The classification of 8×8 blocks is implemented by a 2 level classifier. All 8×8 blocks are first classified as either uniform blocks ($\mathbf{S}_3$) or nonuniform blocks ($\mathbf{S}_4 \cup \mathbf{S}_5$) by comparing $d(B_k)$ with a threshold T_2. Each nonuniform block is then approximated by a first order 2D polynomial

$$g_k(x, y) = a_k \cdot x + b_k \cdot y + c_k \quad (4)$$

The coefficients a_k, b_k and c_k are determined using the minimum mean square error rule [5]. The approximation error is then estimated as follows:

$$MSE(B_k) = \frac{1}{n^2} \sum_{(x,y) \in B_k} [f(x, y) - g_k(x, y)]^2 \quad (5)$$

If $MSE(B_k) < T_3$, $B_k \subset \mathbf{S}_4$, otherwise $B_k \subset \mathbf{S}_5$, where T_3 is a threshold. $T_1 - T_3$ are the parameters used to control compression ratio.

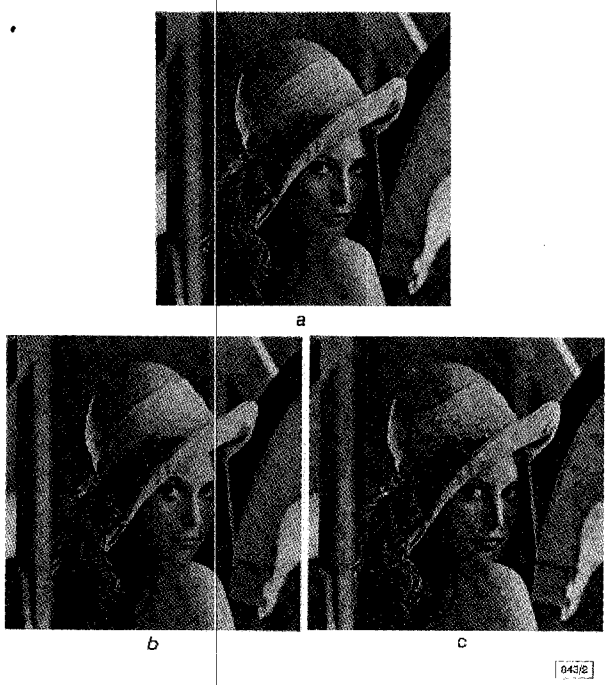

Fig. 2 *Experiment results*

a Original image
b Reconstructed image from proposed algorithm at 0.22bpp
c Reconstructed image from JPEG at 0.22bpp

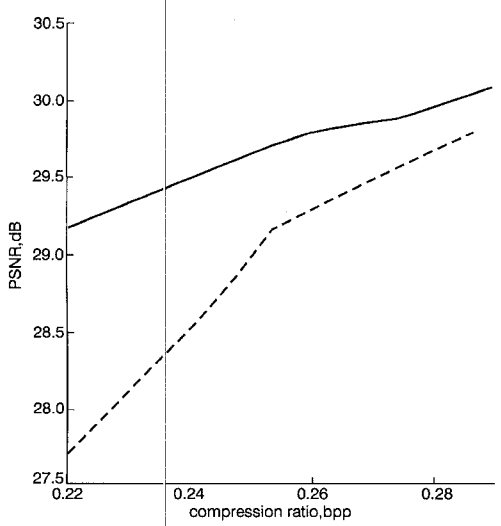

Fig. 3 *Performance of proposed algorithm compared with JPEG*

——— proposed algorithm
– – – – JPEG

Hybrid coding: The uniform blocks of 8×8, 16×16 and 32×32 are encoded using the block means to reach a high compression ratio. Each smooth block is approximated by a 2D polynomial in eqn. 4. The polynomial coefficients (a_k, b_k and c_k) are encoded. Polynomial approximation may obtain very good results when applied to highly correlated data [6, 7]. The non-smooth blocks are encoded by DCT based coding. The DCT coefficients of such a block are computed. They are then quantised and converted into a 1D sequence via the zigzag scan, as performed in JPEG. The first L low frequency coefficients are encoded using run-length coding, where L is an integer, while the remaining non-zero coefficients are represented by their values and positions.

