Successive Approximation Quantization for Image Compression

by Eduardo A. B. da Silva, Décio A. Fonini Jr., and Marcos Craizer

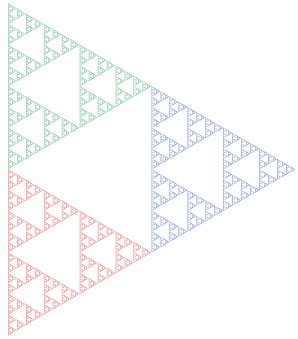
Abstract—*Successive approximation (SA) quantization is part of many of the state-of-the-art image and video compression methods. In this article we first make a review of it, starting from the classical optimality considerations of Equitz and Cover and then proceeding to Mallat and Falzon results concerning low bit-rate transform coding. We then develop a general theory of SA quantization which we refer to as α-expansions. This theory explains the published results obtained by both scalar and vector SA quantization methods, and indicates how further performance improvements can be obtained.*

1531-636X/10/$10.00©2002IEEE

Introduction

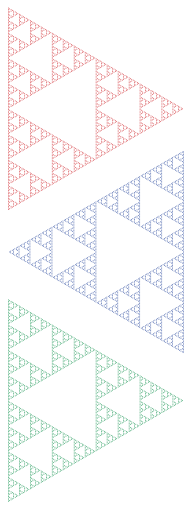
Images may be worth a thousand words, but they generally occupy much more space in a hard disk, or bandwidth in a transmission system, than their proverbial counterpart. So, in the broad field of signal processing, a very high-activity area is the research for efficient signal representations. Efficiency, in this context, generally means to have a representation from which we can recover some approximation of the original signal, but which doesn't occupy a lot of space. Unfortunately, these are contradictory requirements; in order to have better pictures, we usually need more bits.

The signals which we want to store or transmit are normally physical things like sounds or images, which are really continuous functions of time or space. Of course, in order to use digital computers to work on them, we must *digitize* those signals. This is normally accomplished by sampling (measuring its instantaneous value from time to time) and finely quantizing the signal (assigning a discrete value to the measurement) [1]. This procedure will produce long series of numbers. For all purposes of this article, from here on we will proceed as if these sequences were the original signals which need to be stored or transmitted, and the ones we will eventually want to recover. After all, we can consider that from this digitized representation we can recover the true (physical) signal, as long as human eyes or ears are concerned. This is what happens, for example, when we play an audio CD. In our case, we will focus mainly on image representations, so the corresponding example would be the display of a picture in a computer monitor. However, the discussion in this paper, and especially the theory developed here, apply equally well to a more general class of signals.



Several state-of-the-art image and video compression methods make use of successive approximation quantization, in one form or another [2–6]. It's found in the literature under several names, as successive refinements, embedded coding, or progressive quantization [4, 7–9]. However, all these refer to the same basic idea, namely that of achieving a representation for a signal made up of several segments; from the first, we can recover a coarse approximation of the original, and then the next segments give us finer and finer detail. Thus, by carefully controlling how many segments we store or transmit, we may control either the amount of data involved or the quality of the recovered approximation.

Images may be worth a thousand words, but they generally occupy much more space in a hard disk, or bandwidth in a transmission system, than their proverbial counterpart. So, in the broad field of signal processing, a very high-activity area is the research for efficient signal representations.



This is in fact a very simple idea and is implemented, for example, in the most common form of number representation, which is the decimal expansion. For example, when we want to refer to the value of π, we may use

Eduardo A. B. da Silva and Décio A. Fonini are with PEE/DEL/COPPE/EE, Universidade Federal do Rio de Janeiro, C.P. 68504, Rio de Janeiro, RJ, 21945-970, Brazil, e-mail: {eduardo, fonini}@lps.ufrj.br.

Marcos Craizer is with DMAT, Pontifícia Universidade Católica do Rio de Janeiro, Rua Marquês de S. Vicente, 225, Rio de Janeiro, RJ, 22453-900, Brazil, e-mail: craizer@mat.puc-rio.br.

But what we are really looking for is an acceptable compromise between compression levels and image quality. In general, acceptable means a quality loss not readily apparent to a casual viewer. In particular, coarser quantization of brightness levels is usually unacceptable, because our visual system is very aware of the resulting difference [10]. Moreover, this difference is usually perceived as somewhat aggressive to the eye. For comparison, take a low-pass filtered image: although the difference to the original is still quite visible, the result has an agreeable, soothing effect.

the string "3.14"; if we need more precision (and we can afford the cost), we may then use "3.14159". This second string contains the first at its beginning, and then some more data which does not mean anything by itself, but does give us more information about the value of π when used together with the first part.

Sadly, the decimal expansion is good only for human consumption; besides, it's not optimal for representing signals. Its digital cousin, the binary expansion, is better suited for computer use, but is equally non-optimal. Suffice it to show that the optimal (least error) 5-digit representation for π is 3.1416; to reach the optimal 6-digit representation, we must get rid of the final 6, substituting 59 for it. So, this representation does not make optimal use of the allowed space.

Compression

Image compression is usually obtained by leaving out "unnecessary" detail. For example, suppose our image is composed of 8-bit pixels, where the values from 0 to 255 represent the brightness of the corresponding pixel. We could retain only the most significant 4 bits of each pixel, which would result in an image with only 16 gray levels. We would have thus obtained 2:1 compression, although at the cost of a very degraded image. But what we are really looking for is an acceptable compromise between compression levels and image quality. In general, acceptable means a quality loss not readily apparent to a casual viewer. In particular, coarser quantization of brightness levels is usually unacceptable, because our visual system is very aware of the resulting difference [10]. Moreover, this difference is usually perceived as somewhat aggressive to the eye. For comparison, take a low-pass filtered image: although the difference from the original is still quite visible, the result has an agreeable, soothing effect.

Transforms

Invertible transforms are widely used in image processing. Their purpose is to represent an image as a weighted sum of elementary images. Mathematically, this is obtained by multiplying the number sequences which represent the signal (taken as a vector) by a matrix [10]. The elements of the resulting vector—the transformed image—have now a totally different meaning, not directly correlated

Invertible transforms are widely used in image processing. Their purpose is to represent an image as a weighted sum of elementary images.

with the brightness value of any single pixel. On the contrary, each coefficient of the transformed signal is a function of several pixels of the original image. Conversely, any alteration in a transform coefficient will translate to a modification in most of the pixels of the recovered image, when the inverse transform is applied. This is where most of the compression is obtained in modern methods: by carefully quantizing the different coefficients of the transformed image [11]. Depending on the transform used, the loss of precision in certain coefficients is less visible than in others, when considering the effects of the resulting recovered images in the human visual system [12–13].

Of course some transforms are better suited than others for this purpose. The JPEG standard [14] for compression of static images uses the Discrete Cosine Transform, and so do the several variants of the MPEG standard [15–17] for compression of moving images. The modern JPEG2000 standard [6], based on the EBCOT algorithm [4], uses the Wavelet Transform. MPEG-4 also allows (optionally) the use of wavelets.

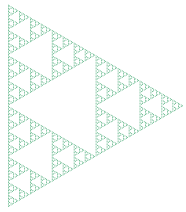
Image Quality, and Lack Thereof

The ultimate assessment of the quality of a compression method is the subjective impression caused by the recovered images on suitably trained human individuals. There's hardly any need to say that this is all but impossible to determine *a priori* algorithmically, so any objective methods for quality assessment fall very short of this ideal case [13]. Nonetheless, some objective methods must be used, at least as a first-order approximation for the quality of a method. Usually, what is used is some function of the brightness differences of the pixels in the original and the recovered images. The most common objective measure of the quality of an image (in relation to an original) is the MSE—Mean Squared Error, defined [10] as

$$\mathrm{MSE} \equiv \frac{1}{N} \sum (x_i - \hat{x_i})^2$$

where x_i is the value of an original pixel, and $\hat{x_i}$ is the corresponding approximation. The square function is used to make the definition more easily tractable in mathematical terms.

The careful reader will have noticed that this is just the Euclidean distance from the original image to its approximation: if the approximation is identical to the original, the MSE is zero, and it grows as the approximation departs from the original. This is a measure of *distortion*. For those who think that more is better, there is an inverse measure derived from the MSE, which is the PSNR—Peak Signal to Noise Ratio. For 256-gray level images, this is defined as



The elements of the resulting vector—the transformed image—have now a totally different meaning, not directly correlated with the brightness value of any single pixel. On the contrary, each coefficient of the transformed signal is a function of several pixels of the original image. Conversely, any alteration in a transform coefficient will translate to a modification in most of the pixels of the recovered image, when the inverse transform is applied. This is where most of the compression is obtained in modern methods: by carefully quantizing the different coefficients of the transformed image [11].

$$\text{PSNR} \equiv 10 \log_{10}\left(\frac{255^2}{\text{MSE}}\right)[\text{dB}]$$

Given the simplistic definition of distortion used here, we should keep in mind that those measures have a somewhat restricted use. The comparison of MSE or PSNR values is meaningful only when comparing similar algorithms applied to a common original image, and even so these results must be taken with a grain of salt. That said, the MSE is by far the most used form of comparing the image quality obtained from competing algorithms. Computational cost is often also taken into account.

Data Rates

The data rate is the other factor in the compression equation. We can define the data rate as the average number of bits per pixel needed to represent an image, and it has direct consequences on file sizes and bandwidth needs. For any given algorithm, better quality will imply larger data rates. The set of distortion versus rate measurements for an algorithm is referred to as its rate-distortion performance.

Information theory provides us with a theoretical upper bound on the performance of any coding algorithm, given the statistical behavior of the source of the coefficients [18]. A certain representation will be said to be optimal when this upper bound is achieved.

Successive Approximation Quantization

Although being part of many state-of-the-art image compression methods, successive approximation quantization is not generally optimal from a rate-distortion perspective. In this section, we will first describe the EZW algorithm [2], a wavelet-based image encoder that was the first one to efficiently use the concept of successive approximation quantization in image processing. We will then present a more formal definition of successive approximation, and then proceed to show why it's not always optimal. That said, we will present the reasons why it's used in so many algorithms nonetheless.

The EZW Algorithm

When the EZW—Embedded Zerotree Wavelet—algorithm for image compression was introduced [19], it represented a breakthrough in image compression methods. It is capable of compressing an image with excellent rate x distortion performance; besides, the generated bit-stream is such that the encoding rate can be precisely controlled, with optimal performance for all rates. Several other algorithms have since been introduced, all based on the same principle [3–5]. The key to EZW efficiency lies in a clever combination of wavelet transforms and successive approximation quantization.

The 3-stage wavelet transform of a natural image [20], as can be seen in Figs.1 and 2, has the following features:

- Its coefficients can be grouped in several frequency bands, obtained from an octave band decomposition. Band 0 is a low-pass band; bands 1, 4, and 7 have mainly vertical high frequency detail; bands 2, 5, and 8 have mainly horizontal detail, and bands 3, 6, and 9, diagonal detail. In each group, frequency increases as the band indices increase.
- It has a few high-energy coefficients and a large number of low-energy coefficients.
- The low-energy coefficients tend to appear in clusters.

Figure 1. 3-stage wavelet image transform.

