SPEEDING UP FRACTAL IMAGE CODING BY COMBINED DCT AND KOHONEN NEURAL NET METHOD

J.-M. Mas Ribés, B. Simon and B. Macq

Laboratoire de TéLécommunications
Université catholique de Louvain
2, Place du Levant - BE-1348-Louvain La Neuve

ABSTRACT

Iterated Transformation Theory (*ITT*) coding, also known as Fractal Coding, in its original form, allows fast decoding but suffers from long encoding times. During the encoding step, a large number of block best-matching searches have to be performed which leads to a computationally expensive process. We present in this paper a new method that significantly reduces the computational load of *ITT* based image coding. Both domain and range blocks of the image are transformed into the frequency space. Domain blocks are then used to train a two dimensional Kohonen Neural Network (*KNN*) forming a code book similar to Vector Quantization coding. The property of *KNN* (and Self-Organizing Feature Maps in general) which maintains the topology of the input space allows to perform a neighboring search so as to find the piecewise transformation between domain and range blocks.

1. INTRODUCTION

Image compression techniques try to exploit some form of signal redundancy or irrelevance. A signal representation is said to be redundant if another, more compact, representation can be found without introducing any distortion on the signal [1]. On the other hand, a coder that exploits irrelevance on a signal introduces distortion that may, or not, be perceived by a human observer [2].

Fractal Image Coding (*FIC*) based on Iterated Transformation Theory [3,4], as well as Vector Quantization (*VQ*) or DCT-based techniques, are considered as second generation methods for Image Coding. According to Shannon's Rate Distortion Theory, block and vector based coding methods are superior to scalar coding for achieving the theoretical high compression rates [5,6]. *FIC* exploits a signal property called piecewise self-similarity [7] to remove redundancy and irrelevance from the signals of interest. A signal is said to possess self-similarity if parts of it in some sense resemble or are similar to the whole signal or other parts of it [8]. Therefore, if we can find an affine contractive mapping, defined by a set of affine transformations on the support of the image itself, we will be able to reconstruct the image as the fixed point of the mapping.

In this paper we present the combined application of Discrete Cosine Transforms and Kohonen Neural Nets, as a particular case of Self-Organizing Feature Maps (*SOFM*), to reduce the time complexity in fractal block coding, which is $O(N^2)$ time complex for N domain blocks. In our approach we build a feature code book mapped onto a two-dimensional Kohonen neural network. *SOFM* have successfully been used to reduce time complexity in Fractal Image Coding [9,10]. Nevertheless, our method differs on an important point: we drastically reduce the feature code book size by using some coefficients of DCT transformed blocks.

In the first step of our approach, the *KNN* is built upon vectors obtained after Discrete Cosine Transform of the image domain blocks. These vectors are mapped onto the *KNN* obtaining a set of domain block classes. Each range block, after transformation, is then mapped to a neuron (code word) and from there to a domain block class. This class is searched for best-matching between range and domain blocks. The self-organizing properties of *KNN*'s allow searching for a better solution, if necessary, among the domain block classes associated to neighboring neurons of the initial match.

This paper is organized as follows. In Section 2 we present the background literature on Iterated Transform Theory, Discrete Cosine Transform and Kohonen Neural Nets. In Section 3 we then describe the combined method proposed for domain block classification in *FIC*. Finally we summarize our major results and outline the major improvements on the complexity and computation time.

2. BACKGROUND THEORY

2.1 Iterated Transform Theory

Iterated Transformation Theory provides a way to represent an element in a general metric space by a contractive transformation defined on such space. In image coding, the element, or fixed point, is the target to be encoded. The transformation is the code for the image, and to achieve compression, its representation has to be simpler than the original image.

Operating a contractive mapping on two arbitrary points of the space X (image space) with the metric d it always causes the two points to approach. Banach's fixed point theorem states that a contractive mapping T on a complete metric space X defines a unique point x_t invariant with respect to T and is the limit of applying n times T.

