Available online at www.sciencedirect.com
SCIENCE DIRECT®
Signal Processing: *Image Communication* 19 (2004) 393–404
SIGNAL PROCESSING: IMAGE COMMUNICATION
www.elsevier.com/locate/image

A fast no search fractal image coding method

Shen Furao[a,*], Osamu Hasegawa[b,c]

[a] *Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology (TIT), Tokyo, Japan*
[b] *Imaging Science and Engineering Laboratory, Tokyo Institute of Technology (TIT), 4259 Nagatsuda, Midori-ku, Yokohama, 226-8503 Japan*
[c] *PRESTO, Japan Science and Technology Agency (JST), Japan*

Received 26 June 2003; accepted 6 February 2004

Abstract

A new no search fractal image coding scheme is introduced which is able to improve the speed of fractal image compression greatly. Every time-consuming part of fractal coding is redesigned and accelerated with new techniques. Compared with the most recent scheme of Tong and Wong, this method speeds up the encoding process by 22 times and maintain the compression quality. Experiments on standard images show that the proposed scheme gets the fastest speed of fractal image coding up to the present and holds high reconstruction fidelity. For example, using PII 450 MHz PC, the proposed scheme spends 0.515 s to compress the Lena ($512 \times 512 \times 8$) with 36.04 dB PSNR decoding quality. Using Dell PIV 2.8 GHz PC, it spends only 0.078 s to finish the encoding process and gets 36.04 dB PSNR.

Keywords: Fractal image coding; No search; Similar degree; Quadtree

1. Introduction

Fractal image coding has been used in many image processing applications such as feature extractions, image watermarking, image signatures, image retrievals and texture segmentation [19]. It has the advantage of very fast decompression as well as the promise of potentially very high compression ratios. Another advantage of fractal image compression is its multi-resolution property, an image can be decoded at higher or lower resolutions than the original, and it is possible to "zoom-in" on sections of the image [13]. These advantages make fractal-image coding a very attractive method for applications in multimedia: for example, Microsoft adopted it for compressing thousands of images in its *Encarta* multimedia encyclopedia [2].

Several natural image features such as straight edges and constant regions are unchanged by rescaling and this type of redundancy is not exploited by transform coders. This local scale invariance is one of the main properties of fractal-image coding. Although pure fractal coders cannot get as good performance as wavelet transform, the hybrid fractal coders based on transform coding and fractal image compression have been shown to give appreciable improvement in image quality, and better compression than transform coders [21]. One such coder proposed by Zhao and Yuan

*Corresponding author.
E-mail addresses: furaoshen@isl.titech.ac.jp (S. Furao), hasegawa@isl.titech.ac.jp (O. Hasegawa).

0923-5965/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.image.2004.02.002

[22] combined the discrete cosine transform (DCT) and block-based fractal coding; and Li and Kuo's fractal-wavelet coder [11] worked better than wavelet coders such as the embedded zerotree wavelet coder (EZW) proposed by Shapiro [17]. However, the pure fractal coders or the fractal part in the hybrid fractal coders suffers from long encoding time, which is its major deficiency.

According to the idea that there is affine redundancy in the real-world images, Barnsley and Sloan [3] brought forward the original idea of fractal image coding in 1988. They tried to set up the iterated function system (IFS) of one image to compress that image. In 1989, Jacquin [10] gave a fractal compression scheme based on block searching. This scheme was fit for computer processing and established the foundation of fractal image compression. He separated one image to non-overlapping range blocks, and then searched the domain pool globally or partially to get the optimal affine transformation.

The drawback of Jacquin scheme is that the block matching process is very time consuming. Plenty of research focused on how to improve the speed of fractal-image coding and almost all of them explored how to reduce the amount of domain blocks in domain pool. One kind of methods classified image blocks in some way. A range block was compared with the domain blocks in the same class. For example, Jacquin [9] classified blocks on their edge content and Jacobs et al. [8] used block brightness orientation. Monro and Dudbridge [12] localized the domain pool with domain blocks closing to a given range block. Saupe [16] excluded the domain blocks with the smallest variance. Tong and Pi [18] only searched the domain blocks that satisfied some adaptive searching condition. Shen and Hasegawa [6] searched the domain pool with the help of simulated annealing. Let N_D be the size of the domain pool, these time complexity reduction methods can only reduce the factor of proportionality in the $O(N_D)$ complexity.