Experimental results and conclusions: We have tested the performance of the proposed algorithm on the 'Lena' image of 512×512. Fig. 2*a* is the original image. Fig. 3 shows the performance of the proposed algorithm compared with that of JPEG in terms of peak-signal-to-noise ratio (PSNR). The reconstructed images obtained at 0.22bpp, by applying the proposed algorithm and JPEG, are shown in Fig. 2*b* and *c*, respectively. In the case of low bit rate coding, our algorithm greatly outperforms JPEG owing to hierarchical segmentation and hybrid coding. As the bit rate increases, the performances of our method and JPEG tend to approach one another since more blocks are encoded by DCT in our method, which is similar to JPEG. In terms of subjective quality, the reconstructed images from our algorithm are much better than those from JPEG. This is due to segmentation and classification based on the visual features of the HVS.

In summary, we incorporate a visual model into image segmentation and propose a segmentation based hybrid coding algorithm. The major advance is the derivation of a novel luminance masking based criterion for quadtree based segmentation. This criterion has better performance than the popular criterion of block variance. Another advance is hybrid coding, in which we combine polynomial representation (block means can be considered as zero-order polynomials) and DCT coding, and exploit their advantages efficiently. The algorithm exhibits good performance.

10 March 1998
Electronics Letters Online No: 19980568

Jiwu Huang (*Department of Electronic Engineering, Shantou University, GD 515063, People's Republic of China*)

Yun Q. Shi (*Department of ECE, New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA*)

Yun Q. Shi: corresponding author

E-mail: shi@tesla.njit.edu

References

1 WATSON, A.B.: 'DCT quantization matrices visually optimized for individual images'. SPIE: Human vision, visual processing and digital display IV, 1993, Vol. 1913, pp. 202–216
2 GONZALEZ, R.C., and WINTZ, P.: 'Digital image processing' (Addison-Wesley Publishing Co., 1987), 2nd edn.
3 LEGGE, G.E., and FOLEY, J.M.: 'Contrast masking in human vision', *J. Opt. Soc. Am.*, 1980, **70**, (12)
4 WATSON, A.B.: 'Efficiency of a model human image code', *J. Opt. Soc. Am.*, 1987, **4**, (12)
5 HUANG, J., and DAI, R.: 'Image block coding: classification method', to be published in *J. Images Graphics*, 1998
6 KUNT, M., BQENARD, M., and LEONARDI, R.: 'Recent results in high-compression image coding', *IEEE Trans. CAS*, 1987, **CAS-34**, (11)
7 DE NATALE, F.G.B., DESOLI, G.S., GIUSTO, D.D., and VERNAZZA, G.: 'Polynomial approximation and vector quantization: A region-based integration', *IEEE Trans. COM*, 1995, **COM-43**, (2/3/4)

Split decision functions in fractal image coding

R. Distasi, M. Polvere and M. Nappi

Quadtree partitions are often used in fractal image compression to achieve variable bit rates. The authors present an entropy based split decision function that can improve image quality and speed up the encoding process. Furthermore, other widespread split decision functions are discussed, along with the advantages offered by adaptive thresholds.

Introduction: In fractal image compression, an image is partitioned into a set of image blocks called ranges. A pool of larger image blocks called domains is used as a codebook from which ranges are approximated with affine mappings of the intensity value. It is common practice to use square blocks for both ranges and domains, as well as to enlarge the codebook by including all rotations and reflections of each domain. For further details, see [1, 2].

Bit rates and image quality are strongly related to range block size: large ranges are hard to approximate but lead low bit rates, while small ranges are easily approximated but yield higher bit rates.

To achieve both good fidelity and low bit rates, one possible solution is to employ partitions of variable size that tune range size to the complexity of the different image areas. The most popular partition mechanism uses a quadtree scheme [1]: a square block is recursively broken up into four quadrants until the resulting blocks are considered sufficiently simple to be approximated by a domain chosen from the codebook. This decision is usually made by comparing the value of a function of the block, the so-called split-decision function, against a given threshold. The choice of both function and threshold affects the fidelity of the reconstructed image as well as the coding time

This Letter provides a comparison between the most widespread split decision functions and proposes a split decision function based on entropy, also addressing choice of the threshold.