An Iterated Function System (*IFS*) is a collection of contractive affine transformations on a complete metric space. Banach's

fixed point theorem can be generalized when applied to *IFS*: each *IFS* has a unique attractor which is the union of the transformations of the attractor.

This approach requires some modifications to be applied to *FIC*. For coding, the class of mappings is further constrained by restricting each $T \in \mathbf{T}$ to be the sum of partial affine transformations T_i, each of which operates on a range block indexed by i. For each i, T_i is only allowed to take a domain block, indexed by j, to which an isometry operator can be applied, map it into range block i and set the rest of the image to zero [3] (see **equation 1**[1]).

$$T(x) = \sum_i T_i(x) = \sum_i S_{I(i)}\left(\alpha_i \cdot x_{(i)} + a_i\right) \qquad (1)$$

Coding of an image $x \in X$ is equivalent to finding a partial mapping T_i for each range block of the image. Each range block is then coded as a gain α_i, a translation block a_i, an isometry operator $S_{I(i)}$ and the translation parameters for range block j.

For decoding, the fractal code of T is iterated on any initial image $z \in X$ until the distortion between two consecutive images is not noticeable.

2.2 Discrete Cosine Transform

The Discrete Cosine Transform has successfully been used for Fractal Image Coding in the past. On a first approach, domain and range blocks are transformed onto the DCT domain [12,13]. But since the inverse DCT is performed for every image block at decoding time, the decoding stage is not as rapid as fractal compression in the spatial domain. A later approach [14] proposed to use the energy packing properties of the DCT to improve existing multidimensional nearest-neighbor search algorithms applied to Fractal Image Coding [15] and avoiding the inverse DCT on decoding.

The Discrete Cosine Transform, as an orthogonal transform, performs a change of basis. The transformed coefficients result from the projection of a vector onto an orthogonal basis defined by the transformation matrix. In every block of an image we can consider the energy[2] as being distributed over all pixels. The DCT is a sub-optimal orthogonal transform which highly decorrelates the data in a block. That is, it packs most of the energy in a few number of transformed coefficients.

The main interest of using Discrete Cosine Transforms in our application is its energy packing properties. Being the DCT coefficients ordered in a block of size $N1 \times N2$, the DC value[3] is placed in the upper-left corner. The higher vertical frequencies are placed in higher line indexes, and the horizontal frequencies are placed in a similar way. When the DCT coefficients are scanned in "zigzag" order, the lower the index is, the higher its variance and the energy it contains [16]. This property is adequate for pattern recognition based on feature selection [17].

[1] $x_{(i)}$ stands for domain block j used to code range block i.

[2] In image processing, the energy of a block is defined as the squared value of every pixel.

[3] Transformed mean value of block luminance.

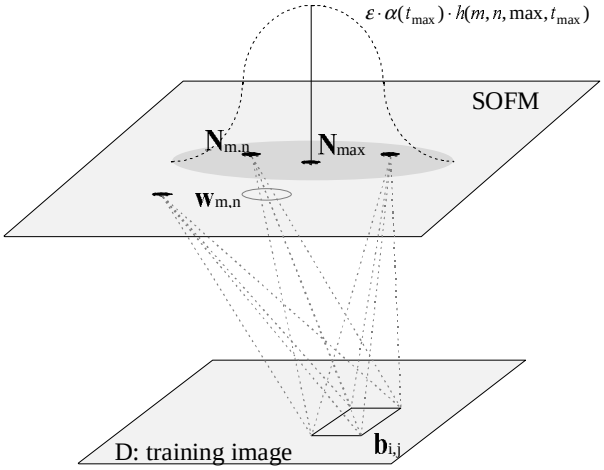

Figure 1: Training of a *KNN* for building the code book. The training set $\mathbf{b}_{i,j}$ is obtained by taking blocks from the image **D**. The learning step (adjustment of $\mathbf{w}_{m,n}$) of a neuron m,n depends on its distance from $\mathbf{N}_{max}$ and on a time decreasing function for the convergence of the *KNN*.

The method retains those features, DCT coefficients, with high variances and drops the rest, which are less useful for classification purposes.

