Fractal image compression based on intrablock variance distribution and vector quantization

Shin-Si Chen
Chang-Biau Yang
Kuo-Si Huang
National Sun Yat-sen University
Department of Computer Science and Engineering
Kaohsiung, Taiwan 804
E-mail: cbyang@cse.nsysu.edu.tw

Abstract. In the encoding phase of fractal image compression, most of the time is taken in finding the closest match between each range block and a large pool of domain blocks. We use the intrablock variance distributions of domain blocks to reduce the search space. For finding a close match, we need search only the domain blocks whose maximal intrablock variance quadrants are at the same corner as the range block. Thus, we reduce the number of transforms applied on each domain block from eight to two. We also adopt the longest-distance-first vector quantization scheme to divide the large pool of domain blocks into clusters. Thus, the number of domain blocks to be searched is also reduced. The experimental results show that our algorithm can reduce encoding time with only slight loss of quality. [DOI: 10.1117/1.1510743]

Subject terms: image compression; fractal; intrablock variance; vector quantization; classification.

Paper 010258 received July 27, 2001; revised manuscript received Mar. 27, 2002; accepted for publication Apr. 2, 2002.

1 Introduction

Fractal image compression is a good scheme for image compression with high quality and compression ratio. It is based on the representation of an image obtained from contractive transforms of fixed points close to the original image.[1–8] However, the conventional fractal encoding takes much time for searching domain blocks; thus, much effort has been put into reducing the fractal encoding time.[2,9–14]

The concept of classification can help us to reduce the encoding time in fractal image compression.[2,9,12,13] The main idea is that we only find the best-matching block in some subsets of all blocks. Though we may not find the best-matching block among all blocks, we can usually find a near-best-matching block. If the classification method is a good enough one, we have a good chance to find the best- or near-best-matching block. We can use the classification method instead of the exhaustive searching method to reduce the encoding time.

In this paper, to reduce the searching space, we propose a new classification method that applies the longest distance first (LDF) classification[9] on the intrablock variance distribution of domain blocks. For finding a close match, we need search only the domain blocks whose maximal intrablock variance quadrants are at the same corner as the range block. We reduce the number of transform calculations applied on each domain block from eight to two, due to the intrablock variance distribution. We also adopt the LDF vector quantization (VQ) scheme to partition the large pool of domain blocks into clusters. Thereby the number of domain blocks needing to be searched is also reduced. The experimental results show that our algorithm can reduce the encoding time with only slight loss of quality.

The rest of this paper is organized as follows. In Sec. 2, we briefly introduce the fractal compression scheme and review some related algorithms. In Sec. 3, we present our algorithm for fractal compression based on the intrablock variance distribution. The experimental results and performance analysis, compared with other fractal algorithms, are given in Sec. 4. Finally, we state some conclusions in Sec. 5.

2 Previous Work

Fractal image compression is based on the representation of an image by contractive transforms of which the fixed points are close to the original image.[7] The contractive transform and the iterated function system (IFS) are two fundamental tools of fractal image compression. The contractive transform ensures that all points will be sent to fixed points if we repeat the contractive transform n times where n is large enough. The IFS is the set of contractive transforms that map R^2 into R^2, so we can represent an IFS as

$$T=\{t_i:R^2\rightarrow R^2, \quad i=1,...,m\}.$$

We can iteratively apply the corresponding contractive transform to produce a reconstructed image. If the block size is large, fractal image compression is a method with high compression ratio, because we need store only a few bits for the fractal parameters.

In the fractal encoding scheme,[7] the original image is partitioned into N_R nonoverlapping range blocks of size $n \times n$, denoted as $R=\{R_1,\ldots,R_{N_R}\}$. The original image is also divided into N_D overlapping domain blocks of size $2n\times 2n$, denoted as $D=\{D_1,\ldots,D_{N_D}\}$. Then we contract domain blocks to the size of a range block with the subsample scheme, denoted as $\hat{D}=\{\hat{D}_1,\ldots,\hat{D}_{N_D}\}$.

For a range block R_i, to seek the closest match in the pool of domain blocks, we use a sliding window of size $2n \times 2n$ on the image, which slides column by column and then row by row, and we subsample the sliding window $\hat{D}$ to find the domain block with the minimum distortion $\mathrm{Dist}_{\mathrm{fractal}}(t_r(\hat{D}_j), R_i)$.