Different from reducing the domain blocks in domain pool, some other researchers focused on making the search faster. Hurtgen and Stiller [7] proposed a hierarchical encoding scheme that is based upon a two level codebook search and a structural classification of its entries. Bani-Eqbal [1] arranged the domain block in a tree structure to direct the search and speed up the searching without noticeable loss of image quality. The tree search approaches are able to fundamentally reduce the order of the searching time from $O(N_D)$ to $O(\log N_D)$.

In this paper, we shall discuss some of the drawbacks of the basic fractal compression scheme and propose a new scheme without searching that can get the fastest speed of fractal encoding up to the present and hold the coding fidelity in the meanwhile. The proposed method can be extended to speed up the hybrid fractal coders and improve the performance of hybrid fractal coders.

The organization of this paper is as follows. Section 2 analyzes the basic fractal compression method and discusses some techniques to improve every part of the fractal image encoding. Section 3 adjusts the Quadtree scheme, defines a method to measure the similarity between two blocks, and introduces the no search scheme. In Section 4, experiments for some famous images are processed by the proposed method. The results show that this method can get the fastest speed of fractal image coding up to the present and maintain high reconstruction fidelity.

2. Fractal image coding

2.1. Review of basic scheme

We suppose that the size of original image f is $M \times N$. During the process of fractal image coding, image f will be segmented into non-overlapping range blocks. The size of every range block is $B \times B$. For each range block $f|R_k$, we search the domain pool Ω in order to get one domain block $f|D_k$, and then apply contractive affine transformation W_k to $f|D_k$. The optimal selection of $f|D_k$ and W_k must assure that the image $g = \{W_k(f|D_k), k = 1, 2, \ldots, N\}$, which is obtained by transforming $f|D_k$ with W_k ($k = 1, 2, \ldots, N$), has the minimum difference from $f = \{f|R_k, k = 1, 2, \ldots, N\}$. The size of $f|D_k$ must be greater than the size of $f|R_k$ to assure W_k contractive. Usually we set the size of domain

block $2B \times 2B$. The domain pool is extracted by sliding a window of size $2B \times 2B$ from the top left corner of original image f with integer step I_{step} in horizontal or vertical directions. When I_{step} equals 1, the searching is full searching and there will be $(M - 2B + 1) \times (N - 2B + 1)$ domain blocks in domain pool.

If $W_k,\ k = 1, 2, \ldots, N$, is contractive, the set of all transformation $W = \{W_k, k = 1, 2, \ldots, N\}$ is also contractive. According to the contractive mapping fixed-point theorem [5], there would be one unique image f^* and $W(f^*) = f^*$. The collage theorem guarantees that f^* will approximate to original image f. Transformation W_k and position of domain block $f|D_k$ constitute the fractal code of range block $f|R_k$. Fractal codes of all range blocks form the iterated function system (IFS) of image f.

Contractive affine transformation W_k is composed of spatial contractive transform S and gray-level transform G.

Spatial contractive transform S maps $f|D_k$ (with size $2B \times 2B$) to D_k (with size $B \times B$). We define gray-level $g(i,j),\ (i = 1, 2, \ldots, B; j = 1, 2, \ldots, B)$ of D_k as the mean of corresponding neighboring four pixels of $f|D_k$, it is

$$g(i,j) = \frac{1}{4} \sum_{(p,q) \in u_k^{-1}(i,j)} f(p,q), \tag{1}$$

where u_k is the mapping from $f|D_k$ to D_k.

