

Matching Pursuit through Genetic Algorithms

Rosa M. Figueras i Ventura, Pierre Vandergheynst

Abstract

Matching Pursuit is a greedy algorithm that decomposes any signal into a linear expansion of waveforms taken from a redundant dictionary. Computing the projection of the signal on every element of the basis has a high computational cost. To reduce this computational cost, optimized computational error minimization methods have to be found. Genetic Algorithms have shown to be a good tool to this approach.

Keywords

Matching Pursuit, low bit-rate image coding, non-linear coding, redundant dictionaries, fast computation techniques, Genetic Algorithms.

I. INTRODUCTION

Even though quite a lot of research have been performed in the use of non-orthogonal basis for image coding, most of the techniques used at the moment rely on orthogonal basis. This is mainly because, as orthogonal basis have a single solution to a given problem, the computation of the coding is easier and a great amount of computation time is saved.

Orthogonal basis are normally formed by the tensor product of two mono-dimensional signals optimized for mono-dimensional space representation. But this way of creating a bidimensional orthogonal basis (which is good in terms of code optimization and computation acceleration) is not optimal when dealing images, because this base is unable to detect the contour continuity (see Fig. 1(a)). This incapability is what causes these basis to be inappropriate for very low bit-rate image coding.

Non-orthogonal basis are more adequated to detect the contour continuity and take advantage of that (see Fig. 1(b)) than tensor product basis. But as they present an infinite number of solutions to the same problem, conventional linear methods cannot be used, and some specific algorithms have to be created. Among these algorithms there is Matching Pursuit. Matching Pursuit [1] is a greedy algorithm that pretends to give a solution to a non-orthogonal problem by choosing at each iteration the basis component that takes more energy to the signal to code.

Matching Pursuit with anisotropic refinement atoms has been shown to be quite good in terms of image and video decomposition [2], [3], [4]. But MP has a big inconvenient: an extremely high computational time. In fact, when having a relatively large dictionary, it turns to be almost impossible to perform the full search and code the image in a finite time... That is the reason why some faster approximation techniques have to be found.

Among fast approximation techniques there is Genetic Algorithms. These allow to find a sub-optimal solution to the problem, and make computation time decrease considerably.

II. MATCHING PURSUIT

Tensor product basis have been shown not to be adapted to bidimensional signal coding due to the fact that they are not able to detect contour continuity. Working with non-orthogonal basis may give some better approximation rate, because they allow to detect continuities of contours and specific image characteristics such as scaling and rotations, which are normally not easily represented by orthogonal functions (see Fig. 1). In addition, its redundancy gives, when dealing with very low bit-rates, a higher approximation rate than any orthogonal basis. But non-orthogonal basis have some disadvantages. The first, and most important, is that they offer infinite solutions to the same problem so standard linear mathematical tools are not useful when dealing with them. In the last years, some algorithms to give non-linear solutions to a problem have appeared (such as fractal coding [5]), and among them Matching Pursuit.

The Matching Pursuit technique was first introduced by Mallat and Zhang [1], for mono-dimensional time-frequency signals. This method produces suboptimal function expansions by iteratively choosing the waveforms from a general dictionary (typically a rich collection of potential atoms in a Hilbert space) that best match the function's structures. The choice of the functions is performed through a progressive refinement of the signal approximation with an iterative procedure [6]. The Matching Pursuit method has already found applications in medicine [7] and in image [2], [8] and video coding [4] (though in video coding it is usually used to code the motion estimation error [9]). Most of the

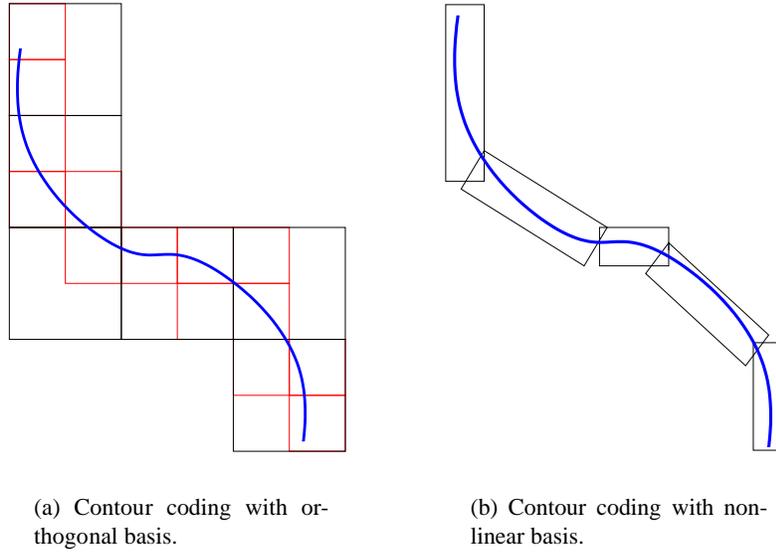


Fig. 1. Tensor product inadaptability to contour continuity is represented in figure (a). In (b) the adaptability of non-linear basis with anisotropic scaling and rotations is shown.

applications of Matching Pursuit use Gabor functions or symmetric dictionaries. Some use also orthogonalized MP [6] to be able to have a zero estimation error, though this is not very used because it increases too much the computational cost.

Matching Pursuit [10], [1] is a **greedy algorithm** [6] that decomposes any signal into a linear expansion of waveforms that are selected from a **redundant dictionary** \mathcal{D} . 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 (in terms of energy) at each step. Even though the expansion is linear, it gives a non-linear signal decomposition, because the set of functions selected depends on the signal to be coded.

Let $\mathcal{D} = \{g_\gamma\}_{\gamma \in \Gamma}$ be a dictionary of $P > N \times M$ vectors, having unit norm. This dictionary includes $N \times M$ linearly independent vectors that define a basis of the space $\mathbb{C}^{N \times M}$ of signals of size $N \times M$. A matching pursuit begins by projecting f on a vector $g_{\gamma_0} \in \mathcal{D}$ and computing the residue Rf [10], [1]:

$$f = \langle f, g_{\gamma_0} \rangle g_{\gamma_0} + Rf, \quad (1)$$

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

$$\|f\|^2 = |\langle f, g_{\gamma_0} \rangle|^2 + \|Rf\|^2. \quad (2)$$

As the term that must be minimized is $\|Rf\|^2 = \|f\|^2 - |\langle f, g_{\gamma_0} \rangle|^2$, the $g_{\gamma_0} \in \mathcal{D}$ to choose is the one that maximizes $|\langle f, g_{\gamma_0} \rangle|$. In some cases it is not computationally efficient to find the optimal solution, and a suboptimal solution is computed instead:

$$|\langle f, g_{\gamma_0} \rangle| \geq \alpha \sup_{\gamma \in \Gamma} |\langle f, g_\gamma \rangle|, \quad (3)$$

where $\alpha \in (0, 1]$ is an optimality factor which is 1 when the optimal solution has been chosen. In the present case the suboptimality factor α will be given by the Genetic Algorithm and will depend on the GA parameters, such as number of individuals in the population or of generations before choosing the fittest individual, as well as on the mutation probability (see section V).