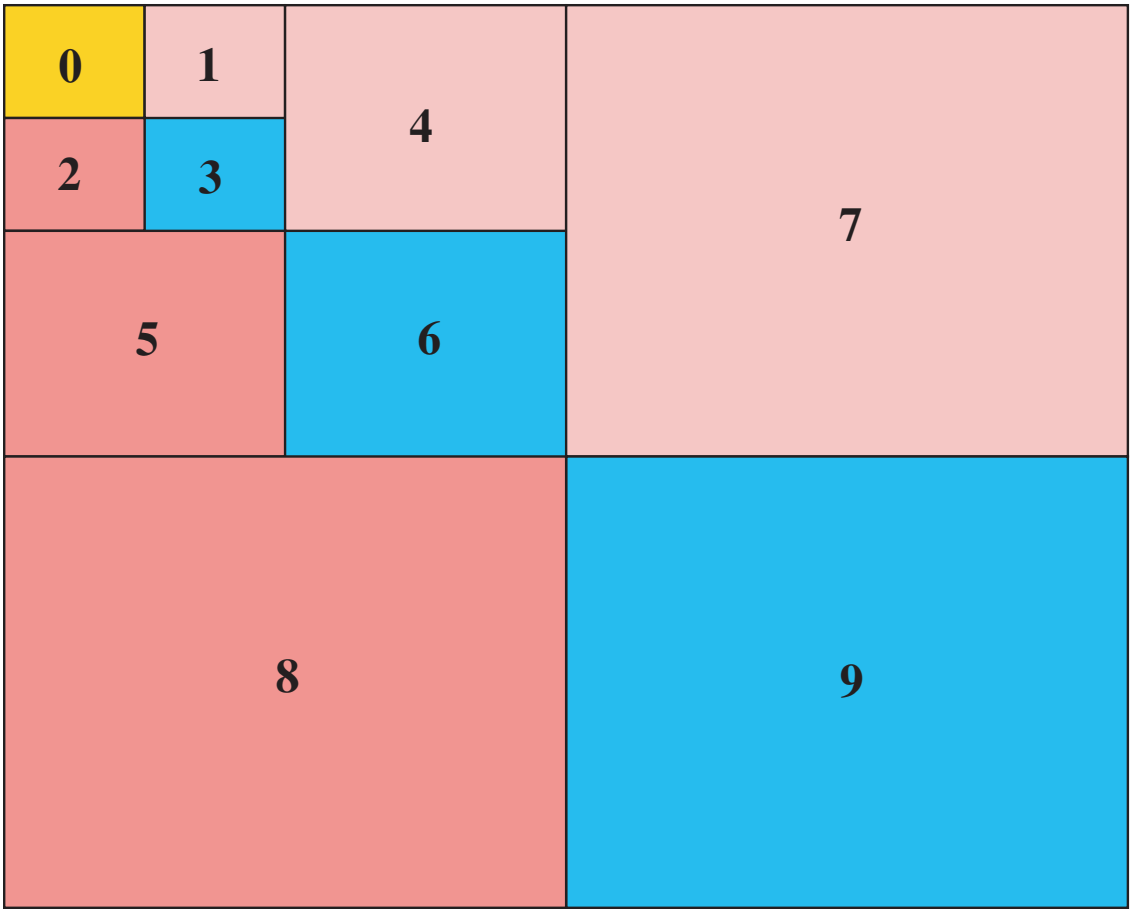
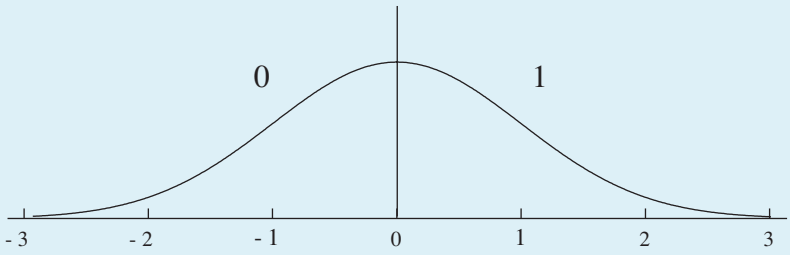
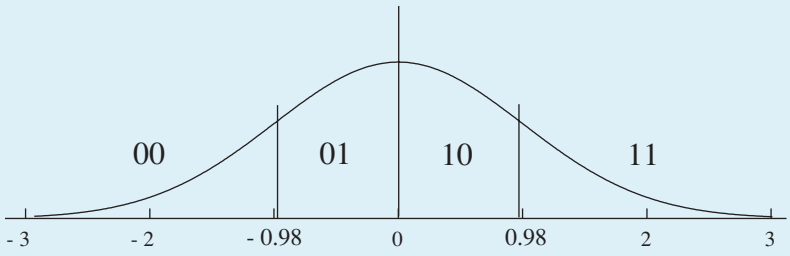


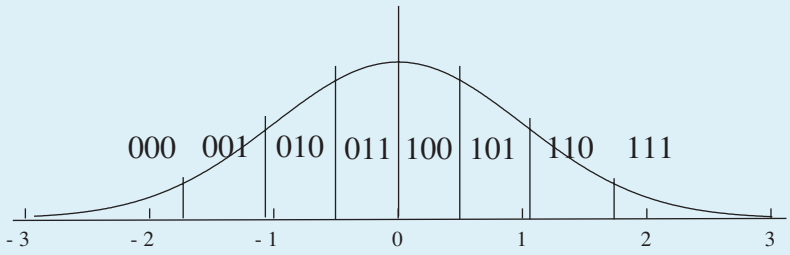
Figure 2. Wavelet sub-bands.



(a) 1-bit coding.



(b) 2-bit coding.



(c) 3-bit coding.

Figure 3. Coding a Gaussian random number.

- The coefficients in lower frequency bands tend to have higher energies than those in higher frequency bands.
- The bands of same orientation (1, 4, and 7; 3, 5, and 8; 3, 6, and 9) tend to have their low-energy coefficients in the same corresponding spatial locations.

The EZW algorithm works by quantizing all coefficients using successive approximation quantization. It does so by first computing the magnitude of the highest energy coefficient, and then setting a threshold T equal to half this value. It then transmits the sign of all the coefficients with magnitude higher than T. It is important to note that it uses a very efficient method to encode the location of these coefficients. It considers coefficients with magnitude lower than T as non-significant; if a coefficient in a band is non-significant, and all spatially corresponding coefficients in the higher bands of the same group are also non-significant, this whole set is then encoded as a zerotree. Otherwise, that non-significant coefficient is simply encoded as a zero. Note that with a zerotree a single symbol is used to represent a large number of coefficients; therefore, the zerotree concept is very efficient. Since the bands are self-similar, it is likely that a large number of zerotrees will occur, and thus just a small number of bits will be spent to encode all non-significant coefficients.

After the sign of the non-significant coefficients is transmitted, the threshold is halved. Then, the coefficients that are already significant are refined (by adding or subtracting $\frac{T}{2}$ from its reconstruction value), and the signs of the coefficients that just became significant are then transmitted. The whole process is repeated until a target distortion or bit-rate is achieved. All symbols are encoded using an arithmetic encoder [21]; details can be found in [2]. The rate x distortion characteristics of this encoder are very good; several variations of it have been proposed [3–5], based on the same basic principles. Such algorithms represented a breakthrough in image compression, and today they represent the state-of-the art in the field [4].

Successive Approximation Quantization for Image Compression

The successive approximation quantization lends an important characteristic to those algorithms: they are all embedded, that is, the bit-stream they generate for one rate contains the bit-streams for all achievable lower rates. This is detailed in the next section.

Embedded Coders

Shapiro [2] characterizes an embedded code as possessing two defining properties:

- When coding the same data with two different rates (and, implicitly, with two different distortions), the two resulting representations must match exactly for the extent of the smaller one. In other words, the smaller one must be reproduced exactly in the beginning of the larger one. This way, the coded representation for a given data rate contains all the representations for smaller data rates. In short, the representations get more precise as we add more symbols to them.
- It should obtain a good representation (low distortion) for a given data rate.

The former conveys the main idea of embedding, while the latter guards against excessive generalization[1]. For example, binary expansions can be considered embedded, if we take some precautions on reconstructing values, such as adding $\frac{1}{2^n}$ to a truncated number. On the other hand, representations generated by vector quantization [22] are usually not embedded: a high-resolution data stream cannot always be truncated to provide a low-resolution one.

[1] Just appending a whole high-rate description to a low-rate one, for example, should not qualify as embedding.

Not Always Optimal

Equitz and Cover (who use the terms Successive Refinements, in [7]) have shown that "optimal descriptions are not always refinements of one another"; they establish the conditions under which optimal successive refinements can occur. A simple example where this does *not* happen is also given, which goes as follows.

Take a sequence of numbers from a standard normal distribution $X \sim N(0, 1)$. The optimal 1-bit description for these numbers is obviously that which uses this bit to specify the sign of the number (see Fig. 3a). If we now decide to use 2 bits, we can have 4 quantizer bins, as in Fig. 3b. Notice that the new bins are subsets of the old bins. The placement of the bins' frontiers are defined by the Lloyd-Max conditions [22], in order to minimize the MSE of the reconstructed values. So far, we still have an embedded code; all numbers which were coded as 0 are now coded with a 0 as the first symbol. But if we proceed to 3-bit codes, the new bins (Fig. 3c) are not subsets of the previous bins anymore; so, we do not have embedded coding if we decide to keep the codes optimal.

It should be noted that optimality with respect to rate x distortion depends on how we decide to measure distortion. Although the MSE is the

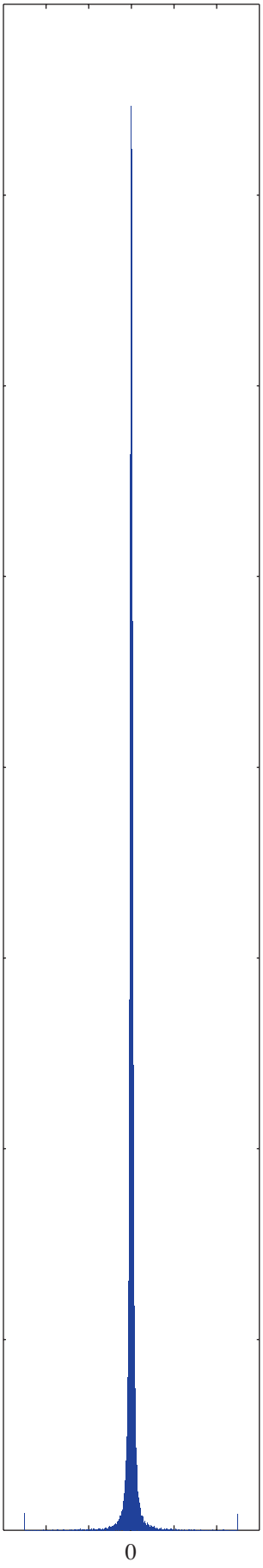


Figure 4. Histogram for DCT of the Lena image.

most common form, nothing prevents us from choosing another metric as, for example, the average absolute value of the error.

In short, given a random source with known properties, and a distortion function, there is a lower bound on the average rate necessary to represent messages (symbol sequences) from that source with a given distortion [18, 23]. Optimal representations would be those which reach this lower bound, and they usually would not be embedded.

The Case for Successive Approximations

State-of-the-art algorithms like EZW, described above, can presently represent most images with less than 1 bit per pixel (bpp), with no discernible loss of quality [2–5]. Most of those algorithms do use embedded coding of transform coefficients.

This is in apparent contradiction with the discussion shown in the previous section, *Not Always Optimal*. However, it turns out that no practical image coder is optimal, and embedded coders can be best, in certain cases, amongst the practical, non-optimal coders.

Low bit-rate quantization—Mallat & Falzon [24] have shown a mathematical analysis of the improvements obtained by embedded coding of transform coefficients when working with low bit rates, as follows.

A quantizer is considered to have high-resolution when the probability density function $p(x)$ of the quantized random variable X is approximately constant inside each quantizer bin. The rate x distortion performance of high-resolution quantizers for Gaussian processes is well known; the MSE will vary as $C(2^{-2R})$, with C depending on the bit allocation. In this case, the uniform quantizer has been shown to be optimal [22].

Current algorithms, however, operate on the $R < 1$ bpp region; therefore, the high-resolution hypothesis does not apply. In this case, the rate x distortion performance behaves differently.