Gray level transform G can be defined by the linear function

$$G(D) = aD + bI, \tag{2}$$

where the affine parameter a is the scaling coefficient, b is the luminance offset. D is the spatially contracted domain block, and I is a block of the same size of D, but with all elements equal to 1. With this definition of gray level transformation (2), the fractal encoding problem will become searching the domain pool Ω to find $\tilde{D}$, a and b to make

$$\begin{aligned} E(R, \tilde{D}) &= \min_{D \in \Omega} E(R, D) \\ &= \min_{D \in \Omega} \min_{a,b} \|aD + bI - R\|^2. \end{aligned} \tag{3}$$

Here, R is range block. The least-squares results of optimal affine parameters are

$$a = \frac{\langle R - \bar{r}I, D - \bar{d}I \rangle}{\|D - \bar{d}I\|^2}, \quad b = \bar{r} - a\bar{d}, \tag{4}$$

where $\bar{r}$ is the mean of range block, $\bar{d}$ is the mean of domain block.

However, such a least-square solution is unsatisfactory. One reason is that the solution takes no account of the constraint on scaling as required by the condition of contractivity. And in IFS, the scaling and offset coefficients must be quantized for digital storage and transmission. It is well studied that the post-quantization often leads to poorer reconstructing results and pre-quantization can get much better decoding quality. Thus, the scaling coefficient must be constrained in $[-1, 1]$ and both transformation parameters must be quantized to $a_i,\ i = 1, 2, \ldots, m$ and $b_j,\ j = 1, 2, \ldots, n$, here, m and n are determined by the bits allocated to the scaling and offset parameters. And the fractal encoding problem becomes

$$\begin{aligned} E(R, \tilde{D}) &= \min_{D \in \Omega} E(R, D) \\ &= \min_{D \in \Omega} \min_{i,j} \|a_i D + b_j I - R\|^2. \end{aligned} \tag{5}$$

The basic fractal image coding method is outlined in the pseudo-code in Algorithm 2.1.

Algorithm 2.1. Basic fractal image coding algorithm

- LOOP Range Block: For every range block $R_i,\ i = 1, 2, \ldots, N_R$, do
  {
  - LOOP Domain Search: For every domain block $D_j,\ j = 1, 2, \ldots, N_D$, do
    {
      LOOP a: For every $a_k,\ k = 1, 2, \ldots, m$, do
      {
        LOOP b: For every $b_l,\ l = 1, 2, \ldots, n$, do
        {
          Error Calculation
          $error = \|a_k D + b_l I - R\|^2$ (6)
        }
      }
    }

- Record the a, b and position of D that make the *error* minimum.

}

Here, N_R is the number of range blocks, N_D is the number of domain blocks in domain pool Ω.

2.2. Computational problems and the solution

In Algorithm 2.1, there are quadruple LOOP in the process and for every step we need to calculate the *error* defined by (6). Thus, the minimization problem defined by (5) needs quite a lot of time to find the best matching domain block and affine transform. The computational time is composed of four parts:

(1) Parameter search: the search of a_i, $i = 1, 2, \ldots, m$ and b_j, $j = 1, 2, \ldots, n$. It depends on how many bits allocated to a and b.
(2) Error calculation: the calculation of $\|a_i D + b_j I - R\|^2$. It depends on the size of range block.
(3) Number of range blocks N_R: if the number of range blocks is larger, the encoding speed will be slower.
(4) Domain search: the search of domain pool. It depends on the number of domain blocks N_D in the domain pool.

In order to improve the encoding speed, we need speed up the error calculation and cut down the parameter search times, domain search times and number of range blocks.

- Parameter search problem

  Different from (2), Øien and Lepsøy [14] subtracted the DC component of all the image blocks (both range blocks and domain blocks) and defined the gray-level transform G as

$$G(D) = a_i(D - \bar{d}I) + \bar{r}I. \tag{7}$$

  They used (7) to analyze some properties of fractal image coding and proved (7) can work well. During our research, we find that (7) is suitable for our "speed up" target and it can solve the parameter search problem partially. With (7), the fractal encoding problem (5) will be

$$E(R, \tilde{D}) = \min_{D \in \Omega} E(R, D) = \min_{D \in \Omega} \min_i \|a_i(D - \bar{d}I) - (R - \bar{r}I)\|^2. \tag{8}$$

  In (8), the IFS parameters become a_i and $\bar{r}$. Instead of searching the a_i, $i = 1, 2, \ldots, m$ and b_j, $j = 1, 2, \ldots, n$ in (5), now we only need search the a_i, $i = 1, 2, \ldots, m$. To further limit the searching times, we can restrict the bits allocated to scaling parameter a.