Given a range block $R_i = (b_1, b_2, \ldots, b_N)$, where $N = n^2$, if the coordinate transform function v_r is used to estimate R_i, where $v_r(\hat{D}_j) = (a_1, a_2, \ldots, a_N)$, then the complete transform we use is as follows[7]:

$$t_r(\hat{D}_j) = s_i \cdot v_r(\hat{D}_j) + o_i . \tag{1}$$

The distortion, s_i, and o_i are defined, respectively, as follows:

$$\mathrm{Dist}_{\mathrm{fractal}}(t_r(\hat{D}_j), R_i) = \sum_{k=1}^{N} (s_i \cdot a_k + o_i - b_k)^2, \tag{2}$$

$$s_i = \frac{N\Sigma_{k=1}^{N} a_k b_k - (\Sigma_{k=1}^{N} a_k)(\Sigma_{k=1}^{N} b_k)}{N\Sigma_{k=1}^{N} a_k^2 - (\Sigma_{k=1}^{N} a_k)^2}, \tag{3}$$

$$o_i = \frac{\Sigma_{k=1}^{N} b_k - s_i \Sigma_{k=1}^{N} a_k}{N}. \tag{4}$$

If a range block is sufficiently smooth, that is, the variance of the block is smaller than a predefined threshold, then we use the mean of the block to represent all its pixels. Otherwise, we search all the contracted domain blocks $\hat{D}_j$ to get the near-closest match with the range block, that is, we apply the transform t_r (the r'th-coordinate transform and grayscale transform) to find the smallest distortion. Note that, if the contracted domain block $\hat{D}_j$ is smooth, $\hat{D}_j$ is removed from the pool of domain blocks that we are going to search.

Many algorithms have incorporated efforts to reduce the encoding time of fractal image compression.[2,9,10–13] Lee and Lee proposed a simple method[10] to reduce the searching time for finding a close match between a range block and a large pool of domain blocks. Their method is based on the fact that two blocks are not similar if the difference of their variances is large. A search window is used to reduce the size of the search pool of contracted domain blocks. A small search-window size will reduce the encoding time, but a large one will produce reconstructed images with better quality. The algorithm first sorts all contracted domain blocks according to their self-variances. Then for each range block, it finds the domain block in the search window with the closest match, that is, whose self-variance is closest to that of the range block.

Lee[11] proposed a method similar to that of Lee and Lee.[10] The main difference is that Lee[11] uses a flexible search area instead of a fixed search-window size. The search area is bounded by a distortion inequality. We usually estimate the distortion by the squared Euclidean distance (SED). Lee defined the squared variance distance (SVD) and derived an inequality between SED and SVD. We need to search for the minimum distortion only in those domain blocks that satisfy the distortion inequality. Thus, we have a flexible search area for finding the minimum distortion, and we do not have to pay attention to setting the search-window size.

In order to speed up fractal encoding, one can apply classification to reduce the number of domain blocks to be searched for each range block. Fisher[2] proposed a classification scheme with three major classes and 24 subclasses for each major class. While searching for the match of domain blocks for each range block, he searches only the domain blocks that are in the same class as the range block. Hence the encoding time required for searching can be reduced.

Pfefferman et al.[13] also proposed a classification scheme, called γ_{class}. In it, each block is divided into four quadrants, and all blocks are classified into 24 classes by their quadrant ordering relations $(4! = 24)$. When searching for the closest match for each range block in the pool of domain blocks, we can choose the one of the 24 transforms that has the same quadrant ordering relation, so that the transformed domain blocks will be in the same class as the range block. This scheme can reduce the encoding time, but the compression ratio is a little worse than with the conventional fractal encoding.

For training a local VQ codebook,[14–16] the original image is partitioned into a set of training vectors. At the end of the classification, similar vectors are put into the same cluster. The VQ clustering concept can be applied to classification of domain blocks.[12] An efficient codebook generation algorithm for VQ is also very important. The LDF algorithm is one efficient method to generate the codebook.[9]

In the clustering scheme of fractal compression with VQ,[12] the training set consists of all domain blocks, and the LDF algorithm is used to train a codebook with size N_C. In other words, the pool of domain blocks is divided into N_C clusters after LDF is performed. When we search for the closest match for each range block, we first find the closest codeword in the codebook; then we search the domain blocks associated with the codeword (cluster). The LDF fractal encoding method effectively reduces the encoding time, and the quality of the reconstructed images is comparable to that obtained by other methods.

3 The Intrablock Variance Distribution Scheme

Lee and Lee[10] pointed out that two blocks are not similar if the difference of their variances is large. We utilize this fact and consider the shape of each block. We observe that if two blocks are similar, the distributions of intrablock variances of these two blocks should also be similar. How to represent the notion of similarity between distributions of intrablock variance is, however, a problem. We propose a simple heuristic: two blocks are similar if their quadrants with the maximal intrablock variance are in the same corner. More precisely, in our heuristics, to find the best match for a range block, we search only the set of transformed domain blocks whose quadrants with the maximal intrablock variance are in the same corner as the range block.