The wavelet or block cosine transform of most images has histograms resembling that of Fig. 4, which depicts the histogram of the DCT of the Lena image. The clustering of coefficients near the zero region, together with the coarse quantization, results in a large number of coefficients being quantized as zero. Thus, the zero symbol deserves special treatment, and most algorithms treat it differently from the other values. For example, the EZW algorithm, as seen before, uses zerotrees in order to efficiently encode the zero elements. Mallat & Falzon, in [24], show that when there is a known transform behavior as to the usual location of large, average and small coefficients—as is the case with wavelet transforms in which small or zero-valued coefficients tend to be clustered—embedded coders can present an advantage over classical coders. The net result is to have the MSE varying approximately as R^{-1}. This explains the good performance of embedded encoders such as EZW and the like [2–5].

Applications of embedded codings—Besides presenting improved efficiency, embedded coders are better suited for several applications where the possibility of dynamically choosing rate or distortion is a plus.

Consider, for example, the multicasting of a movie, i.e., the transmission, through a network, from one source to several destinations. The network may form a tree, with the source as root and the destinations as leaves. Each branch may have a different bandwidth. Now, if the source limits the transmission to the rate allowed by the narrower branch, the other branches will not get all the quality they could. On the other hand, if the source transmits at full rate, some branches may not be able to cope with the large data rate. However, if the transmission uses an embedded code, special hardware at each branching point may dynamically select from the received bitstream just the amount of data which can be sent toward each of its outgoing branches. This way, all destinations get the largest possible rate—and thus image quality—allowed by the channel extending from them to the source.

As another example, imagine a high-resolution image database set up for remote browsing and downloading. We want to be able to make a fast browsing to select the images before actually downloading them. For this we must have a low-quality version of each image available, but the best dimensioning of these previews depends on the bandwidth of our connection to the server as well as the time available to complete the job, and may vary from user to user.

Consider, for example, the multi-casting of a movie, i.e., the transmission, through a network, from one source to several destinations. The network may form a tree, with the source as root and the destinations as leaves. Each branch may have a different bandwidth. Now, if the source limits the transmission to the rate allowed by the narrower branch, the other branches will not get all the quality they could. On the other hand, if the source transmits at full rate, some branches may not be able to cope with the large data rate. However, if the transmission uses an embedded code, special hardware at each branching point may dynamically select from the received bitstream just the amount of data which can be sent toward each of its outgoing branches.

If the image files are produced with a non-embedded coding, the only choice is to have separate files for the previews, which is inefficient; besides, there will be a fixed set of preview resolutions. Using embedded coding, the viewer can select[2] the data rate which best suits her as a compromise between image quality and download time.

A Real-World Example

Another nice example of a real-world method which uses successive approximations is Taubman's EBCOT [4]—Embedded Block Coding with Optimized Truncation, which is the basis for the JPEG2000 standard [6].

[2] We suppose the server can generate the previews at the selected data rate by selectively transmitting the proper subsets of the embedded coding.

The successive approximations in EBCOT appear at the bit-plane coding of the quantized values of the wavelet transform coefficients. It uses uniform, scalar quantization, except for the near-zero region. In bit-plane coding, the most significant bits of a set of values are sent first, then the next most significant, and so on, until all bit-planes have been sent. Of course, like other modern algorithms, the bit-planes are also further entropy coded to take advantage of the redundancy present in the transform coefficients.

Successive Approximations with α-Expansions

We will now develop a theory of successive approximations based on an embedded coding referred to as α-expansions. They can be seen as a generalization of the successive approximation quantization used in the state-of-the-art image transform coding algorithms. With it, we develop a general framework for analyzing embedded representations. We will also show that α-expansions are capable of improving the coding of wavelet coefficients when compared to the usual successive approximation scalar quantization.

Our first goal is to develop a representation for generic sequences of real numbers, through the use of α-expansions. Later, we will apply it to the coding of wavelet coefficients.

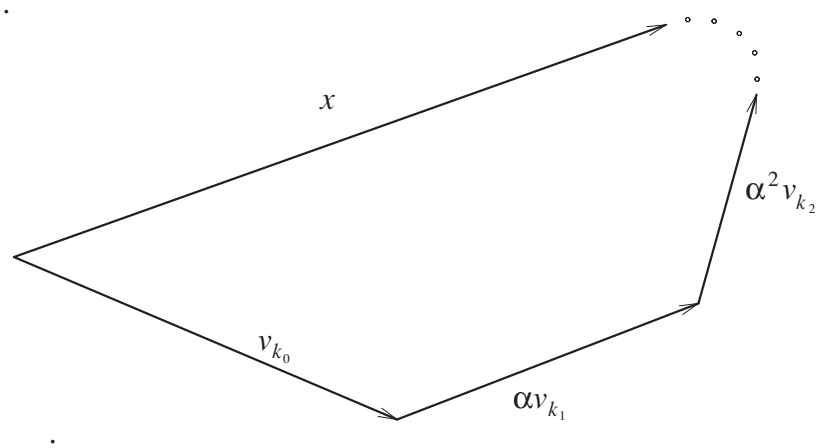


Figure 5. First Approximations.

Definition of α-Expansions

Let x be a vector in $\mathbb{R}^N$, and let $V = \{v_k \in \mathbb{R}^N, k \in \{1, 2, \ldots M\}\}$, where x is the vector to be coded and V is the DICTIONARY or CODEBOOK.

The SUCCESSIVE APPROXIMATIONS (SA) representation of the vector x is

$$x = \sum_{i=0}^{\infty} \alpha^i v_{k_i} \qquad (1)$$

where $0 < \alpha < 1$ and $k_i \in \{1, 2, \ldots M\}$. This is what we will call an α-expansion over the codebook V.

This representation can be coded as the sequence $(k_0, k_1, \ldots)$ of integers (which we will call a *k-sequence*). Geometrically, the relation expressed by Eq. 1 could be shown as in Fig. 5.

As the figure suggests, given only the first L elements $(k_0, k_1, \ldots k_{L-1})$ of the k-sequence, we can find an approximation x_L to the vector x:

$$x_L = \sum_{i=0}^{L-1} \alpha^i v_{k_i} \qquad (2)$$

To measure the error incurred by this approximation we define the residue

$$r_L = x - x_L \qquad (3)$$

For this truncated representation to be useful, we want $|r_L|$ to have a smoothly decreasing upper bound, so that r_L tends to zero as L grows. To be comparable to other quantized signal representations, we need $|r_L|$ to be bounded (at least) by $C\beta^L$, where C is some positive real constant and $0 < \beta < 1$. Observe that this only means that as we proceed with the coding, the *maximum error* in the reconstructed vector will decrease exponentially; $|r_L|$ itself may vary otherwise, as long as it stays below this upper bound.

The successive approximations coding of a vector satisfies the first re-

quirement for an embedded coder: higher rates are obtained by adding elements to the sequence $(k_0, k_1, \ldots)$. So, it contains all achievable lower-rate representations. The second requirement may also be met, depending on the algorithm chosen to produce the sequence.

The Search for Convergence

In [25], α-expansions were used to quantize wavelet coefficients. The method used was to group the coefficients into N-dimensional vectors and then find k-sequences to represent them. The method was restricted to codebooks of unitary vectors.

A *greedy* algorithm was used to find the k-sequence corresponding to a vector x, as follows:

1. Take $r = x$.
2. Find v_k, the code-vector nearest to r, i.e., the one which minimizes $|r - v_j|$. As all code-vectors are of unit length, this is equivalent to finding the code-vector which minimizes ang (r, v_j), the angle between r and v_j.
3. Take $r - v_k$ as the next value of r.
4. Multiply all code-vectors by α.
5. If $|r|$ is small enough, stop; else, go back to step 2.

The algorithm is said to converge when $|r|$ decreases exponentially, i.e., it is bounded by $C\alpha^n$ after the n-th step, where C is some real constant. In this case, the α-expansion is defined by the sequence of numbers k found at each iteration of step 2. The fact that we minimize the approximation error at each step justifies the classification of the algorithm as a greedy one.

Now, take an x formed by a given, random k-sequence. Observe that there is no guarantee that the algorithm will find the same k-sequence. In fact, it can even fail to find any k-sequence at all, i.e., it may diverge. Of course, we need to find a codebook V and a value α such that the algorithm converges for any $x \in \mathbb{R}^N$ [26].

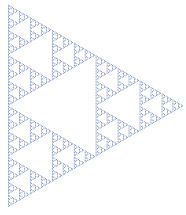
> We will now develop a theory of successive approximations based on an embedded coding referred to as α-expansions. They can be seen as a generalization of the successive approximation quantization used in the state-of-the-art image transform coding algorithms.

The first experiments [25] conducted to study the convergence of the algorithm suggested a connection between α and a certain geometric property of the codebook, an angle which we will call $\theta_{\max}$. Simply put, $\theta_{\max}$ is the maximum angle between any vector x and any codebook vector v_k. So, for a given number of vectors in a codebook, $\theta_{\max}$ can be minimized by having the codebook vectors as uniformly spaced as possible.

Later [27], a more precise evaluation of the exact relationship between α and $\theta_{\max}$ needed for convergence was obtained. The main results are:

1. Convergence can be guaranteed for any codebook, provided that (i) $\theta_{\max} < \pi/2$ and (ii) $|x| \le \beta$, where β is defined in Eq. 4;
2. Under the above conditions, there exists a special value $\alpha_{\min}$ such that convergence can be obtained with any $\alpha \ge \alpha_{\min}$.
3. $\alpha_{\min}$ is given by the following

$$\begin{cases} \alpha \ge \frac{1}{2\cos\theta_{\max}} \\ \beta = 2\cos\theta_{\max} \end{cases} \quad , \quad \text{if} \quad \theta_{\max} \le \pi/4$$
$$\begin{cases} \alpha \ge \sin\theta_{\max} \\ \beta = \frac{1}{\cos\theta_{\max}} \end{cases} \quad , \quad \text{if} \quad \theta_{\max} \ge \pi/4 \qquad (4)$$



The relationship between α_{min} and θ_{max} is shown graphically in Fig. 6.

Thus, "good" codebooks have small θ_{max}, so they converge with a small α, which in turn gives us faster decreasing errors in the approximations. Codebooks with more vectors will generally have smaller θ_{max}, but the corresponding k-sequences will have more entropy, so we will need a larger data rate to code them. Note that the conditions given here are sufficient, but not necessary, providing just an upper bound for α_{min}; the real α_{min} for a given codebook may be smaller.

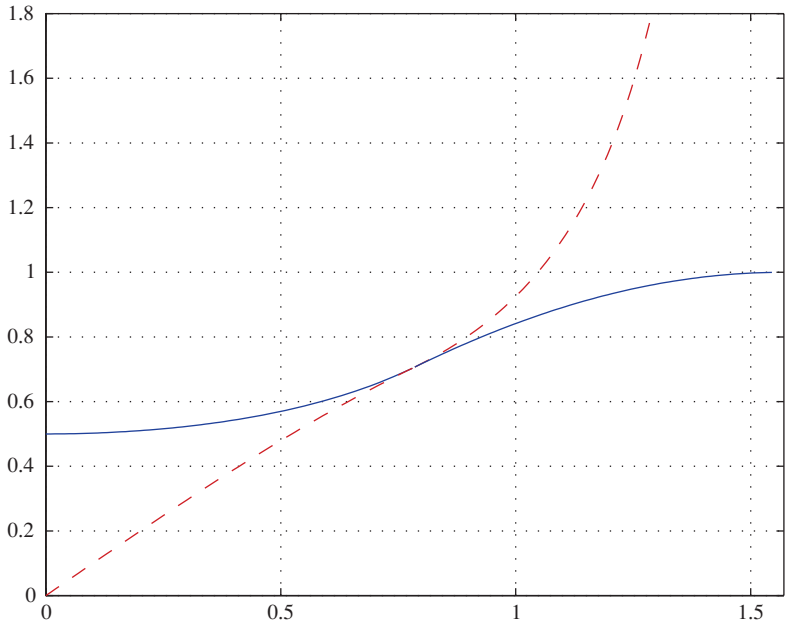


Figure 6. $\alpha_{min} \times \theta_{max}$.