Split decision functions: One aim of a fractal coder is to minimise the root-mean-square (RMS) error between the original and the transformed image. To achieve this aim, the natural way of driving the partition process is to adopt RMS error as the splitting function. The classical split decision function computes the RMS error between the range R and the optimal transformed domain D^*:

$$S_1(R) = \mathrm{RMS}(R, D^*) = \frac{\|R - D^*\|}{\sqrt{n}}$$

where n is the area in pixels. Using the function S_1 implies that, before subdividing, every attempt is made to encode bigger ranges, thus leading to optimal rate-distortion curves. On the other hand, because of the unsuccessful attempts, much computation time is wasted. To avoid this, it would be better to use a function which can be computed before even trying to encode the current range block.

In [3], the authors consider a splitting function called n-fold variance that, for a generic block B (range or domain), is defined as

$$S_2(B) = n\mathrm{Var}(B) = \sum_{k=1}^{n}(B_k - \mu(B))^2$$

where $\mu(B)$ is the mean value of the intensities B_1, ..., B_n. This choice not only accelerates the encoding process but also outperforms S_1 terms of rate-distortion curves. Although S_1 should be optimal from the point of view of fidelity, S_2 is an adaptive criterion, while S_1 is not. In fact, the n-fold variance S_2 takes into account the size of the block: on average, its value on a given block is four times bigger than on a block of half its linear size. If the threshold is fixed for all levels of the quadtree, then the subdivision of bigger blocks, which are more difficult to approximate accurately, is favoured over that of smaller ones. The overall effect is a better rate-distortion curve.

Using n-fold variance with a fixed threshold is equivalent to the use of standard variance

$$S_3(B) = \mathrm{Var}(B) = \frac{1}{n}\sum_{k=1}^{n}(B_k - \mu(B))^2$$

with an adaptive threshold $T_i = 4T_{i-1}$, where T_i is the threshold at level i of the quadtree. The fact that adaptive thresholds improve image quality has also been observed in [4] for RMS based split decision functions.

Entropy is another interesting splitting function. For an 8 bit grey scale block R, it is defined as

$$S_4(R) = H(R) = -\sum_{k=0}^{255} f_k \lg(f_k)$$

where f_k is the frequency within R of the intensity value k. Like variance, entropy saves computing time by avoiding computation for unsuccessful attempts.

Regarding threshold adaptivity, there seems to be a close correlation between the split decision function scaling factor on adjacent levels of the quadtree and the optimal thresholds on those levels. In fact, for the variance criterion, $T_i = 4T_{i-1}$ yields the best rate-distortion curve, while for RMS the best curve is obtained by $T_i = 2T_{i-1}$.

The situation is different for entropy. Entropy is a convex function that reaches its maximum when the frequency distribution is uniform. Since we are dealing with 8 bit images, with 256 intensity values, the distribution inside a small block (e.g. 8×8) is likely to be noticeably less uniform than the distribution in a larger block (e.g. 32×32): the number of possible pixel values is comparable to the number of pixels in a block. This means that larger blocks tend to have higher entropy: entropy is thus intrinsically adaptive. Furthermore, being a logarithmic function, entropy does not have a scaling factor. For all these reasons, the optimal threshold on level i should depend not only on that on level $i - 1$, but also on the size of the block itself or equivalently on i. In fact. we experimentally obtained the best curve using the threshold relation $T_i = T_{i-1} + 1/i$. However, both entropy and variance share the problem of precisely tuning the threshold value to obtain the desired degree of fidelity.

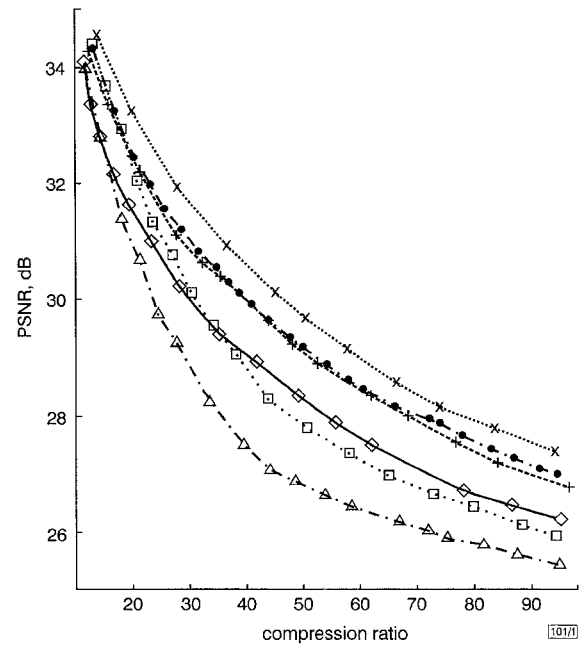

Fig. 1 *Rate-distortion curves of 512 × 512 Lena image obtained using 4-level quadtree*