A Matching Pursuit is an iterative algorithm that subdecomposes the residue Rf by projecting it on a vector of \mathcal{D} that matches Rf (almost) at best. If we consider $R^0 f = f$ and we suppose that the n^{th} order residue $R^n f$ ($n \geq 0$) has been computed, the next iteration will chose $g_{\gamma_n} \in \mathcal{D}$ such that:

$$|\langle R^n f, g_{\gamma_n} \rangle| \geq \alpha \sup_{\gamma \in \Gamma} |\langle R^n f, g_\gamma \rangle|. \quad (4)$$

With this choice $R^n f$ is projected on g_{γ_n} and decomposed as follows:

$$R^n f = \langle R^n f, g_{\gamma_n} \rangle g_{\gamma_n} + R^{n+1} f, \quad (5)$$

$R^{n+1} f$ and g_{γ_n} are orthogonal, so the quadratic module of the previous equation is:

$$\|R^n f\|^2 = |\langle R^n f, g_{\gamma_n} \rangle|^2 + \|R^{n+1} f\|^2. \quad (6)$$

From (5), we can see that the decomposition of f is given by:

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

and with the same principle we can also deduce from (6) that the module of the signal f is:

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

where $\|R^N f\|$ converges exponentially to 0 when N tends to infinity (see Appendix VII for a proof).

The convergence rate λ given by (31) in Appendix VII decreases when the size of the signal space increases. On the other hand even in finite spaces an infinite number of iterations is needed to completely reduce the residue (only with Orthogonalized Matching Pursuit [10], [6] the residue could be reduced to 0 in a finite number of iterations). In most signal processing applications having a non-zero residual is not relevant, due to the fact that when the image distortion is under the visible distortion threshold it does not matter. In addition, some quality loss is allowed when targeting very low bit rate applications.

The decreasing of the error in Matching Pursuit is highly dependent on the dictionary. In this case the dictionary used is with Anisotropic Refinement. The basic function is a Gaussian in one axis and the second derivative of the Gaussian in the other axis [3]:

$$g_\gamma(x, y) = (2 - 4x^2)e^{-\frac{1}{4}(x^2+y^2)}. \quad (9)$$

The dictionary is formed by the above basis function with rotations, anisotropic scaling and translation. These transformations allow the basis functions to detect the object contour continuity quite efficiently. Anisotropy introduces an extra redundancy to the dictionary, and an extra parameter to code, but it is worth because the addition of this parameter implies a great increase in efficiency (see Fig. 3). The set of functions formed by expression (9) with anisotropic scaling, rotation and translation forms an over-complete basis.

As the search of the optimal function means computing a great amount of scalar products between images, MP has a very high computational cost. To speed the coding, a multi-resolution scheme has been chosen. In this scheme, the image is downsampled by two several times. The MP algorithm is first applied to the smallest image and when the desired number of coefficients in the lowest resolution layer has been reached, an up-sampling of the recomposition has to be performed. The subtraction of this upsampled recomposition to the next resolution level image is performed, and the MP is applied to this residual (see scheme in Fig. 2). This can be done with any number of layers, but a compromise has to be found to optimally choose the number of coefficients at each layer and the number of layers. One scheme that has demonstrated to give quite good results is working with three resolution layers and in every bigger layer double number of coefficients than in the previous resolution level.

The use of a redundant basis seems interesting to an image representation point of view, but it represents a heavy computational cost. In fact, when dealing with large dictionaries, the fact of computing a scalar product of every element of the dictionary and the signal to represent and take the atom with a larger projection energy becomes almost impossible. In this scope, the use of efficient approximation tools, as Genetic Algorithms, is needed.

Genetic Algorithms do not give the optimal solution, but an approximation. Though, this fact does not represent a problem when dealing with MP decomposition. Having a suboptimal solution will, of course, slightly decrease the final coding quality, but thanks to redundancy this will not represent a significant quality loss.