Generated Spaces

We will now proceed toward a still more general theory of α-expansions. Instead of looking directly for convergence conditions, let us try to look at the problem from another point of view: we may consider the set of all vectors that can be generated by the right side of Eq. 1; then we can investigate the conditions for this set to contain the unit ball on $\mathbb{R}^N$.

Let us consider a 2-D example. Let V be the set defined in Fig. 7, where the three vectors are of unit length and evenly spaced on the circle. Let $\alpha = 0.5$. The possible values of x_1 are the codebook elements themselves:

$$x_1 = \begin{cases} v_1 \\ v_2 \\ v_3 \end{cases}$$

On the next iteration, we have nine possible values for x_2:

$$x_2 = \begin{cases} v_1 \\ v_2 \\ v_3 \end{cases} + \begin{cases} \alpha v_1 \\ \alpha v_2 \\ \alpha v_3 \end{cases}$$

which can be seen in Fig. 8a. As L grows, we can see that the set of possible points x_L is as depicted in Fig. 8b.

Now, this last figure looks a lot like a fully fledged fractal[3]. If we can prove that it is indeed a fractal (when $L \to \infty$), and that this happens for any codebook and scale factor α, we can then use fractal theory to verify under what conditions this fractal will contain the unit ball, so that all values $|x| \le 1$ can be represented as an α-expansion over that codebook.

Iterated Function Systems (IFS)

Let us define x, k, V, and α as in the section *Definition of* α*-Expansions*. Now define the functions

$$f_k(x) = \alpha x + v_k \qquad (5)$$

and

$$F(A) = \bigcup_1^M f_k(A) \qquad (6)$$

where A is any subset of $\mathbb{R}^N$ and $f_k(A) = \{f_k(a), a \in A\}$. Note that F is defined on the set[4] of the subsets of $\mathbb{R}^N$, and that f_k is extended to apply to elements of this set also.

[3] Indeed, this set approximates the Sierpinski triangle [28], of fractal theory fame.

[4] This set may become a *space*, if we define a proper *metric* for it.

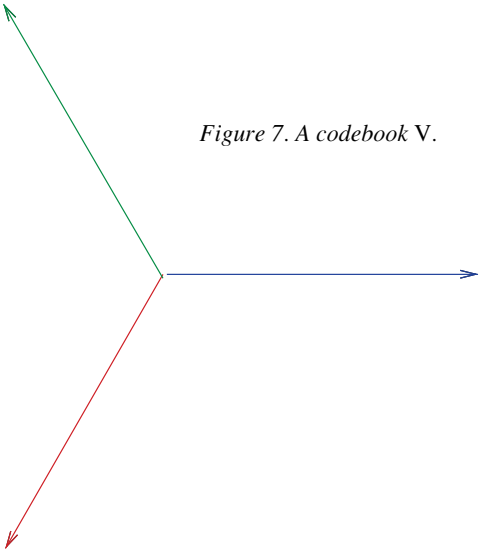


Figure 7. A codebook V.

From fractal theory [28], since each f_k is a *contraction mapping* (specifically for $0 < |\alpha| < 1$), it follows that the set $\{f_k\}$ forms an *iterated function system*. One of its properties is that there is a set Λ, which is the unique fixed point of F (i.e., $F(\Lambda) = \Lambda$) and is called the *attractor* of the IFS.

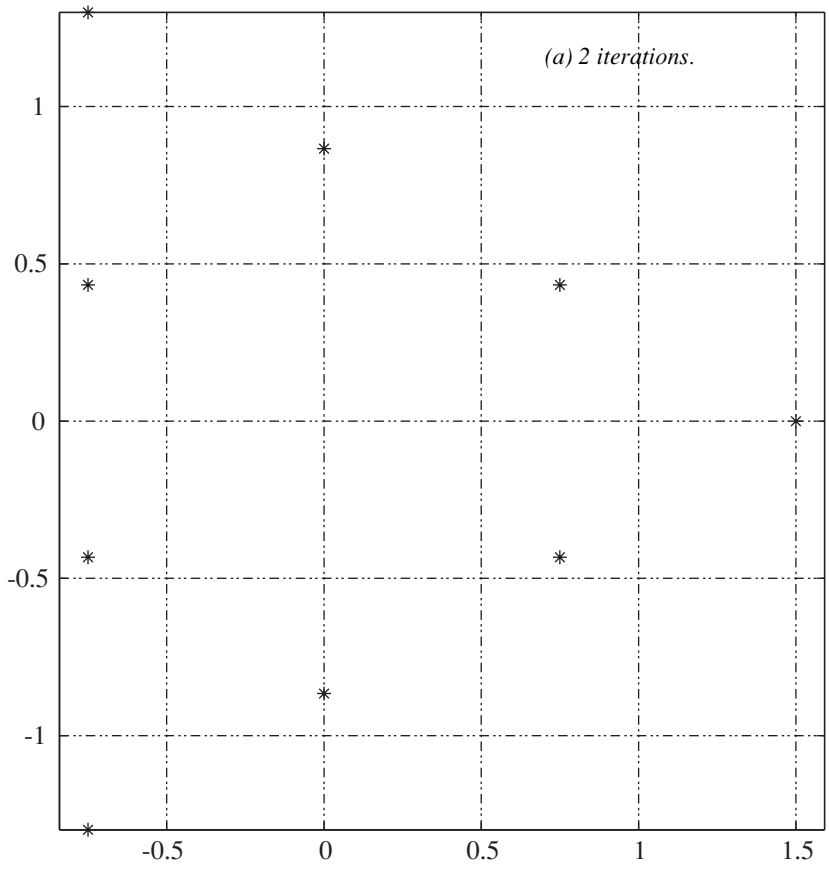
Theorem: The set of all points that can be represented by the right side of Eq. 1, is the attractor of the corresponding IFS.

It is a fact that IFS attractors are generally fractals; they can also degenerate into "normal" (i.e., non-fractal) sets. For example, in Fig. 8b, Λ is the whole punctured triangle; each colored smaller copy is one of the $f_k(\Lambda)$. We must find out the cases in which the attractor will contain the unit ball.

Determining Λ

Geometric background—In the following discussion, we will use some readily available [29] geometric definitions and results. These are summarized here.

Combination: Let $a_1, \ldots a_k \in \mathbb{R}$, and $p_1, \ldots p_k \in \mathbb{R}^N$. The linear combination $x = \sum_{i=1}^{k} a_i p_i$ is said to be:



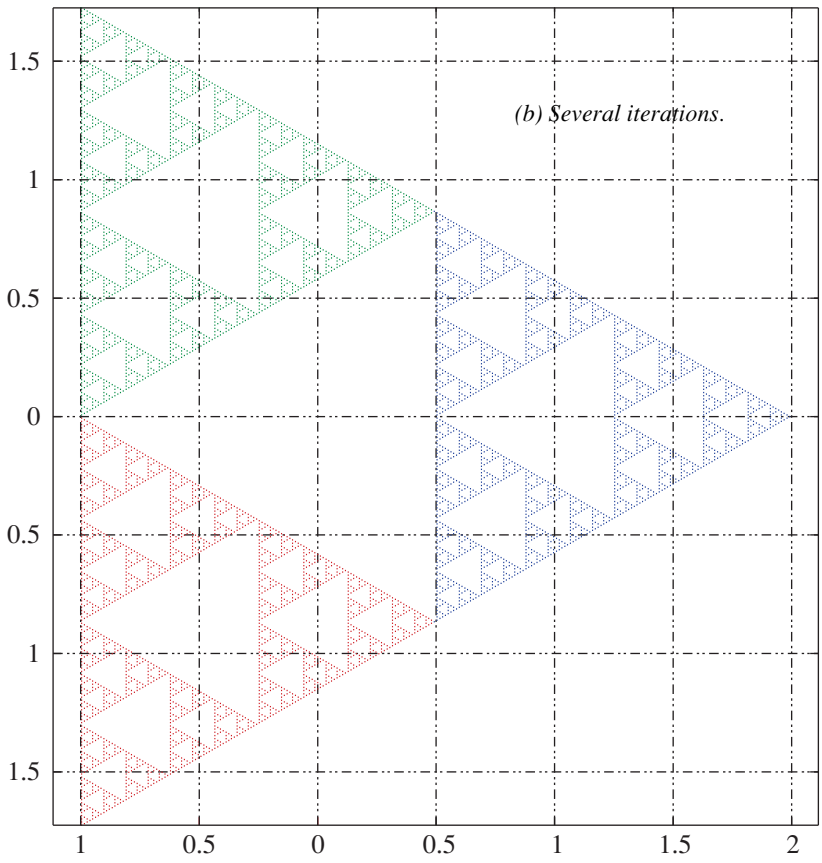


Figure 8. Possible values for x_2 *and* x_L.

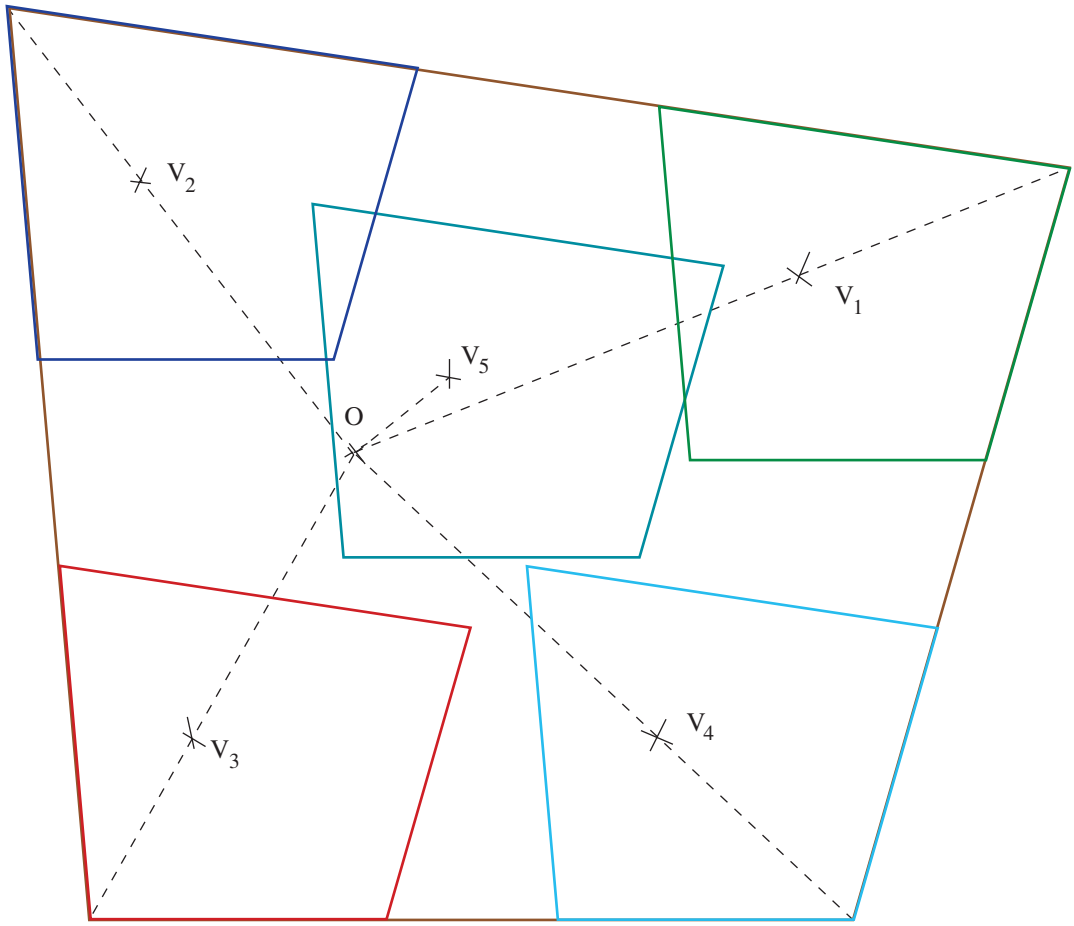


Figure 9. 5-point codebook and P_α.