We first define the intrablock variance. We divide a block A into four quadrants $A^{(1)}$, $A^{(2)}$, $A^{(3)}$, and $A^{(4)}$, as shown in Fig. 1(a). The intrablock variance of quadrant

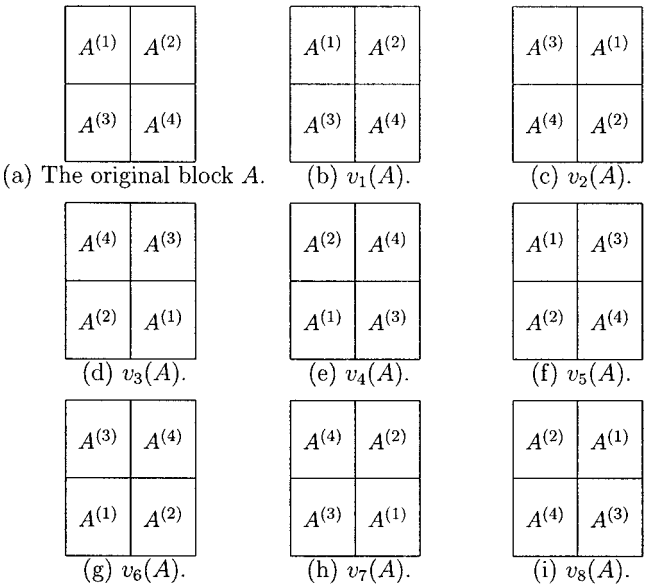


Fig. 1 The eight self-symmetrical transformations of a block A with four quadrants.

$A^{(i)}$, $1 \leqslant i \leqslant 4$, with respect to A is defined as

$$\sigma_{A^{(i)}} = \sum_{a_j \in A^{(i)}} (a_j - \overline{A})^2, \qquad 1 \leqslant i \leqslant 4,$$

where a_j represents one pixel value and $\overline{A}$ is the mean of block A. Note that the variance of block A is defined as

$$\sigma_A = \sum_{a_j \in A} (a_j - \overline{A})^2 = \sum_{i=1}^{4} \sigma_{A^{(i)}}.$$

Hence we can calculate the four intrablock variances of each block A (each range block and each domain block), and then find the quadrant position q_A with maximal intrablock variance (MIV) among $A^{(1)}$, $A^{(2)}$, $A^{(3)}$, and $A^{(4)}$. That is, $A^{(q_A)}$ has the MIV among the four quadrants.

For seeking a close match between a range block and the set of domain blocks, we need search only the set of transformed domain blocks whose MIV quadrants are in the same corner as the range block. For example, assume the MIV quadrant of the contracted domain block D_j is in the lower left corner and the MIV quadrant of the range block R_i is in the upper right corner. The eight self-symmetrical transformations on one block with four quadrants are defined clockwise as follows: identity; rotations through $+90$, $+180$, and $+270$ deg; and reflections about the -45-deg diagonal line, the horizontal midline, the 45-deg diagonal line, and the vertical midline. The eight self-symmetrical transformations on one block with four quadrants are shown in Fig. 1. Note that in the figure, $A^{(i)}$, $1 \leqslant i \leqslant 4$, is not the same as the original $A^{(i)}$ once the transformations have been applied. In this example, after transformations on domain block $\hat{D}_j$, the MIV quadrants of both $v_3(\hat{D}_j)$ and $v_5(\hat{D}_j)$ are in the upper right corner. Thus, we only try to find a close match between R_i and $v_3(\hat{D}_j)$ and between R_i and $v_5(\hat{D}_j)$, and ignore other transformations on $\hat{D}_j$. Then the number of comparisons (transformations) between one range block and one domain block is reduced from 8 to 2. For convenient comparison, in our algorithm we always transform each range block and each domain block so that the MIV quadrant is in the upper left corner. After the four intra-block variances of one domain block are calculated, we need only one transformation, instead of eight, to put the MIV quadrant in the upper left corner.

Our scheme is to reduce the transforms applied to contracted domain blocks so that we can speed up the encoding of the fractal image compression. Our algorithm is given as follows.