2.3 Kohonen Neural Nets

The Kohonen Neural Net belongs to the class of Self-Organizing Feature Maps (*SOFM*) [18,19,20]. It is an Artificial Neural Net (*ANN*) in which cells are tuned to various input signal patterns or classes of patterns through an unsupervised learning process. The self-organizing process can be considered as a mapping of the probability density of a high dimensional input space onto a low dimensional space (typically 1, 2 or 3-dimensional space) where output nodes corresponding to nearby input patterns are topologically close. The aim of *KNN* is to cluster the input data in such a way that similar inputs are classified in the same cluster.

The training phase of a *KNN* belongs to the class unsupervised competitive learning algorithms. Unsupervised since the algorithm does not require a reference value for the *ANN*, as opposed to other learning algorithms such as back-propagation. Competitive learning algorithm because all neurons receive the same set of inputs from the input layer. The neuron with higher activation factor (similarity measure between the weights and the input block) makes neighboring neurons to adapt their weights in that direction (see **Figure 1**, for further details see [19]).

3. COMBINED METHOD FOR FRACTAL IMAGE CODING

In this section we present an algorithm based on the combination of DCT transformed blocks and Kohonen Neural Nets to improve encoding time for Fractal Image Coding. This new method speeds up coding by efficiently classifying image

blocks and thus improving the best-matching search. Since the *FIC* representation of the image remains the same as when performing exhaustive search, the decoder complexity does not increase.

1. Build a domain pool that will be used to train the *KNN*. This domain pool is made up of vectors obtained after pre-processing of the domain blocks. The first step is to down-sample domain blocks to size (b_x',b_y')[1]. The resulting blocks are normalized according to **Equation 2**[2]. For each normalized block generate 8 isometries[3] and apply a DCT to all of them. Vectors in the domain pool are created from some of the AC coefficients[4]. This pre-processing of domain blocks removes the unnecessary information for *FIC* contained in domain blocks: the constant component and the range of pixel values.

$$d_i'(x,y) = \frac{d_i(x,y) - \overline{d_i}}{\max(d_i) - \min(d_i)} \tag{2}$$

2. Build the code book in the *KNN* using the normalized vectors from the domain pool obtained in step 1 as a training set. The *KNN* is initialized with random values. During the learning phase, vectors from the domain pool are presented to the network *n* times or until the map converges (infinitesimal changes in the weights of the *KNN*).

3. Establish the correspondence between the vectors in the domain pool and a neuron in the *KNN*. Each neuron keeps a list of associated vectors, closest to the pattern stored in their weights, the domain image block and the isometry operator indicator that generated the vector.

4. Create a non-overlapping partition with blocks of size (b_x,b_y) on the domain image to be coded. Transform each range block $\mathbf{b}_{i,j}$ of the partition as defined in step 1. If we allow range block splitting, these blocks may need to be down or up-sampled to size (b_x',b_y'). Establish the correspondence with the neuron $\mathbf{N}_{k,l}$ which stores the closest pattern to $\mathbf{b}_{i,j}$ in its weights.

5. Establish the correspondence between every range block with one of the domain blocks from the image. Perform a best-matching search among the domain blocks associated to the activated neuron $\mathbf{N}_{k,l}$. If a minimum quality criteria is not held, one can search in the classes associated to neighboring neurons. In case the quality criteria is not satisfied, the range block can be split[5] into four blocks of the same size[6] and code each one of them separately (repeat steps 4 and 5). Once a best-matching is found, assign to block $\mathbf{b}_{i,j}$ the fractal code of the transformation (see **Section 2.1** and **Equation 1**).

This algorithm as exposed, spends most of its computational time on building a *KNN* for each image we encode. Nevertheless there are other solutions to reduce this time. A first solution is to use a pre-defined general purpose *KNN*. This will reduce the classification accuracy but on the other hand does not require the time expensive learning phase. One could also start with an existing code book and adapt it to the characteristics of the image to be coded. The use of these alternatives is out of the scope of this paper.