- a *positive combination*, if $a_i \geq 0$, $\forall i \in (1, \ldots k)$;
- an *affine combination*, if $\sum_{i=1}^{k} a_i = 1$;
- a *convex combination*, if both previous conditions hold.

Convex sets: A set S is said to be convex if for every $x, y \in S$, every convex combination of x and y is also in S.

Convex hull: The convex hull of a set S is the smallest convex set containing S.

Polytope: The convex hull of any finite set S is a polytope; conversely, every polytope is the convex hull of a subset of its points, called its vertices. Polygons and (3-D) polyhedra are examples of polytopes. Note that infinite *sets* may have a convex hull which is not a polytope, e.g. a sphere.

Remark: The more generic definition of polyhedron includes unbounded sets defined as convex intersections of half-spaces, and polytopes as bounded polyhedra; for our purposes, however, we are considering only bounded sets.

Vertex: A vertex of a polytope can also be defined as those points which cannot be expressed as a convex combination of any other points of the polytope. A polytope is also the set of all convex combinations of its vertices. Observe that not all points in S (the finite set over which we build the convex hull) will necessarily be vertices; some may be in the interior of the polytope.

Homothety: An homothety of center C and scaling factor α is the transformation

$$h(x) = C + \alpha(x - C)$$
$$= \alpha x + (1 - \alpha)C$$

We can also have $h(S) = \{h(s), s \in S\}$, i.e., extend h to apply to a set.

Homothety of a convex set: Let S be a convex set, $\alpha \in [0, 1]$, $P \in S$, and $T = \alpha S + (1 - \alpha)P$. Then $T \subset S$. Conversely, if $\alpha > 1$, then $T \supset S$.

The codebook's polytopes—Let P be the polytope defined by the convex hull of the vectors $\{v_k\}$. Likewise, let P_α be the convex hull of the set $W = \{w_k = \frac{1}{1-\alpha} v_k\}$. Observe that P_α and W can be obtained from P and V, respectively, by the same homothety with center at the origin and scaling factor $\frac{1}{1-\alpha}$.

By construction, any $x \in \Lambda$ is a positive combination of vectors from the codebook V, with coefficients whose sum is $\frac{1}{1-\alpha}$, thus a convex combination of the vectors w_k. Therefore,

$$\Lambda \subseteq P_\alpha \tag{7}$$

Observe that we can rewrite $f_k(x)$ as

$$f_k(x) = \alpha \left(x - \frac{1}{1-\alpha} v_k \right) + \frac{1}{1-\alpha} v_k$$
$$= \alpha (x - w_k) + w_k$$

Therefore, each f_k is an homothety of center w_k and scaling factor α, which implies[5] that $f_k(P_\alpha) \subset P_\alpha$. This way, $F(P_\alpha)$ is the union of several scaled-down copies of P_α, each of them contained in P_α.

That said, two situations may arise: either

$$F(P_\alpha) = P_\alpha \Rightarrow \Lambda = P_\alpha$$

or

$$F(P_\alpha) \subset P_\alpha \Rightarrow \Lambda \subset P_\alpha$$

where $\subset$ is used here meaning "strictly contained". In the first case, all points inside P_α (and only these points) can be represented by the α-expansion; in the other, only a subset of P_α can be so represented. We must know the conditions under which we will have $\Lambda = P_\alpha$.

Geometrically, the condition $F(P_\alpha) = P_\alpha$ means that the various $f_k(P_\alpha)$ must cover P_α. To better appreciate this condition let's examine a situation where it *fails*, as in Fig. 9. It represents a codebook with $N = 2$ and $M = 5$. Observe that P_α has only four vertices, for the 5th code-vector is inside the convex hull of $\{v_k\}$. In this case, $F(P_\alpha)$ (the union of the copies) *does not cover* P_α.

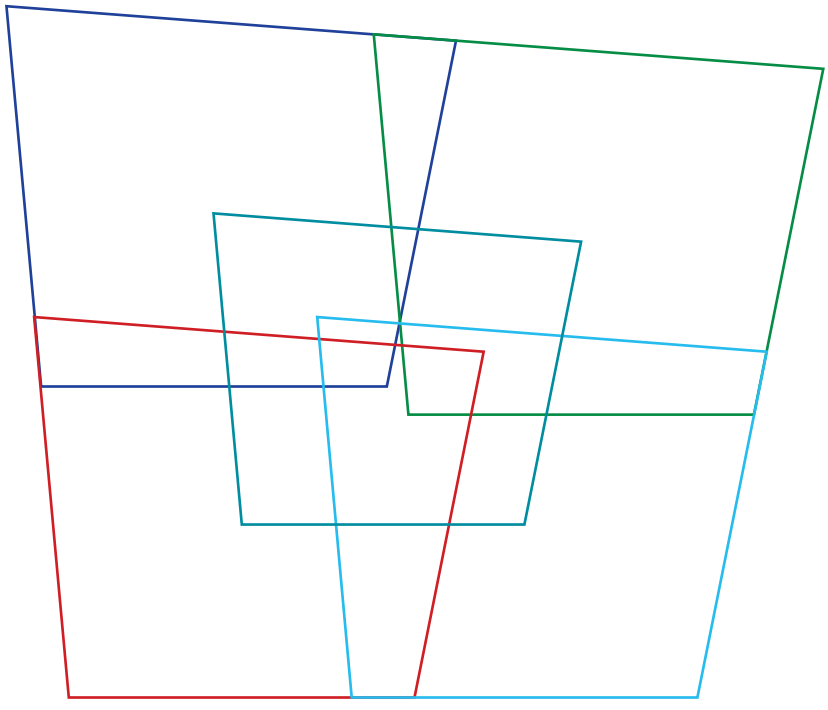
As another example, consider the codebook formed by the four vertices of the inner polygon in Fig. 10. Here we use $\alpha = 0.55$, the smallest value for which the covering is complete ($F(P_\alpha) = P_\alpha$).

Determining the α-Expansion Coefficients

A rule to determine (one) α-expansion for x is:

1. Determine k such that $x \in f_k(P_\alpha)$ (there may exist more than one

Figure 10. A covered P_α.



[5] Since $0 < \alpha < 1$, $w_k \in P_\alpha$ and P_α is convex.

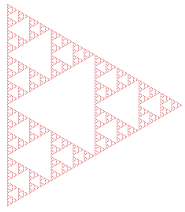
such k); $\Lambda = P_\alpha$ guarantees the existence of at least one such k.

2. Determine $y = (x - v_k)/\alpha$; $\Lambda = P_\alpha$ guarantees that $y \in P_\alpha$.
3. Take this y as the next x.

The repeated application of these steps will generate the desired sequence $(k_0, k_1, \ldots)$.

Each sequence element localizes x as being into one of the copies of the outer region; since this copy has the same form as the outer region, it can also be so divided, thus permitting the process to continue indefinitely. The regions so obtained are smaller and smaller, thus reducing the possible locations of x. In short, the k-sequence defines a region in space where x is contained. Also, this same sequence can be used to find the approximation x_L (see Eq. 2).

Whenever a point pertains to more than one $f_k(P_\alpha)$, any of the corresponding k can be selected, leading to different but equivalent representations (k-sequences) for the same point.



Binary Expansions as a Special Case

The binary representation of a number in the interval [0, 1] is a sequence $(b_1, b_2, \ldots)$ whose value is

$$x = \sum_{i=1}^{\infty} b_i \left(\frac{1}{2}\right)^i, \; b_i \in \{0, 1\}$$

which can also be written as

$$x = \sum_{i=0}^{\infty} b_i \left(\frac{1}{2}\right)^i, \; b_i \in \left\{0, \frac{1}{2}\right\}$$

In this form, it can immediately be recognized as an α-expansion.

Consider the codebook $V = \{v_0 = 0, v_1 = 1/2\}$, and $\alpha = 1/2$. This leads to P_α being the closed interval [0, 1], as depicted in Fig. 11. Since P_α is totally covered by its two halves, we have $\Lambda = P_\alpha$. Thus, any number in this interval can be represented as an α-expansion of this codebook. Take, say, $x = 0.1011 \ldots$:

$$x = v_1 + \frac{1}{2}\left(v_0 + \frac{1}{2}\left(v_1 + \frac{1}{2}\left(v_1 + \ldots\right)\right)\right)$$

$$= \left(\frac{1}{2}\right)^0 v_1 + \left(\frac{1}{2}\right)^2 v_1 + \left(\frac{1}{2}\right)^3 v_1 + \ldots$$

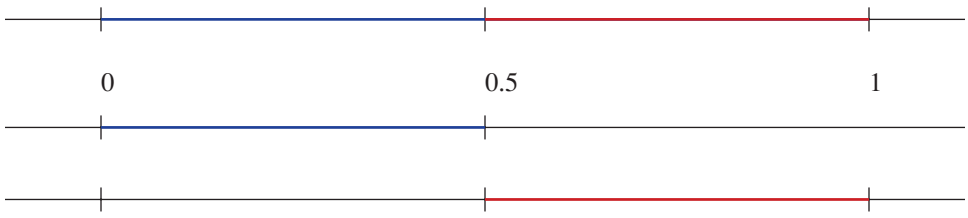


Figure 11. The [0, 1] interval.

A very important conclusion may be drawn from the above discussion:

> *For any codebook, convergence for a given* α *depends only on its geometrical* form, *not on its size, orientation or position with relation to the origin. By* form *we mean the relative position of all elements, including those which may be inside the convex hull.*

The two half-segments intersect at $x = 0.5$; that means that this point can have more than one representation. Indeed, $0.1000 \ldots = 0.0111 \ldots$. Note also that the point $x = 1$, contained in P_α, *can* be represented as $0.1111 \ldots$.

Similarly to the scalar case, any vector in the unit hyper-cube $[0, 1]^N$ can be coded as an α-expansion over the codebook $V = \{[0, \frac{1}{2}]^N\}$ (i.e., a

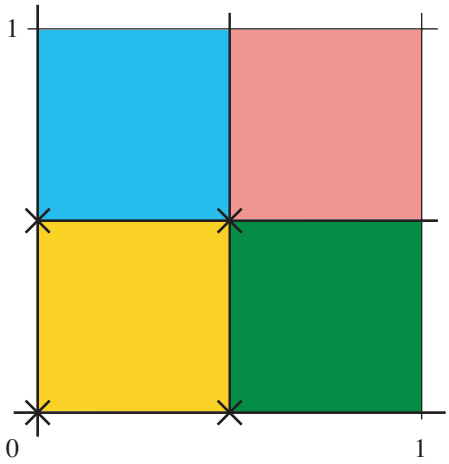


Figure 12. 2-D binary expansion.

2^N-element set of N-dimensional vectors whose coordinates are either 0 or 1/2). The 2-D case is illustrated in Fig. 12.

Any hyper-cube can be covered by 2^N copies of itself, where the side of each copy is 1/2 of the original. Although there are non-empty intersections, the total hyper-volume of these intersections is null. Note that in this case α reaches its theoretical lower bound (see the next section, *Bounds on* α *for* $\Lambda = P_\alpha$).