Algorithm: Fractal with intra-block variances

- *Input:* An original image.
- *Output:* The encoding information.
- *Step 1:* Partition the original image into N_R nonoverlapping range blocks, denoted as $\mathcal{R} = \{R_1, R_2, \ldots, R_{N_R}\}$.
- *Step 2:* Extract N_D overlapping domain blocks from the original image, denoted as $\mathcal{D} = \{D_1, D_2, \ldots, D_{N_D}\}$.
- *Step 3:* Contract each domain block to the size of a range block, denoted as $\hat{D} = \{\hat{D}_1, \hat{D}_2, \ldots, \hat{D}_{N_D}\}$.
- *Step 4:* For each $\hat{D}_j$, calculate its intrablock variances $\sigma_{\hat{D}_j^{(m)}}$, $1 \leqslant m \leqslant 4$, and variance $\sigma_{\hat{D}_j} = \Sigma_{m=1}^{4} \sigma_{\hat{D}_j^{(m)}}$. If $\sigma_{\hat{D}_j} < T_\sigma$, where T_σ is a predefined threshold, then remove $\hat{D}_j$ from $\hat{D}$.
- *Step 5:* Rotate domain block $\hat{D}_j$ (use only one of the four transforms: identity, rotation through $+90$ deg, rotation through $+180$ deg, and rotation through $+270$ deg) so that the MIV quadrant of $\hat{D}_j$ is in the upper left corner, denoted as $d(\hat{D}_j)$, for $j = 1, \ldots, N_D$.
- *Step 6:* Apply the LDF algorithm to split the rotated domain blocks $d(\hat{D}_j)$ into N_C clusters, where N_C is a predefined codebook size. The set of all clusters is denoted as $C = \{C_1, C_2, \ldots, C_{N_C}\}$, and the representative codeword of each cluster C_k is denoted as P_k, $1 \leqslant k \leqslant N_C$.
- *Step 7:* For each range block R_i, do the following.
  - *Step 7.1:* Calculate the intrablock variance of $\sigma_{R_i^{(m)}}$, $1 \leqslant m \leqslant 4$, and the variance $\sigma_{R_i} = \Sigma_{m=1}^{4} \sigma_{R_i^{(m)}}$. If $\sigma_{R_i} < T_\sigma$, where T_σ is a predefined threshold, that is, if the range block R_i is smooth enough, then output the mean of R_i to represent R_i. Otherwise apply two of the eight transforms on R_i, denoted as $r_1(R_i)$ and $r_2(R_i)$, so that the MIV quadrant of R_i is in the upper left corner.
  - *Step 7.2:* Set a search window size w. The window size is the number of clusters within which we want to search for the minimum distortion.
  - *Step 7.3:* Find w codewords (clusters) close to $r_1(R_i)$

and w codewords close to $r_2(R_i)$. The set of the $2w$ corresponding close-cluster indices is denoted as $S_l = \{s_{l,1},\ldots,s_{l,w}\}$, $1 \leqslant l \leqslant 2$, so that $\forall k \notin S_l$, $\mathrm{Dist}_{\mathrm{fractal}}(P_k, r_l(R_i)) \geqslant \mathrm{Dist}_{\mathrm{fractal}}(P_{s_{l,m}}, r_l(R_i))$ $\forall s_{l,m} \in S_l$.

•*Step 7.4:* Find a rotated domain block with the minimum distortion among the $2w$ close clusters. That is, find $d(\hat{D}_{\min})$ such that $\mathrm{Dist}_{\mathrm{fractal}}(d(\hat{D}_{\min}), r_l(R_i))$ is the minimum, $\forall d(\hat{D}_j) \in C_{s_{l,m}}$, $1 \leqslant m \leqslant w$, and $l = 1,2$.

•*Step 7.5:* We can find the relative transform v_r, since we know the rotation used on the domain block and the transform used on the range block. Then output I_i [the values of D_j, v_r, s_i, o_i in Eq. (1)].

In our algorithm, instead of performing classification on all eight transformations of domain blocks, we use only one transformation for classification. After the transformation, the MIV quadrant of each contracted domain block is in the upper left corner. Thus, we can reduce the time taken for the classification. In addition, the number of transforms applied to the comparison of one range block and one contracted domain block is reduced from 8 to 2.

If we directly apply the intrablock variance distribution scheme to the conventional fractal compression method, we may find some cases where the optimal domain block matched with the range block is excluded. The main reason is that we reduce the number of transforms to 2, and we only consider the transforms such that the quadrant with the MIV is in the upper left corner. However, some special blocks may have two or three quadrants with approximately equal MIV values. In this case, our simple intrablock variance distribution method will select one of the quadrants with the same MIV value.

In our algorithm, the intrablock variance distribution method is applied, for reducing the number of transformations, on the domain block pool when performing the LDF classification and on each range block to find the good matching domain block in some cluster. The LDF classification method will partition the pool of domain blocks into many small clusters. The probability that the optimal domain block is excluded for each range block depends on the LDF classification method and intrablock variance distribution method. For finding a better block match, we set the window size equal to the number of clusters to be searched. A small search-window size will reduce the encoding time, but a large one will have a better chance of finding the optimal domain block for each range block and get better reconstructed image quality. In the conventional fractal compression, the optimal domain block is found. In our algorithm, a near-optimal domain block is found; thus the required time is reduced, with slight loss of quality.