4. RESULTS AND FURTHER WORK

We present in this section numerical results obtained in the software simulation of the combined *KNN* and *DCT* Fractal Image Coder. Tests were run on digital gray images of size 512×512 pixels and quantized to 8 bits per pixel. Our goal in this section is to prove that our classification of domain blocks is accurate and if performs good when applied to *FIC*. For analysis of the results, we will compare our algorithm with one performing exhaustive best-matching search in several aspects: encoding time, classification accuracy and image quality.

Coded image quality is measured in terms of *PSNR* between the original and the coded image. The *PSNR* can be seen as the ratio of the dynamic range of the signal and the average energy in the error between the original and the reconstructed image.

On the other hand, to measure domain blocks classification accuracy we will use a *splitting factor* (*Sf*) criteria. We are interested in a classification method for domain blocks specifically oriented to *FIC*. The best way to evaluate this method is by comparison on the amount of splitting required to code an image with a constant quality level. **Equation 4** formalizes the *splitting factor* of an image.

$$Sf = 1 + \frac{1}{lvl} \cdot \log_k \left[\frac{\#b_{i,j} \in D}{\frac{D_x}{b_x} \cdot \frac{D_y}{b_y} \cdot k^{lvl}} \right] \tag{4}$$

where *lvl* is the maximum level of range block splitting, *k* is the number of new blocks generated when split, $\#b_{ij} \in D$ is number of range blocks encoded and the denominator in the logarithm is the maximum number of range blocks.

The *splitting factor* indicates how accurate is our classification. The higher the *Sf*, the worse the classification is since no matching is found.

We have run different simulations to study the influence of the different parameters on the algorithm, in terms of image quality, computational speed and classification accuracy. For the *FIC* coder we use three different parameter sets (see **Table 1**).

On the other hand, for the *KNN* we run the simulations varying the input space dimension between 5, 9 and 14, and a two dimensional array of neurons of sizes 16×16 and 32×32.

---

[1] These values will depend upon the maximum size of range blocks and whether we allow splitting or not.

[2] $\overline{d_i}$ is the average pixel value of domain block d_i, $\max(d_i)$ and $\min(d_i)$ are the maximum and minimum pixel values of d_i.

[3] See [4] for a complete description.

[4] The number of AC coefficients corresponding to horizontal and vertical frequencies should be the same.

[5] If the predefined maximum level of splitting is reached, the best matching is chosen ignoring the quality criteria.

[6] Quad-tree partitioning.

	(b_x,b_y)	$(2b_x, 2b_y)$	(b_x', b_y')	*lvl*
Set 1	(8,8)	(16,16)	(8,8)	0
Set 2	(16,16)	(32,32)	(8,8)	2
Set 3	(32,32)	(64,64)	(8,8)	3

Table 1: Different parameter sets related to Fractal Image Coding used in our simulations.

FIC param. KNN input KNN size	*Sf* comb.	*Sf* exhaus.	PSNR comb. (dB)	PSNR exhaus. (dB)	$T_{comb}/T_{exhaus.}$
Set 1, 9, 32	0	0	29.259	29.322	0.0123
Set 2, 9, 16	0.660	0.642	31.432	32.451	0.0195
Set 3, 9, 16	0.448	0.439	31.225	31.854	0.0976

Table 2: Summary of the best results obtained from simulations (no research)[1] coding 'Lena'.

Table 2 summarizes the best results obtained compared to a fractal encoder performing exhaustive best-matching search. After running several simulations we obtained promising results. The *splitting factor* obtained with our method is always very close to the minimum achievable performing an exhaustive search. This indicates the domain blocks are accurately classified for *FIC* since range blocks are correctly coded and no extra splitting is required to obtain the same quality. On the other hand, the encoding time is drastically reduced, firstly because the number of matchings performed is limited to domain blocks associated to the class of the activated neuron. The energy packaging property of the DCT allows to use a reduced number of AC coefficients. This significantly reduces the dimension of the input space in the *KNN* compared to other methods [9] and thus the computational load related to the *KNN*. The result is a fractal image encoder between 10 and 80 times faster than one performing exhaustive search. This time includes the learning phase of the *KNN* which represents at least 70% of the total time. There exist methods to reduce the learning phase by using a predefined network or adapting an existing one to the image being coded. These alternatives are out of the scope of this paper.