Bounds on α *for* $\Lambda = P_\alpha$

For any codebook, there is a critical value α_c such that $\Lambda_\alpha = P_\alpha \Leftrightarrow \alpha \geq \alpha_c$. In addition, we can show bounds for α_c as functions of N and M—the codebook's dimension and cardinality. These bounds are stated below.

Theorem: $\Lambda = P_\alpha \Rightarrow \alpha \geq M^{-\frac{1}{N}}$

Thus, given only the cardinality M and dimension N, $M^{-\frac{1}{N}}$ is the largest possible such bound for a generic codebook, since this value of α holds for the usual binary (or decimal, and so forth) expansion (see the previous section, *Binary Expansions as a Special Case*). In fact, for the binary expansion of vectors in $\mathbb{R}^N$, we have $M = 2^N$ vectors, yielding $\alpha \geq 1/2$, and convergence is obtained with $\alpha = 1/2$.

Theorem: $\alpha \geq \dfrac{N}{N+1} \Rightarrow \Lambda = P_\alpha$

This bound is effectively reached for all simplices (the N-dimensional polytopes with $N + 1$ vertices). Thus, this is the smallest possible such bound on α_c. However, for generic codebooks, we may possibly have $\Lambda = P_\alpha$ for smaller values of α.

The proofs of the above results are somewhat lengthy and are not in the scope of the present article. A more detailed discussion of this subject, including the pertinent demonstrations, is available on-line [30].

The Nearest Point Algorithm

The main step in the algorithm described in the section *Determining the* α*-Expansion Coefficients* involves knowing whether a point is inside a polytope or not, so that we can find which values of k satisfy $x \in f_k(P_\alpha)$. Do not be deluded by the simplicity of the 2-D case; in general, this is a rather elusive problem, and there seems to be no feasible algorithm to solve it (i.e., one that runs in polynomial time). So, we will resort once again to the greedy algorithm of the section *The Search for Convergence*, with some minor modifications to make it more suitable to a computer implementation, as well as to allow an easier analysis.

Any hyper-cube can be covered by 2^N copies of itself, where the side of each copy is one-half of the original. Although there are non-empty intersections, the total hyper-volume of these intersections is null.

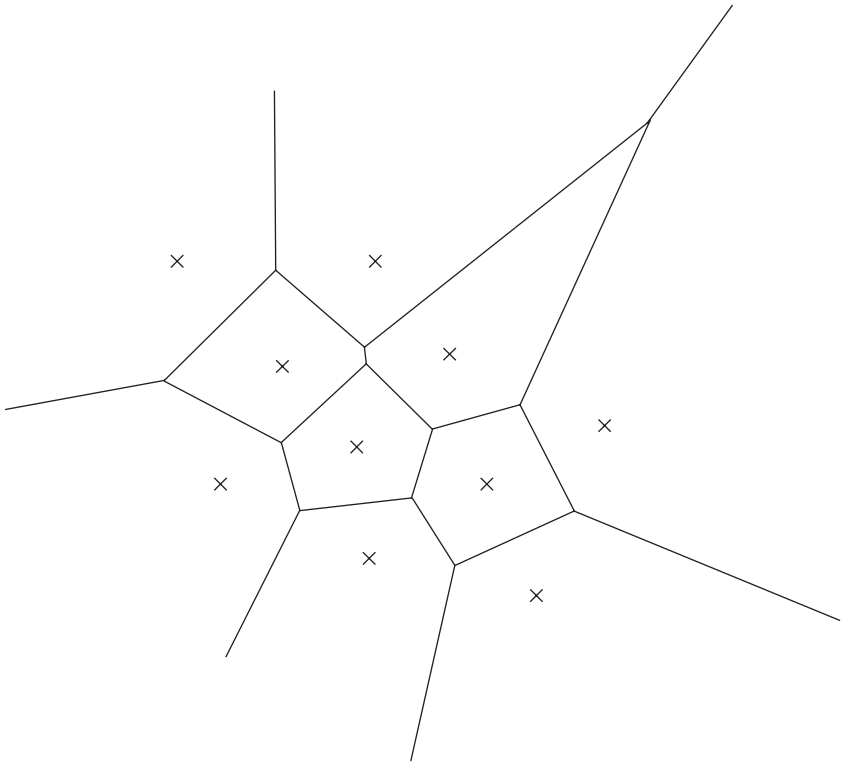


Figure 13. Voronoi cells: (a) Points and cells.

The revised algorithm is as follows:

1. Take $r = x$.
2. Find v_k, the code-vector nearest to r, i.e., the one which minimizes $|r - v_j|$. As all code-vectors are of unit length, this is equivalent to finding the code-vector which minimizes $\text{ang}(r, v_j)$, the angle between r and v_j.
3. Take $(r - v_k)/\alpha$ as the next value of r.
4. If $|r|$ is small enough, stop; else, go back to step 2.

Observe that we modified rule No. 3 and took out rule No. 4 of the algorithm in the section *The Search for Convergence*. This amounts to scaling up the residue at each step instead of working with successively smaller versions of the codebook. Here, we say that the algorithm converges if $|r|$ remains bounded.

The price to pay for the use of this simplified algorithm is that the α needed for convergence will be at least as large as it would be otherwise, but generally larger.

In the case at hand, the next element in the coding sequence is chosen by searching for the code-vector nearest the residue of the previous iteration. This will be the "right" decision if and only if the corresponding $f_k(P_\alpha)$ contains the point in question. If we want this to be always true, we must have $f_k(P_\alpha)$ to contain all points which have v_k as its nearest code-vector. We now need the concept of the *Voronoi cell*.

The Voronoi Cell

Given a set $S = \{s_i \in \mathbb{R}^N\}$, we define the Voronoi cell of point s_i as being the subset $V_i = \{x \in \mathbb{R}^N\}$ of points which are at least as close to s_i as to any other s_j [22, 31]. In other words,

$$x \in V_i \Leftrightarrow d(x, s_i) \leq d(x, s_j), \forall j$$

As an example, Fig. 13a depicts the Voronoi cells of a 2-D set of points. For those points which are inside the convex hull of S, the cells are convex polytopes. For those which are on the surface of the convex hull, the cells are the union of an infinite cone or cylinder with a convex polytope, always forming an infinite convex region. When referring to Voronoi cells related to codebooks, we mean the intersection of the "real" Voronoi cells of the set $\{w_k = v_k/(1-\alpha)\}$ with its convex hull, as depicted in Fig. 13b.

Convergence of the Nearest Point Algorithm

We can now state the conditions under which we will have convergence

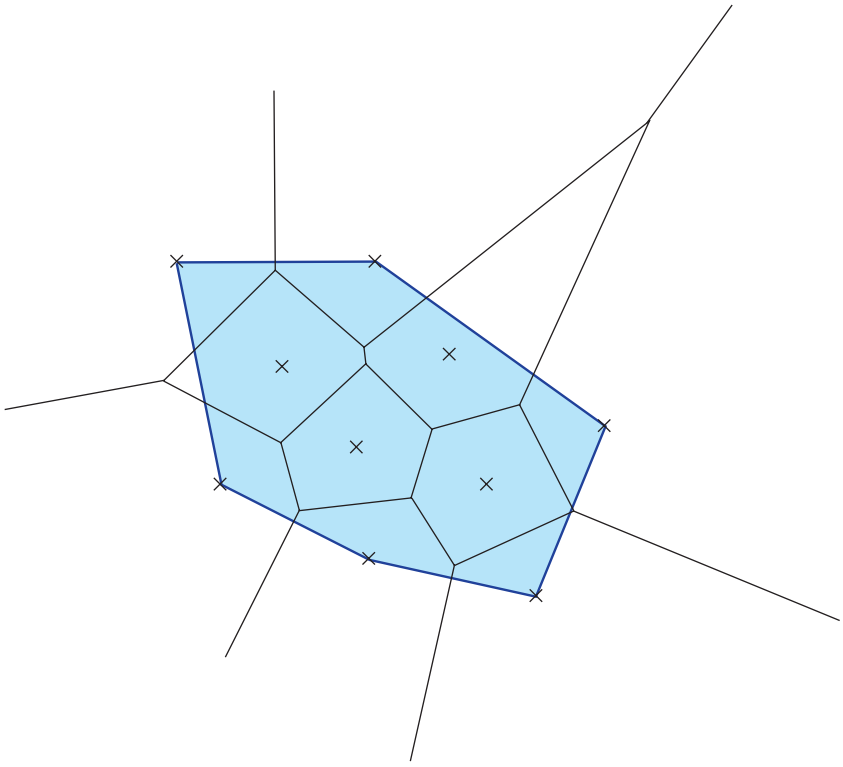


Figure 13. Voronoi cells: (b) Convex hull.

of the greedy algorithm, i.e., we must have both

$$F(P_\alpha) = P_\alpha \qquad (8)$$

and

$$f_k(P_\alpha) \supseteq \left\{ V_k \bigcap P_\alpha \right\} \qquad (9)$$

where V_k is the Voronoi cell of w_k. In fact, the second condition implies the first, since

$$\bigcup_k \left\{ V_k \bigcap P_\alpha \right\} = P_\alpha$$

This way, selecting the w_k nearest to x implies that $x \in \{V_k \bigcap P_\alpha\}$, thus $x \in f_k(P_\alpha)$; this is sufficient for the algorithm to converge.

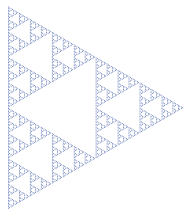
Observe that the minimum α needed to obtain Eq. 9 is at least as large as the critical value α_c defined in the section *Bounds on* α *for* $\Lambda = P_\alpha$, needed for convergence of the basic algorithm. So, in order to use the greedy algorithm *and* have the minimum possible α, we must look for codebooks where Eq. 8 implies the relationship 9.

Performance Analysis

Since α-expansions are a generalization of the uniform scalar quantizer, we wonder whether we can find a codebook with a better rate x distortion performance. After all, it would result in a moot theory indeed if that were not the case.

In the remainder of this section, we will deal with this problem. We will show that, for high rates, the best α-expansion is the one equivalent to uniform scalar quantization (see the section *Binary Expansions as a Special Case*). However, in low bit-rate image transform coding, we have a majority of small magnitude coefficients; in this case, α-expansions using certain multidimensional codebooks can be a better choice, provided that an efficient coding method for the zero values is devised (see the sections *Low bit-rate quantization* and *The EZW Algorithm*).

The procedure used to compare codebooks' performances was to calculate the average MSE for the α-expansions of large numbers of data points. This was done implementing the greedy algorithm of the section *The Nearest Point Algorithm*. The proper α and β values (see the section *The Search for Convergence*) for each codebook were found by trial and error, except for some simple codebooks where those parameters could be readily determined theoretically. Basically, we run the algorithm described in the section *The Nearest Point Algorithm* for a large number of data points, while controlling the error behavior; if



at any moment the error magnitude grows larger than β, either one or both parameters are smaller than the needed critical values. When proper values for α and β are found, we proceed to performance measurements.

The results depend strongly on the spatial distribution of the data sets used. The data sets result from the clustering of coefficients into vectors of the appropriate dimension, i.e., that of the codebook. Two kinds of data sets were used: uniformly distributed points inside the N-dimensional sphere of radius 1, and vectors generated by the clustering of actual wavelet transforms of natural images.

Rate and Distortion Measurements

Remember that each coded vector represents N coefficients. In order to properly compare codebooks with possibly different dimensions, we must determine the per-coefficient distortion and rate.

Clearly, both distortion and rate are dependent on the number of iterations, n, through the algorithm. At each step, average total distortion decreases (since $|r_n|$ is bounded by $C\alpha^n$) and the total rate increases simply as $R_T = n.\log_2(M)$ bits per symbol, where M is the cardinality of the codebook.

