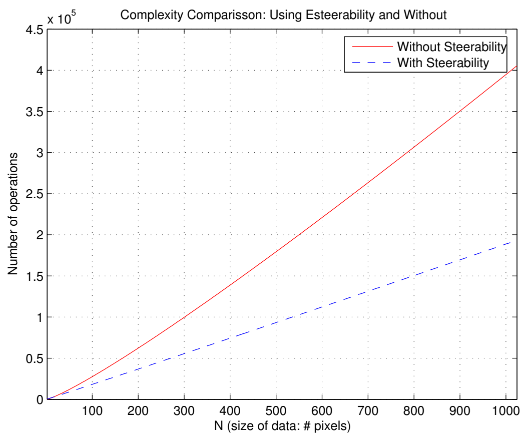


Fig. 14. Comparison of the growth of complexity with and without using steerability as a function of N. In this graph, we have assumed $N_\theta = 36$.

for that. Furthermore, for the particular dictionary used in this work, the use of steerability techniques seem very appropriate to further reduce complexity.

Very fast algorithms can be done for particular dictionaries. Structured dictionaries and those that can profit from the steerability properties may contribute to usable real-time signal decomposition techniques based on redundant dense dictionaries. More detailed research is left as future work.

References

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

[2] A. Cohen, I. Daubechies, O. Guleryuz, and M. Orchard, "On the importance of combining wavelet-based non-linear approximation in coding strategies," *IEEE Transactions on Information Theory*, vol. 48, no. 7, pp. 1895–1921, July 2002.

[3] M. N. Do, P. L. Dragotti, R. Shukla, and M. Vetterli, "On the compression of two dimensional piecewise smooth functions," in *IEEE International Conference on Image Processing (ICIP)*, 2001.

[4] B. Olshausen and D. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1 cells," 1997. [Online]. Available: citeseer.nj.nec.com/article/olshausen98sparse.html

[5] R. Gribonval and P. Vandergheynst, "On the exponential convergence of matching pursuits in quasi-incoherent dictionaries," IRISA, Rennes, France, Tech. Rep., 2004.

[6] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," Texas Institute of Computational and Applied Mathematics (TICAM), Tech. Rep. TICAM Report 03-04, February 2003.

[7] A. P. Korostelev and A. B. Tsybakov, *Minimax Theory of Image Reconstruction*, ser. Lecture Notes in Statistics. Springer-Verlag, 1993, vol. 82.

[8] D. Donoho, "Wedgelets: Nearly-minimax estimation of edges," Dept. Of Statistics, Standford University, Tech. Rep., 1997.

[9] R. M. Figueras i Ventura, L. Granai, and P. Vandergheynst, "R-D analysis of adaptive edge representations," in *Multimedia Signal Processing, 2002 IEEE Workshop on*, 2002, pp. 130–133.

[10] P. Vandergheynst and P. Frossard, "Efficient image representation by anisotropic refinement in matching pursuit," in *Proceedings of IEEE*, vol. 3. Salt Lake City UT: ICASSP, May 2001.

[11] R. M. Figueras i Ventura and P. Vandergheynst, "Matching pursuit with genetic algorithms," F Group, LTS, EPFL, Tech. Rep., 2001.

[12] ——, "Evolutive multiresoltion matching pursuit and its relation with the human visual system," in *EUSIPCO.*

[13] L. Davis, Ed., *Handbook of Genetic Algorithms.* Van Nostrand, 1991.

[14] S. Mallat and Zhang, "Matching pursuit with time-frequency dictionaries," in *IEEE Transactions on Signal Processing*, no. 12, December 1993.

[15] G. Davis, S. Mallat, and M. Avellaneda, "Adaptative greedy approximations," *Constructive Approximations*, 1997, springer-Verlag New York.

[16] V. N. Temlyakov, "Weak greedy algorithms," in *Adv. Comput. Math.*, vol. 12, 2000, pp. 213–227.

[17] P. Frossard, "Robust and multiresolution video delivery: From h.26x to matching pursuit based technologies," Ph.D. dissertation, EPFL, December 2000.

[18] A. Gilbert, S. Muthukrishnan, and M. J. Strauss, "Approximation of functions over redundant dictionaries using coherence," in *14th ACM-SIAM Symposium on Discrete Algorithms (SODA'03)*, January 2003.

[19] K. Sayood, *Introduction to Data Compression*, 2nd ed. Academic Press, 2000.

[20] R. Shukla, "Rate-distortion optimized geometrical image processing," Ph.D. dissertation, Ecole Polythecnique Fédérale de Lausanne, Lausanne, Switzerland, 2004.

[21] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, no. 9, pp. 891–906, September 1991.

[22] R. Manduchi, P. Perona, and D. Shy, "Efficient implementation of deformable filter banks," California Institute of Technology, http://www.vision.caltech.edu/manduchi/deformable.ps.Z, Tech. Rep. CNS-TR-97-04, 1997.

[23] P. Perona, "Steerable-scalable kernels for edge detection and junction analysis," *IVC*, vol. 10, pp. 663–672, 1992.

[24] Y. Hel-Or and P. C. Teo, "A common framework for steerability, motion estimation and invariant feature detection," Stanford University, Tech. Rep. CS-TN-96-28, January 1996.

[25] ——, "Canonical decomposition of steerable functions," *Journal of Mathematical Imaging and Vision*, vol. 9, no. 1, July 1998.

[26] R. M. Figueras i Ventura, P. Vandergheynst, and P. Frossard, "Low rate and scalable image coding using non-linear representations," to appear.

[27] P. Frossard, P. Vandergheynst, and R. M. Figueras i Ventura, “High flexibility scalable image coding,” in *VCIP*, Lugano, July 2003.
[28] R. M. Figueras i Ventura, P. Vandergheynst, P. Frossard, and A. Cavallaro, “Color image scalable coding with matching pursuit,” in *ICASSP*, vol. 3, Montreal, 2004, pp. 53–56.