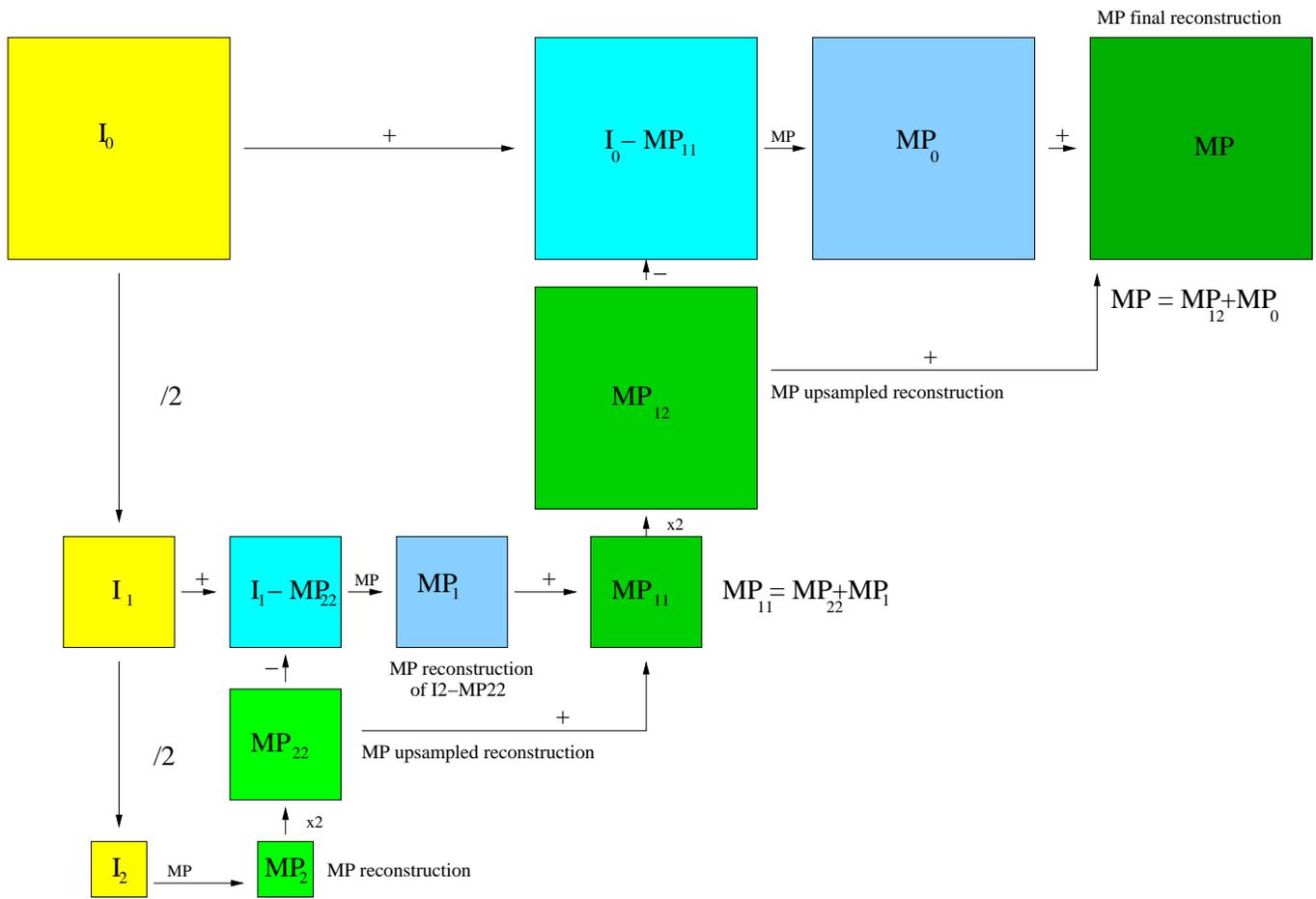


Fig. 2. MP multiresolution scheme.



(a) MP with Gabor atoms.



(b) MP with AR atoms.

Fig. 3. Anisotropic refinement atoms versus Isotropic Gabor atoms. The fact of adding anisotropy gives better contour resolution. Figure (a) has been coded with Gabor atoms with isotropic scaling, rotation and translation, and figure (b) with the anisotropic refinement atoms. Both have 500 coefficients. It can be seen that adding anisotropy brings, with a fixed number of decomposition terms, a higher contour definition.

Matching Pursuit has two kinds of properties: properties that depend on the dictionary (so the MP decomposition will only have them if the chosen dictionary has them as well) and the ones that are independent on the dictionary (so any MP decomposition has them no matter which set of functions has been used to perform the decomposition).

The main properties derived directly from the Matching Pursuit algorithm (so, independent from the dictionary) are:

- *Energy conservation:* In Matching Pursuit, when dealing with an infinite decomposition series, the energy in the transformed domain and the energy in the space domain is the same, as can be deduced from equation (8). As

$$\lim_{M \rightarrow \infty} R^M f = 0 \quad (10)$$

(because of the exponential decreasing of the coefficients and the error), when $M \rightarrow \infty$ (8) turns to:

$$\|f\|^2 = \sum_{m=0}^{\infty} |\langle R^m f, g_{\gamma_m} \rangle|^2, \quad (11)$$

which mimics Parseval's equality for Fourier series.

- *Invertible:* A complete Pursuit recovers a perfect version of the image:

$$f = \sum_{m=0}^{\infty} \langle R^m f, g_{\gamma_m} \rangle g_{\gamma_m}. \quad (12)$$

Thus the image f may be reconstructed from its MP coefficients, but if the decomposition is finite the reconstruction of the coded signal will not be perfect, there will be a reconstruction error given by $R^M f$, where M is the number of coefficients used by the decomposition.

- *Non-linearity:* The fact of having a non-linear dictionary gives to Matching Pursuit the two following characteristics, which are very appreciated when performing coding:

- *Robustness to quantization:* Robustness to quantization comes from the fact the decomposition is overcomplete. Because of overcompleteness, the transformed domain space has dimension P , higher than the dimension $N \times M$ of the original signal. When quantizing the transformed domain, the error quantization is spread allover the P dimensions of the transformed domain. But when applying the inverse transform, some of the information in the transformed domain (the one that belongs to the dimensions that do not exist in the original space) is lost, and so part of the quantization error performed will not affect at all to the quality of the decoded signal.

- *Exponential decrease of the error:* This implies a great decrease in the first coefficients and so a fast initial approximation. After a certain number of coefficients, MP error decrease is no too fast, and a change of dictionary or of coding method may be worth. This exponential decreasing of the error will only happen if the signal has finite dimension, because in infinite dimension the condition of the unit sphere being dense in the working space used in the demonstration in Appendix VII becomes false.

MP may have other properties, depending on the properties of the set of functions in the dictionary used to decompose the signal. In many image processing application, it may be interesting to have certain covariance to image transformations, as translation, rotation and dilation (for example, in pattern recognition). MP will have this covariance if the dictionary used to decompose a signal has also them:

- *Translation Invariance:* A dictionary \mathcal{D} is translation invariant if for any $g_{\gamma}[\vec{n}] \in \mathcal{D}$ and any $\vec{p} = [p_x, p_y] \in [0..N-1, 0..N-1]$ then $g_{\gamma}[\vec{n} - \vec{p}] \in \mathcal{D}$. If Matching Pursuit is computed in a translation invariant dictionary, then its decomposition will be translation invariant. Given the decomposition of f in \mathcal{D} ,

$$f[\vec{n}] = \sum_{m=0}^{M-1} \langle R^m f, g_{\gamma_m} \rangle g_{\gamma_m}[\vec{n}] + R^M f[\vec{n}], \quad (13)$$

it is easy to verify [6] that the matching pursuit of $f_{\vec{p}}[\vec{n}] = f[\vec{n} - \vec{p}]$ selects a translation by \vec{p} of the same vectors g_{γ_m} with the same decomposition coefficients:

$$f_{\vec{p}}[\vec{n}] = \sum_{m=0}^{M-1} \langle R^m f, g_{\gamma_m} \rangle g_{\gamma_m}[\vec{n} - \vec{p}] + R^M f_{\vec{p}}[\vec{n}]. \quad (14)$$