Thus, we have that the per-coefficient rate, in bits, is

$$\mathrm{R} = \frac{n.\log_2(M)}{N} = n.\log_2(M^{\frac{1}{N}}).$$

Naturally, the actual total distortion depends on the spatial distribution of the data set; however, the asymptotic behavior of $|r_n|$ depends only on the form of the codebook: for large n, we will have that $|\bar{r}_n| = C_0\alpha^n$, where C_0 is some positive constant. Thus, for the total distortion, we will have $D_T = C_0\alpha^{2n}$; the per-coefficient average distortion will be $D = \frac{C_0}{N}\alpha^{2n}$. Average per-coefficient distortion as a function of rate will thus be

$$\mathrm{D(R)} = \frac{C_0}{N}\left(\alpha^{\left(\frac{1}{\log_2\left(M^{\frac{1}{N}}\right)}\right)}\right)^{2R} \qquad (10)$$

Eduardo A. B. da Silva was born in Rio de Janeiro, Brazil, in 1963. He received the Engineering degree in electronics from Instituto Militar de Engenharia (IME), Brazil, in 1984, the M.Sc. degree in electronics from Universidade Federal do Rio de Janeiro (COPPE/UFRJ) in 1990, and the Ph.D. degree in electronics from the University of Essex, England, in 1995. In 1987 and 1988 he was with the Department of Electrical Engineering at Instituto Militar de Engenharia, Rio de Janeiro, Brazil. Since 1989 he has been with the Department of Electronics Engineering (the undergraduate department), UFRJ. He has also been with the Department of Electrical Engineering (the graduate studies department), COPPE/UFRJ, since 1996. He has been head of the Department of Electrical Engineering, COPPE/UFRJ, Brazil, in 2002. He won the British Telecom Postgraduate Publication Prize in 1995, for his paper on aliasing cancellation in subband coding. He is also co-author of the book *Digital Signal Processing—System Analysis and Design*, published by Cambridge University Press. He is presently serving as associate editor of the *IEEE Transactions on Circuits and Systems—I*. His research interests lie in the fields of digital signal and image processing, especially signal coding, wavelet transforms, mathematical morphology and applications to telecommunications.

So, to minimize the distortion for large rates, we must have the minimum possible $\alpha^{\frac{1}{\log_2\left(M^{\frac{1}{N}}\right)}}$. Since we know that $\alpha \geq M^{-\frac{1}{N}}$ (see the section *Bounds on* α *for* $\Lambda = \mathrm{P}_\alpha$), we may write $M^{\frac{1}{N}} = \frac{k}{\alpha} \geq \frac{1}{\alpha}$, with $k \geq 1$. Analyzing the function $\alpha^{\left(\frac{1}{\log_2\left(\frac{k}{\alpha}\right)}\right)} = 2^{\frac{\log_2 \alpha}{\log_2 k - \log_2 \alpha}}$, given the restriction $k \geq 1$, we find its minimum at $k = 1$. In this case, from Eq. 10, $D(R) = \frac{C_0}{N} 2^{-2R}$. For the codebooks corresponding to the binary expansion, $M = 2^N$ and $\alpha = 1/2$, hence $k = 1$. Thus, we conclude that for *large rates*, these codebooks are optimal for α-expansions. Since $M^{\frac{1}{N}} = 2$, the rate x distortion performances of these codebooks do not change with their dimension, so in this case scalar and vector quantization are equivalent.

Experiment Setup

For the experiments described below, we determined the resulting average distortion D (MSE) in decibels, that is, $10 \log_{10}(D)$, and the accumulated data rate R in bits, up to 20 iterations of the α-expansion algorithm. Details about the codebooks and datasets are given below.

Codebook generation—As we have seen, convergence will be faster for smaller α. Also, for given M and N, codebooks whose vectors are more uniformly spread in $\mathbb{R}^N$ tend to have smaller α_c. Note that we have not defined "spread" here; we are using the term in a somewhat intuitive sense. The problem is reminiscent of sphere packings, several variations of which are tackled by Conway and Sloane in [31]; some related results can also be found in [32, 33].

For these experiments, we have used a few classes of codebooks; we sometimes refer to them by the name of the polytopes whose vertices we are using as codevectors:

- The binary expansion-equivalent codebook, on several different dimensions: square, cube, tesseract, and so on (including the 1-dimensional "square", equivalent to uniform scalar quantization); we refer to them as the "binary codebooks".
- Some 2-D regular polygons;

Décio A. Fonini Jr. received the M.S. degree in electrical engineering from the Instituto Militar de Engenharia, Rio de Janeiro, 1993, and is presently a post-graduate student at the Universidade Federal do Rio de Janeiro. He has developed simulation software for the Brazilian Army and is now with Upknowledge Oy, a company headquartered in Finland, doing worldwide training and consulting related to Internet and cellular technologies.

Marcos Craizer was born in Rio de Janeiro, Brazil, in 1962. He received the mathematics degree from Universidade Federal do Rio de Janeiro (UFRJ), Brazil, in 1983, and the Ph.D. degree in mathematics from Instituto de Matemática Pura e Aplicada (IMPA), Brazil, in 1989. He is a professor in the mathematics department of the Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) since 1988. His research interest lies in the field of image processing, especially image compression, mathematical morphology and applications of partial differential equations to images.

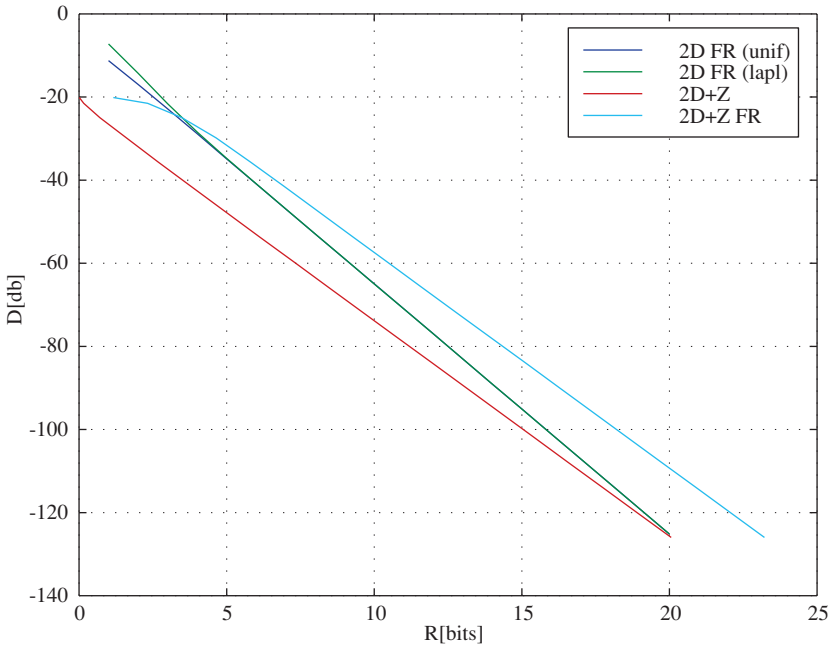
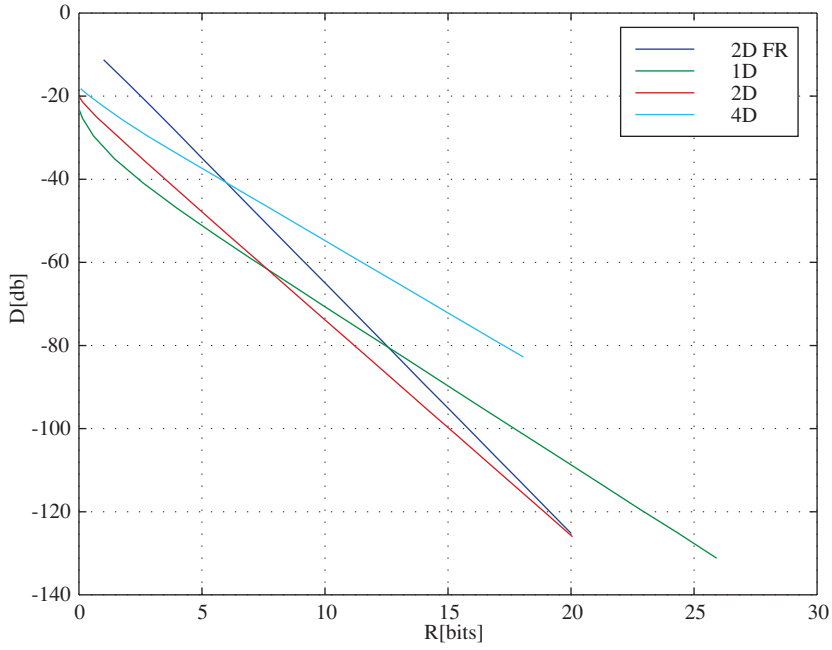


Figure 14. Graph I.

- Shells of some best known sphere packings [31]. For example, *D*4 is the 4-D codebook with the 24 vectors of the first shell of the best known sphere packing in 4-D.

Most modern methods use uniform scalar quantizers, except for the region near the origin, where a double-sized quantizer bin is used. The reasoning behind this behavior is that most of the points to be coded have small magnitude; the oversized bin near the origin helps to net still more points to be coded as zeros. Remember that most of the significant information about the image is concentrated on the large magnitude coefficients. Accordingly, we tested the codebooks listed above with an additional zero-vector as a codevector.

Figure 15. Graph II.



Dataset generation—The datasets used for the experiments were of three kinds:

- Uniformly distributed (in volume) inside the *N*-dimensional sphere of radius 1; those were mostly used for the convergence tests to find suitable values for α and β, but also for performance comparison of codebooks.
- Laplace-distributed coefficients. Those resemble more accurately the datasets obtained by clustering the coefficients of wavelet transforms of natural images.
- Clusterings of the coefficients of real wavelet transforms.

All datasets were normalized so that $|x|_{\max} = 1$.

The coding of the zeros—Usually, in image and video compression, zeros and other values are coded with different methods, in order to take advantage of the overwhelming majority of zeros generated by the quantization stage; that is why zerotrees and the like lend those methods such good performances (see the sections *The EZW Algorithm* and *Low bit-rate quantization*). So, the resulting bitstream is composed basically of symbols which represent the non-zero values. In order to simulate this behavior, in most experiments we have considered only the rate due to the coding of non-zero values. This is justified because we are assuming

that an efficient method will be used for encoding the zeros, resulting in negligible added rate. In the results that follow, the case where the zeros are coded in the same way as other values is shown for comparison; this is indicated as "FR" (full rate) in the graphs.

Comparison of Codebook Performances

The performance of the uniform scalar quantizer on the uniformly distributed dataset is taken as baseline for comparison; it is seen as the dark-blue, straight line on the graphs.

The graph shown in Fig. 14 depicts the $D \times R$ curves for the following cases:

2D FR (unif): A uniformly distributed dataset, coded with the 2D binary codebook; full rate is shown.

2D FR (lapl): Ditto, for a Laplace-distributed dataset.

2D + Z: Laplace-distributed dataset, 2D binary codebook plus the origin as a codevector.

2D + Z FR: Ditto, full rate shown.

Some interesting features can be perceived from the graph:

- The binary codebook, that is equivalent to scalar uniform quantization, has the best asymptotic performance; this is in accord with the results from the section *Rate and Distortion Measurements*.
- The statistical distribution of the dataset influences the results for the first steps, but the codebook dictates the performance as the residues tend to a distribution which depends on the codebook; that is why the curves for the uniform and laplacian distributions merge for high rates.
- The performance improvement obtained by the efficient encoding of zeros is clearly shown by the 3rd and 4th curves ("2D + Z" and "2D + Z FR").