—◇— entropy
– –+– – adaptive entropy
– –□– – RMS
– –×– – adaptive RMS
– ·△· – variance
– ·●· – adaptive variance

Table 1: Speed-up achieved using adaptive entropy against RMS on the 512 × 512 Lena image

CR	Entropy SNR	RMS SNR	Speed-up
12	34.6	35.2	2.28
35	30.1	30.9	1.95
50	29.1	29.7	1.83

Experimental results: Experiments were performed using 4 level quadtrees, with range block linear sizes of 32, 16, 8 and 4 pixels, on the 512 × 512 Lena image. In order not to give an advantage to any particular method, we did not employ a lossy speed-up technique such as block classification. Furthermore, we considered codebooks of the same size for all levels of the quadtree. Indeed, when codebook size increases with block size, RMS has a relative advantage; when codebook size decreases, the advantage is with entropy and variance.

Table 1 shows the speed-ups achieved using adaptive entropy against adaptive RMS. Each row shows an encoding of Lena at

about the same compression ratio (CR) and quality. Depending on the rate, the speed-up ranges from 2.28 to 1.83. Note that the speed-up increases at lower bit rates: this reflects the fact that more computations are avoided in the first levels of the quadtree.

Fig. 1 compares the rate-distortion curves of the three analysed splitting functions. For each method there are two curves: one is obtained with a fixed threshold, while the other is obtained with the best adaptive threshold relation. As expected, adaptive RMS is optimal, yielding the best rate-distortion curve. However, fixed threshold entropy outperforms standard RMS because of its intrinsic adaptivity. Adaptive variance and adaptive entropy are comparable and very close to the optimum, that is, adaptive RMS. It can be seen that adaptive thresholds provide a definite improvement independently from the chosen split decision function.

The main difference between variance and entropy is in the kind of noise added to the reconstructed image: at low bit rates, variance causes low contrast ('flat') zones to be poorly coded; in contrast, entropy causes high contrast (sharp-edged) zones to be poorly coded. The latter kind of image degradation is significantly less noticeable to the eye.

Comparing the two curves for each method in Fig. 1, a dramatic improvement is seen to be caused by adaptive thresholds. This is even more noticeable from a visual point of view. In fact, adaptive thresholds yield a far better visual quality than fixed thresholds. This improvement is basically due to the almost complete lack of blockiness.

Conclusions: This Letter has compared several decision splitting functions for quadtree based fractal image coders. In addition, the effect of using adaptive thresholds on the various levels of the quadtree has been investigated. The experiments show that adaptivity yields greatly improved rate-distortion curves and attenuates the effect of blockiness. However, it should be noted that in the RMS error based criterion the threshold for obtaining a given image quality is independent of the particular image. That is, for a given threshold value we obtain almost the same visual image quality for every image. Unfortunately, this is not the case with entropy- or variance-based criteria. This makes it difficult to obtain an *a priori* estimate of the threshold value for encoding a given image at a given quality level.

3 March 1998
Electronics Letters Online No: 19980597

R. Distasi, M. Polvere and M. Nappi (*Università di Salerno, Dipartimento di Informatica e Applicazioni 'R.M. Capocelli', 84081 Baronissi (SA), Italy*)

References

1 FISHER, Y.: 'Fractal image compression-theory and application' (Springer-Verlag, New York, 1994)
2 SAUPE, D., HAMZAOUI, R., and HARTENSTEIN, H.: 'Fractal image compression - An introductory overview' *in* SAUPE D., and HART, J. (Eds): 'Fractal models for image synthesis, compression and analysis'. SIGGRAPH 96 Course Notes, ACM, New Orleans, 1996
3 SAUPE, D., and JACOB, S.: 'Variance-based quadtrees in fractal image compression', *Electron. Lett.*, 1997, **33**, (1), pp. 46–48
4 CHANG, Y.C., SHYU, B.K., and WANG, S.J.: 'Region-based fractal image compression with quadtree segmentation'. Proc. ICASSP'97, Munich, 1997

Abuses of probabilistic encryption schemes

Yongge Wang

A method to abuse any probabilistic encryption schemes is demonstrated.