The threshold T_σ is used to decide if a block is smooth. With smaller T_σ, we can get the reconstructed image with better quality; with larger T_σ we can reduce the encoding time and increase the compression ratio. In this work, after experimental tests, we set $T_\sigma = 25$, which performs well in most of our experiments. Hence the smooth range blocks are represented as their means; the smooth domain blocks in the search pool are no longer needed.

4 Experimental Results and Performance Analysis

In this section, we show our experimental results and analyze the performance of our algorithms. Our algorithm is implemented by Borland C++Builder on a PC with AMD Thunder Bird™ processor (1 GHz) and 256 Mbyte of RAM. Our testing images include "Lena," "F16," "Pepper," and "Baboon." For a more reliable evaluation of our method, we used another 50 test images. All of these images are of 256 gray levels with resolution 256×256. Our initial image for reconstructing the image from the encoding information is an image whose pixel values are all 128.

To measure the quality of the reconstructed image, we use the peak signal-to-noise ratio (PSNR), which is defined as

$$\mathrm{PSNR} = 10 \log_{10}\left[\frac{255^2}{(1/L \times L)\sum_{i=1}^{L}\sum_{j=1}^{L}(x_{ij} - \hat{x}_{ij})^2}\right],$$

where $L \times L$ = size of image, x_{ij} = pixel value of the original image at coordinate (i, j), and $\hat{x}_{ij}$ = pixel value of the reconstructed image at coordinate (i, j).[9,17] The best measure for image compression is human vision. If we cannot perceive the difference between the original image and reconstructed image, we can conclude the compression method is a good one. But it is a hard task to quantitate the sensorial measure. We use the PSNR measure in this paper, since it is a widely used quantitive and mathematical measure for the reconstructed image quality.

Table 1 shows the performance of various fractal algorithms.[2,4,9,10–13] A small search-window size reduces the encoding time, but a large one yields reconstructed images with better quality. We set the window size at 3000 for the local variance fractal (LVF) method[10]; this size is a good trade-off between fractal encoding time and the quality of reconstructed images. As we can see in the average of the 50 test images, the encoding time of our scheme is between those of the Fisher classification[2] and LDF fractal image compression,[12] and the quality of the reconstructed image is between that of LDF fractal image compression and that of fractal encoding with variance-ordered partial search (VPS).[11]

The only algorithm that takes less time than ours is fractal encoding with Fisher classification, but the quality of its reconstructed images is much worse than that obtained in our algorithm. Fisher[2] uses the mean and variance order relations of quadrants to classify all domain blocks into 3×24 classes. For a given range block, he searches only the class that has the same mean and variance order relation as that block. But the best-matching block may be in some other class after a coordinate transformation is applied. Thus different sequences of classification and transformation will yield different results. Hence, Fisher's method takes less time but yields worse reconstructed image quality.

The VPS method[11] uses the distortion inequality between SED and SVD to bound the search area. The method may reduce the number of domain blocks for matching with each range block. It requires additional time to compute the SED and SVD values, and it also needs to verify the distortion inequality. If an image is complicated, like

Table 1 Performance of various fractal algorithms. The window size of the local variance fractal (LVF) is 3000. Image 256×256, threshold 25. (a) Domain block size 8×8, range block size 4×4. (b) Domain block size 16×16, range block size 8×8.