- *Rotation invariance:* By analogy, a rotation invariant Matching Pursuit can be obtained by using a rotation invariant dictionary \mathcal{D} . A dictionary is rotation invariant if for any $g_\gamma[\vec{n}] \in \mathcal{D}$ and any $\theta \in [0, 2\pi]$ then $g_\gamma[r_\theta \cdot \vec{n}] \in \mathcal{D}$, where r_θ is the rotation operator given by the matrix:

$$r_\theta = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}. \quad (15)$$

Given the decomposition of f in \mathcal{D} ,

$$f[\vec{n}] = \sum_{m=0}^{M-1} \langle R^m f, g_{\gamma_m} \rangle g_{\gamma_m}[\vec{n}] + R^M f[\vec{n}], \quad (16)$$

one can verify that the matching pursuit of $f_\theta[\vec{n}] = f[r_\theta \cdot \vec{n}]$ selects a rotation by θ of the same vectors g_{γ_m} with the same decomposition coefficients:

$$f_\theta[\vec{n}] = \sum_{m=0}^{M-1} \langle R^m f, g_{\gamma_m} \rangle g_{\gamma_m}[r_\theta \cdot \vec{n}] + R^M f_\theta[\vec{n}]. \quad (17)$$

This makes Matching Pursuit a useful technique to rotate images, because the only extra calculation to be done is to modify the index of the reconstruction atoms when computing the coded image, instead of applying the rotation matrix to every pixel of the image.

- *Dilation invariance:* As in the previous two cases, Matching Pursuit is dilation invariant if the dictionary of functions used by the pursuit is dilation invariant. A dictionary \mathcal{D} is dilation invariant when for any $g_\gamma[\vec{n}] \in \mathcal{D}$ and any $s \in [0, s_{max}]$ then $g_\gamma[\frac{\vec{n}}{s}] \in \mathcal{D}$. In this case the matching pursuit of $f_s[\vec{n}] = f[\frac{\vec{n}}{s}]$ will select a dilation by s of the same vectors g_{γ_m} with the same decomposition coefficients:

$$f_s[\vec{n}] = \sum_{m=0}^{M-1} \langle R^m f, g_{\gamma_m} \rangle g_{\gamma_m} \left[\frac{\vec{n}}{s} \right] + R^M f_s[\vec{n}]. \quad (18)$$

The Anisotropic Refinement Atoms defined in the scope of this paper are isotropically dilation invariant: they are covariant to dilations only when they are applied to the whole \vec{n} . If one applies a different scaling for every component of the vector, the property of dilation invariance will be lost.

Joining rotation, translation and dilation invariance gives a useful tool for pattern recognition, because the coefficients do not depend on the position, the orientation or the size of the object. Certain coefficients and certain relations of the atom parameters would mean the presence of a concrete pattern in the analysed image.

IV. SKIPPING MULTIREOLUTION IN THE DECODER

Dilation invariance gives an easy way of scaling an image: the only thing that has to be done is to modify the scaling parameter (equally for x and y axis) when reconstructing the image and a larger or smaller image (larger if the coefficient that multiplies the scaling factor is greater than one and smaller in the contrary case) will be obtained. This can be used in the Multiresolution Matching Pursuit coder to recombine the image without the need of upsampling and interpolating the previous layer in the decoder. Just taking the previous layer atom parameters and multiplying the scale factor by two (in case the downsampling and upsampling was done by two) and adapting the translation parameters to the new image size as well (which in this case is multiply them by two as well) a bigger image can be reconstructed. This skips quite a lot of operations in the decoder, and makes a Matching Pursuit decomposition quite good for asymmetric platforms (simple decoder, while quite complicated coder). Result comparison for multiresolution decoding with upsampling and with atom adaptation can be seen in Fig. 4, which shows that the fact of skipping the multiresolution in the decoder and applying the dictionary translation invariance does not imply a quality loss.

To be able to reconstruct the image without needing a multiresolution decoder, adaptation of the parameters of the previous level to the following level size must be performed. This adaptation is quite easy. Having a decomposition term defined by:

$$C, \quad \gamma = (p_x, p_y, s_x, s_y, \theta) \quad (19)$$



(a) Decoded applying the multiresolution



(b) Decoded adapting the scaling factor to the final size

Fig. 4. Comparison of two types of MP decoding. (a) there is a normally decoded image, following exactly the coder inverse scheme. (b) has been decoded without the upsampling scheme, but with adaptation of the atom parameters of each multiresolution level to the final image size.

where C is the coefficient and γ is the parameter that defines an atom, the new term for the double-sized image will be:

$$\hat{C} = 2 \times C, \quad \hat{\gamma} = (2 \times p_x, 2 \times p_y, 2 \times s_x, 2 \times s_y, \theta) \quad (20)$$

The coefficient has to be multiplied by two as well to keep the energy in the larger scale. Positions have to be multiplied by two to maintain the atom relative position in the larger scale and dilation factors to double the image size.

V. GENETIC ALGORITHM

A. Introduction to Genetic Algorithms

Genetic Algorithms (GA) is a tool for searching and optimizing methodology (see [11], [12], [13], [14], [15], [16], [17]) that has found many useful applications in both the scientific and engineering arenas [18]. The GA (basis first proposed by Holland [19]) work on the Darwinian principle of natural selection: the “Survival of the fittest” [20]. This biologically oriented method to search the best solution does not give an optimal result, but a suboptimal instead, that tends to the best possible one when the number of iterations used in the search tends to infinity. The final objective is to find a compromise between speed and accuracy and try to maximize this relation.

B. Biological inspiration

The living beings found in nature have often been source of inspiration for scientists and artists: cathedrals that are like trees, helicopters that fly like colibries... In computers it is also possible to learn from nature: when a choice must be done, the most ancient method known is *natural selection*, an when there are too much parameters to choose, this method becomes very useful, because a great amount of computation time can be saved with a low quality cost. Genetic algorithms pretend to imitate the natural selection, by considering the parameters of the items to choose as genes, and the different items, elements of a population. To understand this better some previous knowledge of the genes and the natural selection are needed. This information is basically given in the next sections, though numerous information sources [21] exist.