- Error calculation

  With the gray-level transform (7), the error calculation part becomes

$$\text{error} = \|a_i(D - \bar{d}I) - (R - \bar{r}I)\|^2. \tag{9}$$

  The calculation complexity is determined by the size of range block. To speed up the error calculation part, we partition D into two blocks $D_{\frac{1}{2},1}$, $D_{\frac{1}{2},2}$ with size $B \times B/2$ and partition R into $R_{\frac{1}{2},1}$, $R_{\frac{1}{2},2}$ with size $B \times B/2$, then calculate the

$$\text{error}_c = \|a_i(D_{\frac{1}{2},1} - \bar{d}I) - (R_{\frac{1}{2},1} - \bar{r}I)\|^2. \tag{10}$$

  If error_c is larger than one predefined threshold, the a_i will not be chosen and go on the search. This scheme can save nearly half of the calculation and will not influence the compression results.

- Number of range block

  We can increase the size of range block to reduce the number of range blocks, but it will lead to poor decoding quality because there are regions that are difficult to cover well with larger range blocks. To decrease the number of range blocks without loss of reconstruction fidelity, we use larger range blocks for constant regions and smaller range blocks for other regions. Quadtree scheme discussed in Section 3.1 is one of the techniques that can be used to reduce the range blocks and improve the encoding speed and compression ratio.

- Domain search

  For domain search, unlike the search methods with $O(N_D)$ or $O(\log N_D)$ complexity, in this paper, we will give a scheme without search to

speed up the encoding process greatly. For every range block, we fix the position of domain block corresponding to the position of range block and only search for the best $a \in \{a_i, i = 1, 2, \ldots, m\}$. How to fix the position of domain block for every range block is a very important part of this scheme, we will give the detail in Section 3.3.

3. No search fractal image coding

With the discussion in Section 2.2, in Algorithm 3.1 we outline the pseudo-code of all solution for every time-consuming part of Algorithm 2.1.

Algorithm 3.1. Improved fractal image coding of Algorithm 2.1

- Choose a tolerance level T.
- Set $R_1 = f$ and mark it uncovered.
- While there are uncovered range blocks R_i do
  {
  - According to the position of R_i, take out the corresponding domain block from the original image f.
  - LOOP a: For every a_k, $k = 1, 2, \ldots, m$, find the best a_i which minimizes expression (9).
  - If error $< T$ or $sizeof(R_i) = B_{\min}$ then Mark R_i as covered, and write out the transformation parameter a_i and $\bar{r}$.
  - else Partition R_i into smaller ranges which are marked as uncovered, and remove R_i from the list of uncovered ranges.

  }

The $B_{\min}$ in Algorithm 3.1 means the allowance minimum size of range block. Comparing with Algorithm 2.1, Algorithm 3.1 only need duplicate LOOP: range block loop and parameter search of a. We can set m, the number of a_i, very small to further improve the speed.

In Algorithm 3.1, there are two problems must be solved:

(1) How to partition the range block R_i into smaller ranges. We adopt but adjust the traditional Quadtree structure to solve this problem (in Section 3.1).

(2) How to fix the position of domain block according to the position of range block. We define one term "similar degree" in Section 3.2 to measure the similarity between two blocks, then use the similar degree to induce the position relationship of range block and corresponding domain block in Section 3.3.

From now on, we will do some experiments to compare our proposed method with some other methods. We choose *bpp* (bit per pixel) and PSNR (peak-to-peak signal-to-noise ratio) as criterion of comparison. The definition of PSNR is

$$\mathrm{PSNR} = 10 \log_{10}\left(\frac{255^2}{\frac{1}{N}\sum_{i=1}^{N}(f_i - f_i^*)^2}\right), \tag{11}$$

where f_i is the pixel of the original image and f_i^* is the pixel of the decompressed image. N denotes the total number of pixels.