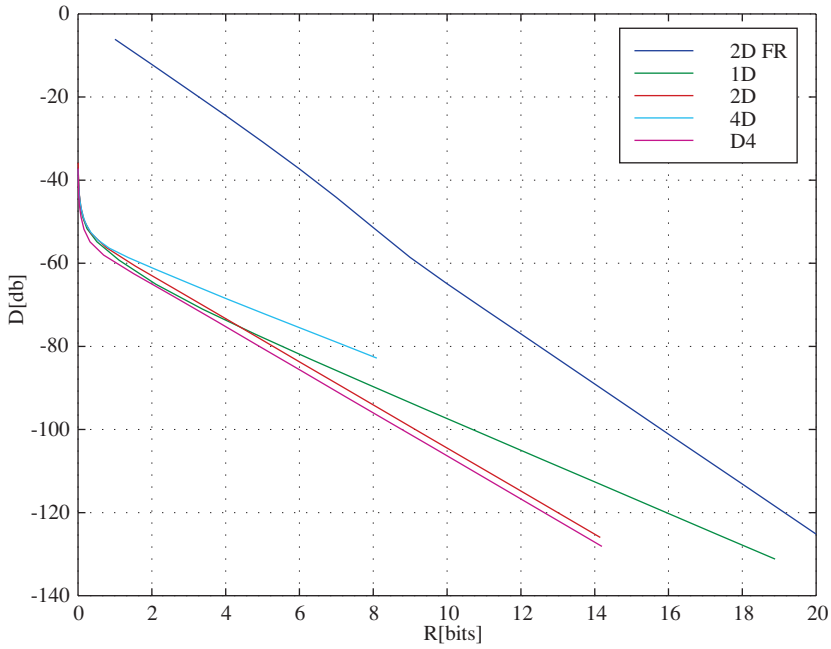
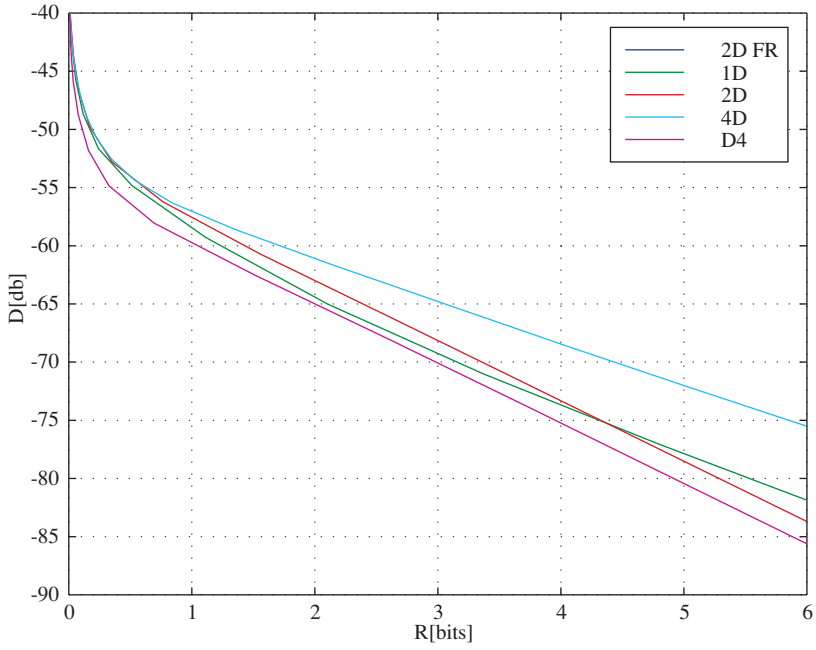
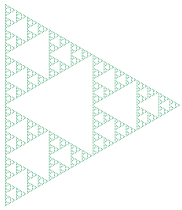


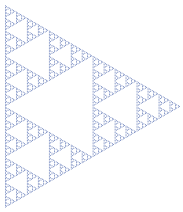
Figure 16. Graph III.

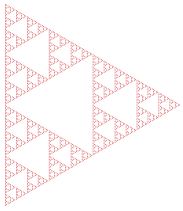
Graph II (Fig. 15) depicts the $D \times R$ curves for the 1-, 2- and 4-D binary codebooks (plus the origin), applied to a Laplace-distributed dataset. We note that for laplacian distributions, even in the binary case, the codebook's dimen-

Figure 17. Graph IV.









sion may have a strong influence on the $D \times R$ performance.

Graphs III and IV (Fig. 16 and 17) show the results obtained on a dataset composed by the wavelet coefficients of the transformed Lena (512 x 512) image (a classical image in this field of research). Besides the same binary codebooks as before, the 4-D binary and the D4 codebooks (plus the origin) are also used. The results are similar to those obtained on the Laplace-distributed dataset, but the coefficient distribution here is much more concentrated around the origin, so the advantages of including the zero-vector are much more pronounced. As can be seen from the graphs, the D4 codebook has a very good performance on a large range of data rates. Thus, although the binary codebooks present the best asymptotic behavior, this shows that for practical data rates other kinds of codebooks can outperform them.

Thus, α-expansions as a generalization of the usual binary expansion do lead to a rate x distortion performance improvement. This explains the results obtained in [25–27] where a generalization of the EZW [2] algorithm for vectors is shown to yield superior performance. We have also conducted similar experiments with the MGE [5] algorithm, for a bit-rate of 0.5 bpp. The PSNR for the scalar case was 36.68 dB, while the modified MGE with the Λ_{16} codebook (the first shell of the best known sphere packing in 16 dimensions [31]) resulted in a PSNR of 37.11 dB, with $\alpha = 0.6$.

Conclusions

We have discussed successive approximation quantization methods. The classical results of Equitz and Cover have been analyzed, as well as the more recent results of Mallat and Falzon regarding low bit-rate encoding of transform coefficients. We then have shown some practical image encoding methods based on successive approximations. Additionally, we developed the theory of α-expansions, a generalization of SA quantization. This theory has been used here to explain some published results [25–27]. We finish the article by showing that α-expansions do lead to improvements in rate x distortion performance of known SA-based image compression methods. This requires the use of suitable N-dimensional codebooks, which opens new avenues for research.

Acknowledgments

We would like to thank Lara C. R. L. Feio for implementing and providing the results of the α-expansions variation of the MGE algorithm.

References

[1] P. S. R. Diniz, E. A. B. da Silva, and S. L. Netto, *Digital Signal Processing: System Analysis and Design*. Cambridge University Press, 2002.

[2] J. M. Shapiro, "Embedded Image Coding Using Zero-Trees of Wavelet Coefficients", *IEEE Transactions on Signal Processing*, vol. 41, pp. 3445–3462, December 1993.

[3] A. Said and W. A. Pearlman, "A New, Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, June 1996.

[4] D. Taubman, "High Performance Scalable Image Compression with EBCOT", *IEEE Transactions on Image Processing*, vol. 9, no. 7, July 2000.

[5] Tse-Hua Lan and A. H. Tewfik, "Multigrid Embedding (MGE) Image Coding", *Proceedings of the 1999 International Conference on Image Processing*, Kobe.

[6] JPEG2000 Verification Model 5.3, ISO/IEC JTC1/SC29/WG1 (ITU/T SG28), 1999.

[7] W. H. R. Equitz and T. M. Cover, "Successive Refinement of Information", *IEEE Transactions on Information Theory*, vol. 37, no. 2, March 1991.

[8] D. Taubman and A. Zakhor, "Multirate 3-D Subband Coding of Video", *IEEE Transactions on Image Processing*, vol. 3, no. 5, September 1994.

[9] D. Taubman and A. Zakhor, "A Common Framework for Rate and Distortion Based Scaling of Highly Scalable Compressed Video", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 46, no. 4, August 1996.

[10] A. K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, New Jersey: Prentice Hall, 1989.

[11] K. Sayood, *Introduction to Data Compression*, 2nd Edition. San Francisco, California: Morgan Kaufmann Publishers, 2000.

[12] N. Jayant, *Image Coding Based on Human Visual Models in Image Processing*. Academic Press, 1994.

[13] Andrew B. Watson (Ed.), *Digital Images and Human Vision*. MIT Press, 1993.

[14] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Compression Standard*. New York: Kluwer Academic Publishers, 1992.

[15] Coding of Moving Pictures and Associated Audio for Digital Storage Media up to 1.5Mbit/s, ISO/IEC JTC1/CD 11172, 1992.

[16] Generic Coding of Moving Pictures and Associated Audio, ISO/IEC JTC1/CD 13818, 1994.

[17] Coding of Moving Pictures and Audio, ISO/IEC JTC1/SC29/WG11/CD 14496, 1997.

[18] T. Berger, *Rate Distortion Theory: A Mathematical Basis for Data Compression*. Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1971.

[19] J. M. Shapiro, "An Embedded Wavelet Hierarchical Image Coder", *Proceedings of the 1992 ICASSP Conference*, vol 4, pp. 657–660, San Francisco, March 1992.

[20] S. G. Mallat, *A Wavelet Tour of Signal Processing*. San Diego, California: Academic Press, 1998.

[21] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression*. Englewood Cliffs, New Jersey: Prentice Hall, 1990.

[22] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. New York: Kluwer Academic Publishers, 1991.

[23] T. M. Cover and J. A. Thomas, Elements of Information Theory. New York: John Wiley and Sons, Inc., 1991.

[24] S. Mallat and F. Falzon, "Analysis of Low Bit Rate Image Transform Coding", *IEEE Transactions on Signal Processing*, vol. 46, no. 4, April 1998.

[25] E. A. B. da Silva, D. G. Sampson and M. Ghanbari, "A Successive Approximation Vector Quantizer for Wavelet Transform Image Coding", *IEEE Transactions on Image Processing*, Special Issue on Vector Quantization, vol. 5, no. 2, pp. 299–310, February 1996.

[26] E. A. B. da Silva, *Wavelet Transforms for Image Coding*. Ph.D. Dissertation, University of Essex, United Kingdom, June 1995.

[27] M. Craizer, E. A. B. da Silva, and E. G. Ramos, "Convergent Algorithms for Successive Approximation Vector Quantization with Applications to Wavelet Image Compression", *IEE Proceedings—Vision, Image and Signal Processing*, vol. 146, no. 3, pp. 159–164, July 1999.

[28] M. F. Barnsley, *Fractals Everywhere*. Academic Press, Inc., 1988.

[29] http://carbon.cudenver.edu/~hgreenbe/glossary/tours/linearalgebra.html.

[30] D. A. Fonini Jr., *Quantization with Alpha-Expansions* (Internal Report), http://lps.ufrj.br/IR/Fonini/alpha-exp.pdf.

[31] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*. Springer-Verlag, 1988.

[32] L. Lovisolo, E. A. B. da Silva, "Uniform Distribution of Points on a Hyper-Sphere with Applications to Vector Bit-Plane Encoding", *IEE Proceedings—Vision, Image and Signal Processing*, vol. 148, no. 3, pp. 187–193, June 2001.

[33] E. B. Saff, A. B. J. Kuijlaars, "Distributing Many Points on a Sphere", *Mathematical Intelligencer*, vol. 19, no. 1, pp. 5–11, 1997.

[34] M. Craizer, D. A. Fonini Jr. and E. A. B. da Silva, "Alpha-Expansions: A Class of Frame Decompositions", to appear in *Applied and Computational Harmonic Analysis*.

[35] M. Craizer, D. A. Fonini Jr. and E. A. B. da Silva, "Quantized Frame Decompositions", in *Curve and Surface Fitting: Saint-Malo 99*. Nashville, Tennessee: Vanderbilt University Press, pp. 153–160, 2000.

[36] P. L. Zador, *Development and Evaluation of Procedures for Quantizing Multivariate Distributions*. Ph.D. Dissertation, Stanford University, 1963.