Reproduction has two main phases: flow of genetic information and genetic information recombination. In some cases mutations, which are not meant to happen but which are crucial for evolution and live conservation, can appear in these processes or during the normal cell life. Through these processes the genetic information is transmitted from one generation to the next.

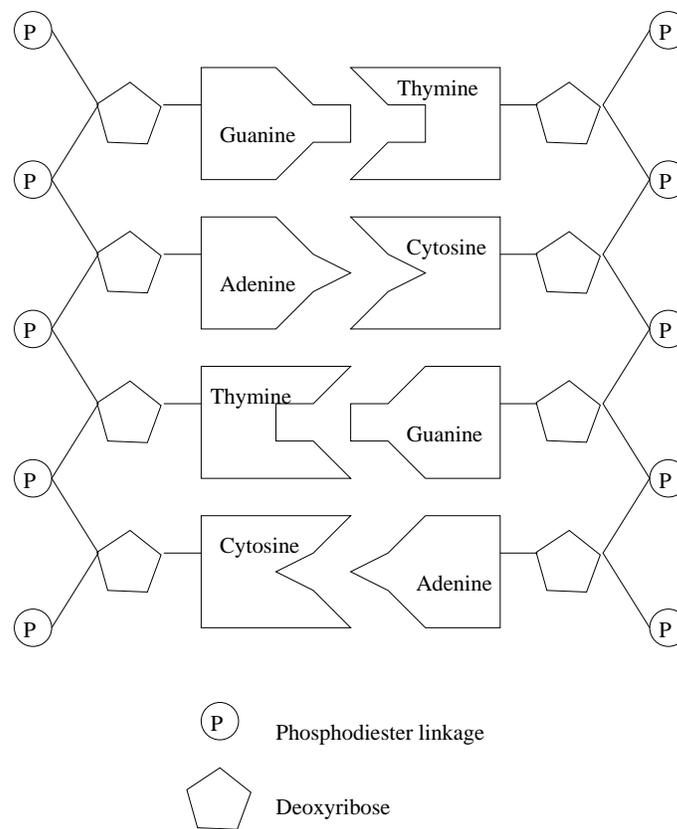


Fig. 5. DNA structure. As can be seen in the figure above, Guanine can only be combined with Thymine and Adenine with Cytosine.

B.1 Flow of Genetic Information

Flow of genetic information is the process of making the genetic information of an individual pass to the next generation during the cellular reproduction. Cellular reproduction concerning the flow of genetic information is performed according to three main steps: replication, transcription and translation.

Replication: Genetic information is preserved by DNA replication. During this process the two parent strands separate, and each serves as a template of the synthesis of a new complementary strand. Each offspring cell inherits one strand of the parental duplex: this pattern of DNA replication is described as semi-conservative.

Transcription: The first step in the communication of genetic information from the nucleus of the cell to the cytoplasm is the transcription of DNA into mRNA. During this process, the DNA duplex unwinds and one of the strands serves as a template for the synthesis of a complementary RNA strand, mRNA. RNA remains single stranded and functions as the vehicle for translating nucleic acid into protein sequence.

Translation: In the process of translation, the genetic message coded in mRNA is translated in the ribosomes into a protein with specific sequence of amino acids. Many proteins consist of multiple polypeptide chains. The formulation of polypeptide involves two different types of RNA, mRNA and tRNA, which play important roles in gene translation. Codons are carried by the intermediary formation of mRNA while tRNA, working as an adapter molecule, recognizes codons and inserts amino acids into their appropriate sequential positions in the polypeptide (a product of joining amino acids).

The three steps above imply a flow of genetic information that keeps and relies the genetic information of a cell, but it does not interact with any other cell. When an exchange of information between two cells is desired, recombination must be present.

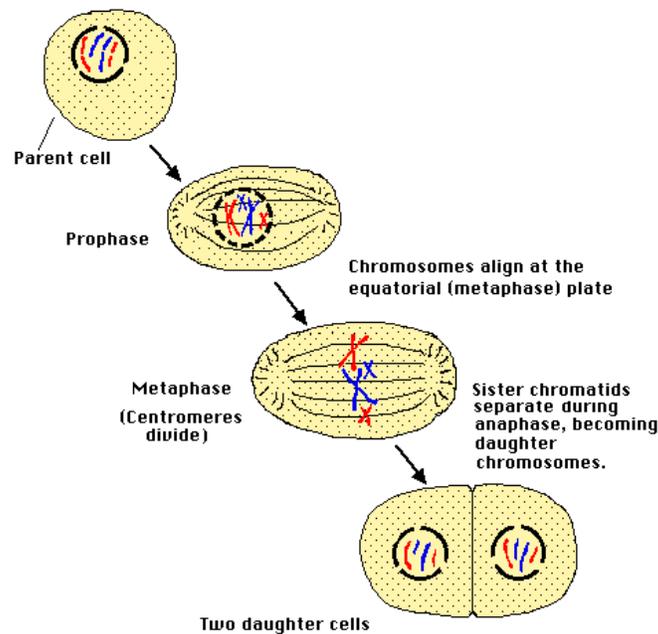


Fig. 6. Mitosis. In mitosis, from one original cell appear two identical cells, each one with the same genetic information than the original

B.2 Recombination

Recombination is the process of the exchange of genetic information between two strands of DNA, and it is based in the ability of the “invading” strand to pair with its complement. Recombination is performed between two non-reciprocal strands which mix one with the other. The final result is a new sequence. The recombination process has several steps:

1. DNA with strand break is aligned with a second homologous DNA.
2. Reciprocal strand switch produces a Holliday intermediate.
3. The crossover point moves by branch migration and strand breaks are repaired.
4. The Holliday intermediate can be cleaved (or resolved) in two ways, producing two possible sets of products.

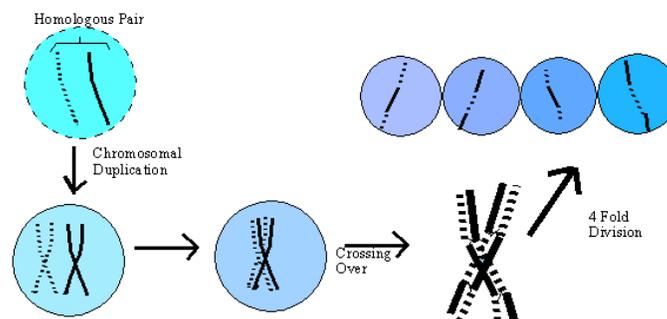


Fig. 7. Recombination of genetic information. We can see in this figure the effect of recombination of genetic information: from 2 different individuals, we get 4 different individuals none of which is equal to the previous two

With this process we have two final different strands, none of which is equal to the original one.