		(a)			(b)		
Image	Algorithm	PSNR	Time (s)	Bit rate (bits/pixel)	PSNR	Time (s)	Bit rate (bits/pixel)
"Lena"	Conv. fractal[4]	33.9594	3355.4650	1.2634	28.1364	2540.7400	0.3566
	γ_{class}[13]	34.0166	328.1510	1.3272	28.1341	297.9330	0.3794
	LVF[10]	32.3305	233.4290	1.2634	27.1589	223.1890	0.3566
	VPS[11]	32.0662	421.8470	1.2634	27.9494	1890.8290	0.3566
	Fisher[2]	30.6072	64.2000	1.2634	26.6649	126.4100	0.3566
	LDF fractal[12]	33.6956	190.6750	1.2634	28.1149	184.1850	0.3566
	Our scheme	33.7067	45.3350	1.2634	28.0945	73.6950	0.3566
"F16"	Conv. fractal[4]	32.1948	2491.4310	1.2054	25.7915	2121.2400	0.3201
	γ_{class}[13]	32.3126	248.1550	1.2638	25.7039	226.0320	0.3390
	LVF[10]	30.2904	178.7400	1.2054	25.1129	167.6090	0.3201
	VPS[11]	29.8026	409.6000	1.2054	25.5892	1662.6250	0.3201
	Fisher[2]	27.7408	30.4300	1.2054	24.3190	20.3800	0.3201
	LDF fractal[12]	32.0254	152.5150	1.2054	25.6301	138.9230	0.3201
	Our scheme	31.9783	46.5400	1.2054	25.6012	67.9810	0.3201
"Pepper"	Conv. fractal[4]	33.3405	3307.2260	1.3524	26.9574	2494.4200	0.3755
	γ_{class}[13]	33.3332	326.7510	1.4242	26.8140	300.9200	0.4002
	LVF[10]	31.9643	229.3550	1.3524	26.2108	225.2850	0.3755
	VPS[11]	29.9808	273.3960	1.3524	26.9016	2665.1300	0.3755
	Fisher[2]	31.1345	83.1660	1.3524	26.3814	102.6600	0.3755
	LDF fractal[12]	33.1467	227.7060	1.3524	26.8843	220.3690	0.3755
	Our scheme	33.1291	73.9350	1.3524	26.7390	83.5310	0.3755
"Baboon"	Conv. fractal[4]	27.1063	3712.4600	1.8157	23.1487	2578.4990	0.4308
	γ_{class}[13]	27.2292	495.4850	1.9296	23.2142	361.1450	0.4614
	LVF[10]	25.3877	357.6900	1.8157	22.8010	266.5100	0.4308
	VPS[11]	26.9375	5299.8650	1.8157	23.1434	3961.4350	0.4308
	Fisher[2]	25.2067	87.8950	1.8157	22.4050	65.8750	0.4308
	LDF fractal[12]	26.9515	330.9240	1.8157	23.0751	155.6600	0.4308
	Our scheme	26.7879	84.6500	1.8157	23.0024	86.9410	0.4308
Av. above 4	Conv. fractal[4]	31.6503	3216.6460	1.4092	26.0085	2433.7250	0.3708
	γ_{class}[13]	31.7217	349.6355	1.4862	25.9666	296.5075	0.3950
	LVF[10]	29.9932	249.8035	1.4092	25.3209	220.6483	0.3708
	VPS[11]	29.5168	1601.1770	1.4092	25.8959	2545.0048	0.3708
	Fisher[2]	28.6723	66.4228	1.4092	24.9426	78.8313	0.3708
	LDF fractal[12]	31.4548	225.4550	1.4092	25.9261	174.7843	0.3708
	Our scheme	31.4005	62.6150	1.4092	25.8593	78.0280	0.3708
Av. 50 test	Conv. fractal[4]	30.6148	2862.9681	1.2084	25.8409	2180.6427	0.3264
	γ_{class}[13]	30.4859	326.7684	1.2886	25.8087	273.4181	0.3501
	LVF[10]	29.1432	232.4405	1.2084	25.1633	204.4043	0.3264
	VPS[11]	29.2564	1423.9290	1.2084	25.7603	2737.5170	0.3264
	Fisher[2]	28.4969	82.8976	1.2084	24.4806	68.3166	0.3264
	LDF fractal[12]	30.2600	198.3854	1.2084	25.7310	153.5858	0.3264
	Our scheme	30.2526	86.8004	1.2084	25.6670	97.0054	0.3264

"Baboon," the VPS distortion inequality may not be satisfied in most range blocks. Almost all matching between range blocks and domain blocks has to be checked. Then the VPS method will take more time than the conventional fractal compression method.

In Table 1, it is interesting that some of these algorithms take more time when the range block size becomes smaller, from 8×8 to 4×4, but the others are not. Our scheme with range block size 4×4 takes less time than with 8×8. More precisely, the "Lena" image with our scheme shows a 40% reduction in encoding time on going from range block size 8×8 to 4×4. One might think that it would require more

Table 2 PSNR comparison for conventional fractal method, our method, and JPEG 2000. Image size 256×256, fractal threshold 25. (a) Domain block size 8×8, range block size 4×4; (b) domain block size 16×16, range block size 8×8; (c) domain block size 32×32, range block size 16×16; (d) domain block size 64×64, range block size 32×32.