These results were obtained searching only a class of domain blocks. If a research is made among classes associated with neighboring neurons, the *splitting factor* is further reduced at the expense of a 2 to 6 times higher encoding time.

The goal of this work has been to study the feasibility of applying both DCT and *SOFM* techniques to *FIC*. From this point there is a multitude of problems that could be addressed such as, among others, adaptive segmentation, classification based on HVS or further optimization of the algorithm.

5. REFERENCES

[1] M. Rabbani and P.W. Jones, Digital Image Compression Techniques, SPIE-The International Society for Optical Engineering, Washington, 1991.

[2] S. Comes, S. Maes and M. Van Droogenbroeck, "Recent and prospective decorrelation techniques for image processing", Annales des Télécommunications, 48(7-8):340-403, Jul.-Aug. 1993.

[3] A.E. Jacquin, A Fractal Theory of Iterated Markov Operators with Applications to Digital Image Coding. PhD thesis, Georgia Institute of Technology, Aug. 1989.

[4] A.E. Jacquin, "Image coding based on a fractal theory of iterated contractive image transformations", IEEE Transactions on Image Processing, vol. 1, pp. 18-30, Jan. 1992.

[5] C.E. Shannon, "A mathematical theory of communications", Bell Systems Technical Journal, 27:379-423 and 623-656, 1949.

[6] T. Lookabaugh and R.M. Gray, "High-resolution quantization theory and the vector quantizer advantage", IEEE Transactions on Information Theory, vol. 35, pp. 1020-1033, Sept. 1989.

[7] B.B. Mandelbrot, The fractal geometry of nature, Freeman, San Francisco, 1982.

[8] M.F. Barnsley, Fractals everywhere, Academic Press Inc., San Diego, 1988.

[9] A. Bogdan and H.E. Meadows, "Kohonen neural network for image coding based on iteration transformation theory", In Proceedings of SPIE's 92 International Symposium, SPIE# 1766-39, 1992.

[10] J. Stark, "Iterated function systems as neural networks", Neural Networks, vol. 4, pp. 679-690, 1991.

[11] W.B. Pennebaker and J.L. Mitchell, JPEG Still Image Data Compression Standard, Van Nostrand Reinhold, New York, 1993.

[12] K.U. Bartel and T. Voyé, "Adaptative fractal image coding in the frequency domain", in Proceedings of the International Workshop on Image Processing, Theory, Methodology, Systems and Applications, (Budapest, Hungary), Jun. 1993.

[13] Y. Zhao and B. Yuan, "Image compression using fractals and discrete cosine transform", Electronics Letters, vol. 30, pp. 474-475, Mar. 1994.

[14] B. Wohlberg and G. De Jager, "Fast image domain fractal compression by DCT domain block matching", Electronic Letters, vol. 31, pp. 869-870, May 1995.

[15] D. Saupe, R. Hamzaoui, "Complexity Reduction Methods for Fractal Image Compression", I.M.A. Conference Proceedings on Image Processing; Mathematical Methods and Applications, Sept. 1994, J.M. Blackledge (ed.), Oxford University Press, 1995.

[16] N. Ahmed, T. Natarajan and K.R. Rao, "Discrete Cosine Transform", IEEE Transactions on Computers, vol. 23, pp. 90-93, Jan. 1974.

[17] H.C. Andrews, "Multidimensional rotations in feature selection", IEEE Transactions on Computers, vol. C-20, pp. 1045-1051, Sept. 1971.

[18] T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps", Biological Cybernetics, num. 43, pp 59-69, 1982.

[19] T. Kohonen, Self-Organization and Associative Memory, Springer-Verlag, Berlin, 3rd edition, 1989.

[20] T. Kohonen, "the Self-Organizing Map", IEE Proceedings, vol. 78, pp. 1443-1464, Sept. 1990

[1] *comb.* stands for combined method, whereas *exhaus.* does for exhaustive search.