B.3 Mutations

Although DNA is a polymer with relatively high stability, some spontaneous relations cause a change in the DNA sequence, a *mutation*, that produces an inheritable change in the phenotype. Mutations can happen during the live of

the individual due to some external factor (for example radiation) or during the reproduction process. Mutations can produce benefit to the cell, but they usually produce damage. When the mutation is beneficial, by natural selection it is probable to be passed to the next generation, and the number of individuals with the mutation will increase, while the others will decrease and, in some cases, disappear. When it causes damage it is probable that it does not even reach the next generation, or simply the mutating individuals will not survive and will disappear.

C. General structure of a genetic algorithm

GA are inspired by the mechanism of natural selection where stronger individuals are likely to be the winners in a competing environment, and use an analogy of such natural evolution. With GA we pretend to find an optimal solution through the genetic evolution method, solution which is represented by the final winner of the genetic game, (in fact, due to the randomness of the algorithm, the final solution will not be optimal, but suboptimal).

The GA algorithm is first initialized with a set of parameters that define a given number of potential solutions. These potential solutions are considered as individuals, and the parameters that define these solutions are treated as genes. The “strength” of each chromosome is reflected by a positive value narrowly related with the solution objective: the fitness value.

With an initial population (generally randomly selected) the algorithm produces genetic evolution (via mutations and crossover) and in every iteration the fittest chromosome survives, and the following fittest are crossed and have descendants. Also some mutations of the fittest are placed in the next generation. This process is repeated, and in each iteration the solution improves and gets closer to the best possible result.

VI. MATCHING PURSUIT WITH GENETIC ALGORITHM

A. Concrete algorithm specification

The algorithm used here is based on an algorithm used in [13] to optimize air-injected hydrocyclone. The basics of this algorithm is to choose randomly an initial population, estimate and make the fittest pass untouched in the next generation, while the other individuals fight by pairs, and the fittest of each pair goes to the recombination pool. Then each of this individuals is randomly recombined with the others and the “sons” pass to the next generation with the fittest. This process is repeated until stability is reached, when only the fittest is kept and the other atoms randomly reinitialized.

The algorithm implemented here has some modifications with respect to the one explained in [13] to maximally adapt it for the scope of this application:

- Stability is reached not by bit to bit comparison, but by making the decimal difference between the atoms and the best current solution. This is because in the specific definition of dictionary atoms, when individuals are really close is when their parameters have close definition values, and a bit to bit comparison loose this sense of proximity.
- Mutations are not bit to bit mutations, but are also implemented by randomly adding a decimal number that must be comprised into the variance intervals chosen by the user depending on their interests (big variance implies more mobility, but if the best solution is near the actual one, it may not appear in the next generation, while small variance gives more accuracy when the solution is near but less mobility, so less probabilities to reach the surroundings of the optimal result).
- All the parameters are discrete to be able to have a discrete dictionary of functions and an easiest way to code them.

The algorithm follows the steps described below, which try to imitate the genetic evolution system:

1. Randomly generate N strings for the initial population (N is a small odd number).
2. Evaluate the fitness of each string, select the best solution based on fitness, place it in position N and send it unchanged into the next generation.
3. Force strings that are adjacent to each other in the population to compete directly with each other for the right to survive. Similarly, the fittest string between strings n and $n + 1$ is placed in the mating pool (where n goes from 1 to $N - 2$). When all the local competitors are held, we have $\frac{N-1}{2}$ strings in the mating pool, which are crossed with a probability of 1.0 and placed in the next generation. The local competitions enhance the Darwinian “survival-of-the-fittest” aspect of the GA. They help ensure that the best solutions thrive in the population.

4. At this point there are $\frac{N-1}{2} + 1$ strings in the next generation and the $\frac{N-1}{2}$ remaining are generated in a random fashion, by mutating the best string with a probability of p'_{mutate} that is related with the variance interval introduced by the user.
5. Return to step 2 until convergence is achieved (that is when none of the strings in the population differs from the population's current best solution by more than four units in every gene) or until the maximum number of iterations fixed by the user is reached. If the maximum number of iteration is reached, the algorithm finishes here and returns the fittest individual in the actual generation. If stability has been reached, the sixth step is executed.
6. Take the best solution and place it in a second "initial generation", generate the other $N - 1$ strings in this second initial generation at random, and begin the cycle again until the maximum number of generations allowed is reached. This is necessary to have an evolutionary population, and so a population that is able of adapting to the environment. When a population has reached stability, it has no possibility of adapting to the environment. Stability may be reached because the current solution is the best possible solution (possible, but not probable) or simply because in the population there were only weak individuals. Reinitializing the population makes it more dynamic, and so more adaptive. So reinitializing the population, when keeping the stronger individual, will give more possibilities to reach a better solution by giving more dynamism to the population.

A general block diagram of this algorithm is given in Fig. 8. Looking at the diagram it is possible to see that the algorithm has two basic branches: one based in mutations of the strongest, to check if it can be mutated to a stronger individual, and a second one based on the cross of the next stronger individuals, to check if any descendant of them will have better performance than any of the actual individuals.

B. Practical advice

In order to program a genetic algorithm, one needs to know a little bit further than the genetic point of view: there are some other details, not related with the genetics, but that are also important when beginning to program a GA.

The first important point is to fix the bounds within which a genuine solution lies. The tighter these bounds are, the faster the algorithm will reach a better solution. The ideal case is when these bounds are previously known, otherwise it is necessary to compute them or estimate them.

The second parameter to fix is the initial population. If a sufficiently high number of generations is chosen, the initial population will not affect the final result. This condition is very important, because the initial conditions are usually a problem. As in the genetic algorithm the initial condition is not really important, the best choice is to randomly initialize the population.

The third characteristic is the range of the mutations. Mutations are crucial to have a faster approximation of the solution. In this case the mutations are implemented by adding a random number comprised in the variance intervals chosen:

$$\overrightarrow{mutation} \in [-\overrightarrow{var}, \overrightarrow{var}]. \quad (21)$$

For large data intervals, \overrightarrow{var} is advised to be a 10% of the whole data interval. If the data interval is so little that the 10% then a variance of at least one unit (and better if it is two, even) has to be taken otherwise the mutation probability would be zero for that parameter.