		(a)		(b)		(c)		(d)	
Image	Algorithm	PSNR	Bit rate (bits/pixel)	PSNR	Bit rate (bits/pixel)	PSNR	Bit rate (bits/pixel)	PSNR	Bit rate (bits/pixel)
"Lena"	Conv. fractal[4]	33.9594	1.2634	28.1364	0.3566	23.8904	0.0898	18.9320	0.0225
	Our scheme	33.7067	1.2634	28.0945	0.3566	23.7092	0.0898	18.8821	0.0225
	JPEG 2000[18]	39.6195	1.2634	30.5544	0.3566	23.5941	0.0898	14.5580	0.0272
"F16"	Conv. fractal[4]	32.1948	1.2054	25.7915	0.3201	21.6580	0.0898	18.0104	0.0225
	Our scheme	31.9783	1.2054	25.6012	0.3201	21.3195	0.0898	18.1620	0.0225
	JPEG 2000[18]	38.1917	1.2054	28.0914	0.3201	22.0216	0.0898	11.5123	0.0272
"Pepper"	Conv. fractal[4]	33.3405	1.3524	26.9574	0.3755	23.1996	0.0898	16.7928	0.0225
	Our scheme	33.1291	1.3524	26.7390	0.3755	23.0782	0.0898	18.0671	0.0225
	JPEG 2000[18]	37.3644	1.3524	30.1077	0.3755	22.6863	0.0898	14.1919	0.0272
"Baboon"	Conv. fractal[4]	27.1063	1.8157	23.1487	0.4308	21.4388	0.0898	19.4220	0.0225
	Our scheme	26.7879	1.8157	23.0024	0.4308	21.3536	0.0898	19.5885	0.0225
	JPEG 2000[18]	32.7895	1.8157	24.5569	0.4308	21.5092	0.0898	16.4020	0.0272
Average	Conv. fractal[4]	31.6503	1.4092	26.0085	0.3708	22.5467	0.0898	18.2893	0.0225
	Our scheme	31.4005	1.4092	25.8593	0.3708	22.3651	0.0898	18.6659	0.0225
	JPEG 2000[18]	36.9913	1.4092	28.3276	0.3708	22.4528	0.0898	14.1661	0.0272

time when the range block size decreases from 8×8 to 4×4, because the number of range blocks is quadrupled. The reduced amount of computation is the major reason for the time reduction.

Indeed, in a 256×256 image, we have $(256-16+1)^2$ and $(256-8+1)^2$ domain blocks when the domain block sizes are 16×16 and 8×8, respectively. There are $(256/8)^2$ and $(256/4)^2$ range blocks when the range block sizes are 8×8 and 4×4, respectively. For calculating the MIV of a block, we need about four times as much computation for range block size 8×8 as for 4×4. Hence, we need the same time for calculating the MIV of all range blocks, but the ratio of MIV computation for domain block size 8×8 to that for 16×16 is $(256-8+1)^2/(256-16+1)^2\times(1/4)=0.26687$. For seeking the matching block, the number of range blocks examined is quadrupled. Since we use the LDF method to classify all domain blocks, we need to search only some domain blocks. We also consider only two of the eight transforms of the conventional fractal method. Hence, our scheme takes less encoding time as the range block size decreases from 8×8 to 4×4. It should be noted that our method may take less time and yield better reconstructed image quality with range block size 4×4 than with 8×8.

Table 2 shows the PSNR comparison for the conventional fractal method, our method, and JPEG 2000.[18] The JPEG 2000 standard is an excellent method with little encoding time and high reconstructed image quality at compression ratios less than 100. In the table, we find that the PSNR of JPEG 2000 is better than that of the fractal method when the range block size is 4×4 or 8×8. But the PSNR of the fractal image compression method is comparable to that of JPEG 2000 when the range block size is 16×16. When the range block size of the fractal compression method is enlarged to 32×32, the PSNR of the fractal compression method is better than that of JPEG 2000. Note that in Table 2(d), the bit rate of JPEG 2000 cannot be made the same as that of fractal compression, because the least bit rate of JPEG 2000 in images with size 256×256 and 8-bit gray levels is 0.0272 bits/pixel.

Our algorithm speeds up fractal encoding successfully with very small degradation in the quality of the reconstructed image. Figure 2 shows the "Lena" image reconstructed by our algorithm. Its image size is 256×256, range block size is 4×4, domain block size is 8×8, and PSNR is 33.7067 with 1.2634 bits per pixel. If we inspect Fig. 2 minutely, we may find some block effects. Fractal image compression is of course a block-oriented method. If the range block size is larger, the block effect will be more apparent. However, we may find the block effect is not apparent for uniform backgrounds, such as clear sky, sea, or solid regions.

5 Conclusion

In this paper, we propose a fast encoding algorithm for fractal image compression based on the intrablock variance distribution. In the conventional fractal encoding algorithm, it takes a lot of time to search for a best match from large pool of transformed domain blocks for each range block. We reduce the number of transforms applied to each domain block from 8 to 2. Thus, the time required for the comparison between one range block and one domain block is reduced. In addition, the LDF VQ scheme is used to

Fig. 2 "Lena" reconstructed by our algorithm. Range block size 4 $\times$4; domain block size 8$\times$8; PSNR 33.7067; bit rate 1.2634 bits/pixel.

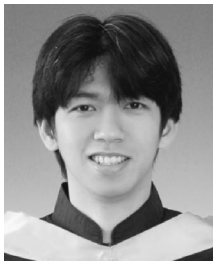
divide the large pool of domain blocks into clusters. Thereby the number of domain blocks needing to be searched is also reduced.