Introduction: In a key escrow cryptosystem, the private-key is broken up into pieces and distributed to different authorities. The authorities can confer and reconstruct the private-key. Key escrow guarantees that the police can eavesdrop on all conversations or personal data files even though they are encrypted. Of course, a cryptographic user gains nothing from key escrow systems at all. He must trust the escrow agents' security procedures, and the integrity of the people involved. However, we will show that even if a user surrenders his private-key for a probabilistic cryptosystem to key escrow agents, he still can abuse the cryptosystem and communicate under the government's very nose without having to worry that it would be detected.

Abuses of probabilistic cryptosystems: In the past, several probabilistic public-key cryptosystems have been introduced by several authors, for example, the Blum-Goldwasser cryptosystem (1985), the Ajtai-Dwork cryptosystem (1997), and the Goldreich-Goldwasser-Halevi cryptosystem (1997). Here is a formal mathematical definition of this concept.

Definition: A probabilistic public-key cryptosystem is defined to be a six-tuple (**P**, **C**, **K**, **E**, **D**, **R**), where **P**, **C**, **K**, **E**, **D**, **R** are the set of plaintexts, the set of ciphertexts, the keyspace, the set of encryption rules, the set of decryption rules, and the set of randomisers, respectively. The following properties should be satisfied: For each key $K \in \mathbf{K}$, there is an encryption rule $e_K : \mathbf{P} \times \mathbf{R} \rightarrow \mathbf{C}$ and a decryption rule $d_K : \mathbf{C} \rightarrow \mathbf{P}$ such that $d_K(e_K(b, r)) = b$ for every plaintext $b \in \mathbf{P}$ and every $r \in \mathbf{R}$.

We first describe the following situation: Alice and Bob work for a criminal or terrorist organisation, and they want to exchange secrets using a probabilistic cryptosystem. But according to some laws, they must surrender their private-keys to some key escrow agents. Thus, at some later time, if their keys have been revealed, all their communications will be decrypted and they will be put into prison. However, for a probabilistic cryptosystem, they do not need to worry about this. They can abuse the system and communicate secrets without being detected.

Now, we demonstrate how Alice sends a secret $s_1s_2...s_n$ ($s_i = 0, 1$) to Bob through a probabilistic cryptosystem without being detected even though their keys have been escrowed. In general, the protocol appears as follows:

(i) Alice and Bob first agree on an 'abuse-key' $k_a = k_1k_2...k_n$ ($k_i = 0, 1$).

(ii) Alice puts $c_i = s_i \oplus k_i$ for $i \leq n$ and generates an innocuous message $x_1x_2...x_n$.

(iii) For each $i \leq n$, Alice uses Bob's public-key K to check the random encryptions $e_K(x_i, r)$ of x_i until finding an encryption e_i such that $\#(e_i) = c_i \pmod 2$ where $\#(e_i)$ denotes the number of 1s in e_i. Note that the odds of an encryption of x_i having the above property is 1 in 2 according to the properties of probabilistic cryptosystems. Therefore, Alice can find e_i by trying two encryptions of x_i, on average.

(iv) After Bob obtains the ciphertext, he reconstructs the secret s simply by counting the numbers of 1s in the ciphertexts of x_is and XORing them with k_is.

It is clear that the above abuses of a probabilistic cryptosystem cannot be detected by a warden. This attack shows that a user of a probabilistic cryptosystem does not need to worry about the key escrow policy. She call still have her own privacy even though his private-key has been escrowed and her communications might be eavesdropped on by distrustful key escrow agents. However, a manufacturer (we will call him Mallory) of a probabilistic cryptosystem implementer can also use this abuse to leak his customer's private-keys.

Mallory puts his implementation of a probabilistic cryptosystem in a tamperproof VLSI chip, so that no one can examine its inner workings. He chooses an abuse-key k_a and embeds the abuse mentioned above in his implementation. Then he distributes the chips to his customers, e.g. Alice, Bob, and everyone else who wants them. Every time Alice sends a message to Bob, the chip reads Alice's private-key k_A, and encrypts the message in such a way that key k_A can be reconstructed by Mallory after he eavesdrops on communications between Alice and Bob.

However, this attack (by Mallory) can easily be overcome because Alice can save her private-key in a safe place and does not let the chip read it when she encrypts a message for others. Of course, the chip can be so designed that it will remember Alice's private-key when decrypting some encrypted messages received by Alice and will then leak Alice's private-key when next encrypting a message for Bob. However, this attack can also be overcome by using different chips for encryptions and decryptions.