Another characteristic that is necessary to comment is the crossing of chromosomes. This is done to check if any mix of the existent chromosomes will lead the algorithm to reach the optimal solution. To do this mix, for each gene a random number decides whether this gene will be kept or not, and if it will not be kept, from which individual it will be taken. This makes the population more dynamic, and as only the genes from the stronger individuals are kept, it should make the population fitter than before.

In the case of Matching Pursuit the genetic algorithm must be used to find, at each iteration, the atom that better represents the image. As already said, the GA does not find an optimal solution. The sub-optimality depends on the number of individuals we have in our population and on the number of generations allowed before taking the fittest individual in the population as the final result. As the number of atoms of the dictionary depends on the size of the image and on the parameter NN (that divides the exponent of the scaling factor), the optimality factor will decrease as NN or the size of the image increase. This shows that the larger the dictionary is, the more generations will be needed to reach the desired optimality factor.

Using the GA described in this section, by imposing 21 individuals in the population and 50 generations before choosing the best individual in the present population as the solution it gives a α factor of 0.6741 for a 32×32 image. If the image is 64×64 , the α is then 0.5254. For the application of image coding these values have shown to be acceptable.

VII. CONCLUSIONS

Non-linear coding is interesting when dealing with very low bit rate applications, but standard mathematical tools are not adapted to it. Matching Pursuit has shown to be a useful tool when wanting to code signals with redundant dictionaries. It has, though, a really hard computational cost which makes it inappropriate for most applications. To be

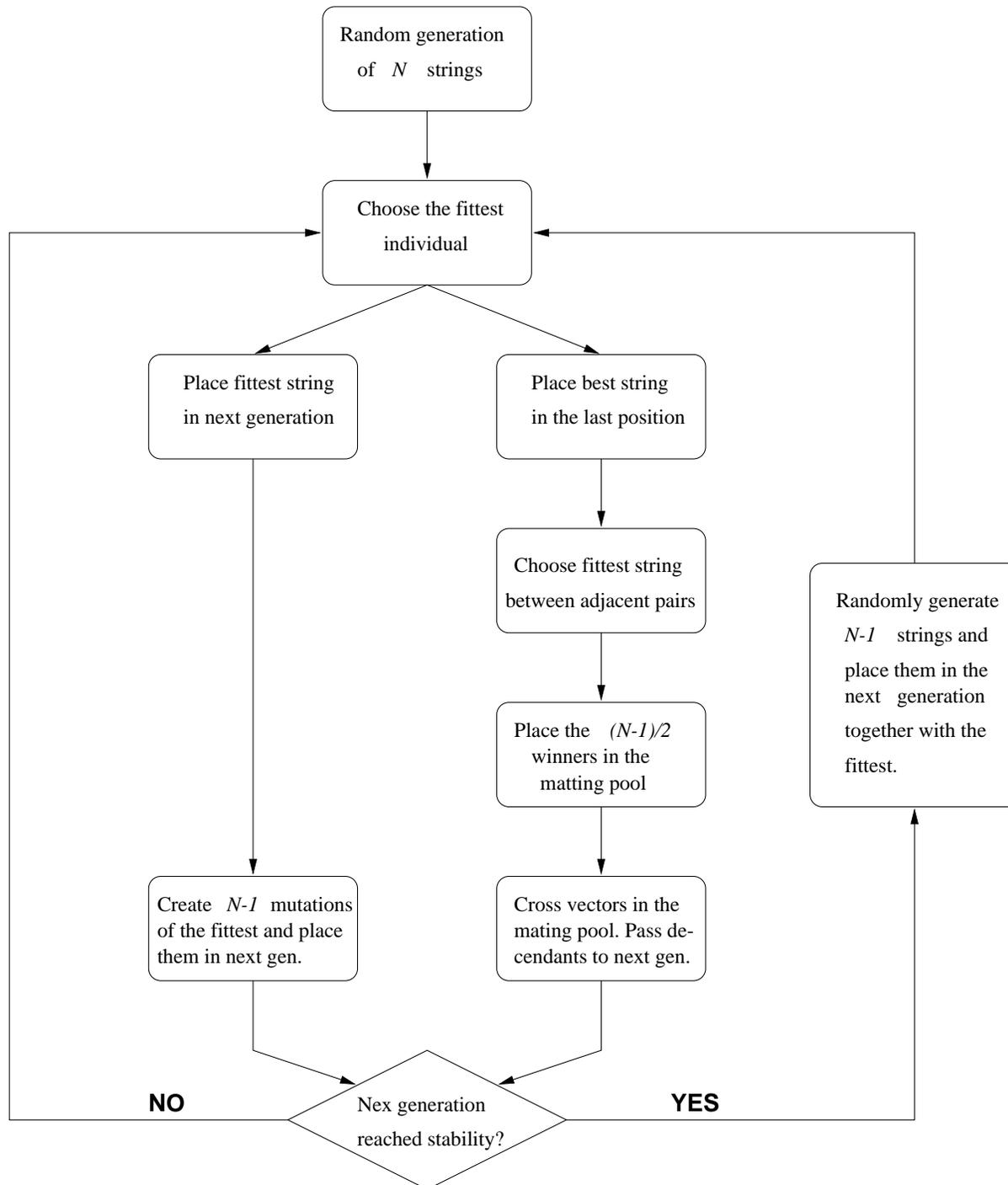


Fig. 8. General block diagram of the genetic algorithm.

able to use Matching Pursuit for image coding a fast computational tool has to be found. Genetic Algorithms seem to be quite adapted to Matching Pursuit needs. Even though they introduce a suboptimality factor, this does not imply a great quality loss in the final solution. So, Matching Pursuit with Genetic algorithms seems a promising approach to non-linear image coding.