Experimental results also show that our method is faster than the conventional fractal encoding method and the LDF fractal compression method, with only slight loss of quality of the reconstructed images under the same compression ratio. We also explain why our algorithm takes less encoding time when the range block size becomes smaller. It is interesting that our method with range block size 4×4 may take less time and yield better reconstructed image quality than with 8×8.

Considering the trade-off between encoding time and reconstructed image quality, the performance of our scheme is better than that of other fractal compression methods. By our experimental results, JPEG 2000 is superior to fractal image compression, if the PSNR is the major consideration the compression ratio is not too high. But if we want a very high compression ratio, the reconstructed image quality of the fractal compression method is better than that of JPEG 2000.

References

1. F. Davoine, M. Antonini, J. M. Chassery, and M. Barlaud, "Fractal image compression based on Delaunay triangulation and vector quantization," *IEEE Trans. Image Process.* **5**(2), 338–346 (1996).
2. Y. Fisher, *Fractal Image Compression: Theory and Application*, Springer-Verlag, New York (1994).
3. D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE* **40**, 1098–1101 (1952).
4. A. E. Jacquin, "Image coding based on a fractal theory of iterated contractive image transformations," *IEEE Trans. Image Process.* **1**(1), 18–30 (1992).
5. A. E. Jacquin, "Fractal image coding: a review," *Proc. IEEE* **81**(10), 1451–1465 (1993).
6. S. Kumar and R. C. Jain, "Low complexity fractal-based image compression technique," *IEEE Trans. Consumer Electron.* **43**(4), 987–993 (1997).
7. S. C. Tai, *Data Compression*, 2nd ed., Unalis, Taipei, Taiwan (1998).
8. B. Wohlberg and G. de Jager, "A review of fractal image coding literature," *IEEE Trans. Image Process.* **8**(12), 1716–1729 (1999).
9. M. C. Huang and C. B. Yang, "Fast algorithm for designing better codebooks in image vector quantization," *Opt. Eng.* **36**, 3265–3271 (1997).
10. C. K. Lee and W. K. Lee, "Fast fractal image block coding based on local variances," *IEEE Trans. Image Process.* **7**(6), 888–891 (1998).
11. S. M. Lee, "A fast variance-ordered domain block search algorithm for fractal encoding," *IEEE Trans. Consumer Electron.* **45**(2), 275–277 (1999).
12. L. C. Lin, C. B. Yang, and K. T. Tseng, "Image compression based on fractal with classification by vector quantization," in *Proc. Int. Comput. Symp., Workshop on Image Processing and Pattern Recognition*, Chiayi, Taiwan, pp. 469–474 (2000).
13. J. D. Pfefferman, P. E. Cingolani, and B. Cernuschi-Frias, "An improved search algorithm for fractal image compression," in *Proc. 6th IEEE Int. Conf. on Electronics, Circuits and Systems 1999*, Vol. 2, pp. 693–696 (1999).
14. J. M. Chen, C. B. Yang, and K. S. Huang, "Image vector quantization with hierarchical tree structure," in *Proc. 14th IPPR Conf. on Computer Vision Graphics and Image Processing, Pingtong, Taiwan* (2001).
15. C. H. Chiou and C. B. Yang, "Compression on the block indexes in image vector quantization," in *Proc. 14th IPPR Conf. on Computer Vision Graphics and Image Processing, Pingtong, Taiwan* (2001).
16. C. H. Yang, C. B. Yang, and S. H. Shiau, "Vector quantization based on block orientation," in *Proc. Workshop on the 21st Century Digital Life and Internet Technologies*, Tainan, Taiwan (2001).
17. C. K. Chan and C. K. Ma, "A fast method of design better codebooks for image vector quantization," *IEEE Trans. Commun.* **42**(2/3/4), 237–243 (1994).
18. M. Rabbani and R. Joshi, "An overview of the JPEG 2000 still image compression standard," *Signal Process. Image Commun.* **17**(1), 3–48 (2002).



Shin-Si Chen received the BS degree in applied mathematics and MS degree in computer science and engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, in 1998 and 2000, respectively. His research interests include image processing, data compression, and parallel processing.

Chang-Biau Yang received the BS degree in electronic engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1982 and the MS degree in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 1984. Then, he obtained his PhD degree in computer science from National Tsing Hua University in 1988. He is currently a professor of the Department of Computer Science and Engineering, National Sun Yat-sen University. His research interests include computer algorithms, interconnection networks, data compression, and bioinformatics.

Kuo-Si Huang received his BS degree in applied mathematics from National Sun Yat-sen University, Kaohsiung, Taiwan, in 1997. He is now a PhD candidate in computer science and engineering at National Sun Yat-sen University, Kaohsiung, Taiwan. His current research interests are computer algorithms and bioinformatics.