REFERENCES

- [1] S. Mallat and Zhang, "Matching pursuit with time-frequency dictionaries," in *IEEE Transactions on Signal Processing*, december 1993, number 12.
- [2] Rosa M. Figueras i Ventura, "Image coding with matching pursuits," M.S. thesis, UPC-EPFL, 2001.
- [3] Pierre Vandergheynst and Pascal Frossard, "Efficient image representation by anisotropic refinement in matching pursuit," in *Proceedings of IEEE*, Salt Lake City UT, May 2001, ICASSP, vol. 3.
- [4] Pascal Frossard, Pierre Vandergheynst, and Murat Kunt, "Redundancy driven a posteriori matching pursuit quantization," in *IEEE Trans. Sig. Process.*, 2001, Submitted.
- [5] M. Gharavi-Alkhansari and T. S. Huang, "Fractal based techniques for a generalized image coding method," in *Proc. ICIP-94 IEEE International Conference on Image Processing*, Austin, Texas, November 1994.
- [6] Geoffrey Davis, Stéphane Mallat, and Marco Avellaneda, "Adaptative greedy approximations," *Constructive Approximations*, 1997, Springer-Verlag New YORK INC.
- [7] G. Devuyt, J-M. Vesin, P-A. Despland, and J. Bogouslavsky, "The matching pursuit: A new method of characterizing microembolic signals?," *Ultrasound in Medicine and Biology*, vol. 26, no. 6, pp. 1051–1056, April 2000.
- [8] F. Bergeaud and S. Mallat, "Matching pursuit of images," in *Proceedings IEEE Int. Conf. Image Processing*, Washington DC, October 1995, vol. I, pp. 53–56.
- [9] Osama K. Al-Shaykh, Eugene Miloslavsky, Toshio Nomura, Ralph Neff, and avideh Zakhori, "Video compression using matching pursuits," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, February 1999.
- [10] S. Mallat, *A wavelet tour of signal processing*, Academic Press, USA, 1998.
- [11] D. Beasley, D. R. Bull, and R. R. Martin, "An overview of genetic algorithms: Part 1, fundamentals," *University Computing*, vol. 15, no. 2, pp. 58–69, 1993.
- [12] D. Beasley, D. R. Bull, and R. R. Martin, "An overview of genetic algorithms: Part 2, researcg topics," *University Computing*, vol. 15, no. 4, pp. 170–181, 1993.
- [13] L. Davis, Ed., *Handbook of Genetic Algorithms*, Van Nostrand, 1991.
- [14] K.F. Man, K. S. Tang, and S. Kwong, *Genetic Algorithms*, Springer, London, 1999.
- [15] K. S. Tang, K. F. Man, S. Kwong, and Q. He, "Genetic algorithms and their applications in signal processing," *IEEE Signal Processing Magazine*, vol. 13, no. 6, 1996.
- [16] E. Cantú-Paz, "A summary of research on parallel genetic algorithms," IlliGAL 05007, Illinois Genetic Algorithm Laboratory, University of Illionis at Urbana-Champaign, 1995.
- [17] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone, *Genetic Programming An Introduction. On the Automatic Evolution of COmputer Programs and Its Applications*, dpunkt.verlag and Morgan Kaufmann Publishers, Inc., 1998.
- [18] M. P. Fourman, "Compaction of symbolic layout using genetic algorithm," in *Proc. 2nd Int. CONf. Genetic Algorithms*, 1985, pp. 141–153.
- [19] J. H. Holland, "Adaptation in natural and artificial systems," *MIT Press*, 1975.
- [20] Charles Darwin, *On the Origin of Species*, John Murray, London, 1859.
- [21] William S. Klug, *Concepts of Genetics*, Prentice Hall, 6th edition, 1997.

APPENDIX

A. EXPONENTIAL CONVERGENCE TO ZERO OF THE MP RESIDUAL

The following theorem proves that the module of the residue in a Matching Pursuit algorithm tends exponentially to 0:

Theorem 1: There exists $\lambda > 0$ such that for all $m \geq 0$ and $\forall f \in \mathbb{C}^{\mathbb{N}}$:

$$\|R^m f\| \leq 2^{-\lambda m} \|f\|. \quad (22)$$

As a consequence

$$f = \sum_{m=0}^{+\infty} \langle R^m f, g_{\gamma_m} \rangle g_{\gamma_m} \quad (23)$$

and

$$\|f\|^2 = \sum_{m=0}^{+\infty} |\langle R^m f, g_{\gamma_m} \rangle|^2 \quad (24)$$

where the convergence of 1 is intended in the strong sense.

Proof:

Let us verify that there exists $\beta > 0$ such that for any $f \in \mathbb{C}^{\mathbb{N}}$

$$\sup_{\gamma \in \Gamma} |\langle f_m, g_\gamma \rangle| \geq \beta \|f\| \quad (25)$$

Suppose that it is not possible to find such a β . This means that we can construct $\{f_m\}_{m \in \mathbb{N}}$ with $\|f_m\| = 1$ and

$$\lim_{m \rightarrow +\infty} \sup_{\gamma \in \Gamma} |\langle f_m, g_\gamma \rangle| = 0 \quad (26)$$

Since the unit sphere of $\mathbb{C}^{\mathbb{N}}$ is compact, there exists a sub-sequence $\{f_{m_k}\}_{k \in \mathbb{N}}$ that converges to a unit vector $f \in \mathbb{C}^{\mathbb{N}}$. It follows that

$$\sup_{\gamma \in \Gamma} |\langle f, g_\gamma \rangle| = 0, \quad (27)$$

so $\langle f, g_\gamma \rangle = 0$ for all $g_\gamma \in \mathcal{D}$. Since \mathcal{D} contains a basis of $\mathbb{C}^{\mathbb{N}}$, necessarily $f = 0$ which is not possible because $\|f\| = 1$. This proves that our initial assumption is wrong, and hence there exists β such that (25) holds.

The decay condition (22) is derived from the energy conservation:

$$\|R^{m+1}f\|^2 = \|R^m f\|^2 - |\langle R^m f, g_{p_m} \rangle|^2 \quad (28)$$

The Matching Pursuit chooses g_{γ_m} that satisfies

$$|\langle R^m f, g_{\gamma_m} \rangle| \geq \alpha \sup_{\gamma \in \Gamma} |\langle R^m f, g_\gamma \rangle| \quad (29)$$

and (25) implies that $|\langle R^m f, g_{\gamma_m} \rangle| \geq \alpha\beta \|R^m f\|$. So

$$\|R^{m+1}f\| \leq \|R^m f\| (1 - \alpha^2 \beta^2)^{\frac{1}{2}} \quad (30)$$

which verifies (22) for:

$$2^{-\lambda} = (1 - \alpha^2 \beta^2)^{\frac{1}{2}} < 1 \quad (31)$$

This also proves that

$$\lim_{m \rightarrow +\infty} \|R^m f\| = 0 \quad (32)$$