3.1. Quadtree scheme

There are regions that could be covered well with larger range blocks as well as regions that are difficult to cover well this way. In a quadtree partition, a square in the image is broken up into 4 equally sized sub-squares, when it is not covered well enough by a domain. This process repeats recursively starting from the whole image and continuing until the squares are small enough to be covered within some specified *rms* tolerance. Small squares can be covered better than large ones because contiguous pixels in an image tend to be highly correlated.

The traditional quadtree scheme [5] used constant rms tolerance but it could not get the most satisfied results. In this paper, we propose the adaptive rms tolerance to improve the performance of the quadtree scheme. The reason is, for large range blocks, we hope the reconstruction fidelity be higher in order to assure the decoding quality of the whole image; and for small range blocks, we have to loose the threshold criterion to get satisfied compression ratio. It means that, for large range blocks, the tolerance error must be stricter than small range blocks. We can increase the tolerance T in Algorithm 3.1 linearly or nonlinearly to improve the compression quality. For

constant tolerance and adaptive tolerance, we tested Lena ($512 \times 512 \times 8$) with Algorithm 3.1. Fig. 1 is the experiment results. From Fig. 1 we know that this proposed adaptive tolerance technique could get better compression results than traditional constant tolerance. In this experiment, linear relation

$$T_{n+1} = 2T_n + 1 \tag{12}$$

is used to change the tolerance. Here n means the Quadtree level, T_n means the error threshold of the n level.

3.2. Similar degree

Self-similarity is one important property of fractal. For real world images have a local self-similarity, fractal image coding is to find the most similar domain block of range block. It means that the matching of two blocks depends on some measure of their similarity. From the fractal coding problem (8) we find that, to make (9) minimum, the block $D_{\text{error}} = D - \bar{d}I$ and block $R_{\text{error}} = R - \bar{r}I$ must be similar, and a_i is the factor of proportionality between two blocks.

To measure the similarity, at first, we normalize the D_{error} and R_{error} to range $[-1, 1]$ to eliminate the influence of factor of proportionality between two blocks; then add 1 to every element of the blocks to make all value of the blocks greater than 0, and we get $D_{\text{error}}^{\text{norm}}$ and $R_{\text{error}}^{\text{norm}}$. There are $n = B \times B$ pixel intensities, $d_1, \ldots, d_n$ (from $D_{\text{error}}^{\text{norm}}$) and $r_1, \ldots, r_n$ (from $R_{\text{error}}^{\text{norm}}$). We define the similar degree as

$$S_{RD} = \sum_{i=1}^{n} \min(r_i, d_i). \tag{13}$$

This similar degree gives a measure of the similarity of two blocks. If the similar degree is larger, the domain block will match the range block better.

Fig. 2 gives the geometric explain of similar degree. The shadow area of Fig. 2 is the similar degree S_{RD} between range block R and domain block D.

Similar degree can be used to accelerate the searching speed. Giving a threshold T, we reject all domain blocks for which $S_{RD} < T$, hence the computational load will be substantially reduced. We must emphasize that this condition only depends on the similar degree of two blocks and most of the calculations can be prepared before the searching process, i.e., in the pre-process part of fractal image coding, the normalization of every domain block needs to be computed only once, and then the results of normalization will be stored for future use. And the similar degree condition is independent of scaling factor a_i.

A constant tolerance will not get very satisfied results. If it is set too low, it cannot lead to considerable speed-up in the matching procedure. If it is set too high, the quality of the matching will suffer poor reconstruction fidelity. From (13), the

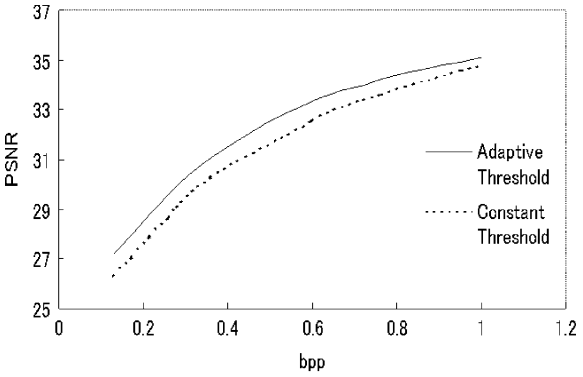

Fig. 1. Comparison of proposed adaptive tolerance with traditional constant tolerance.

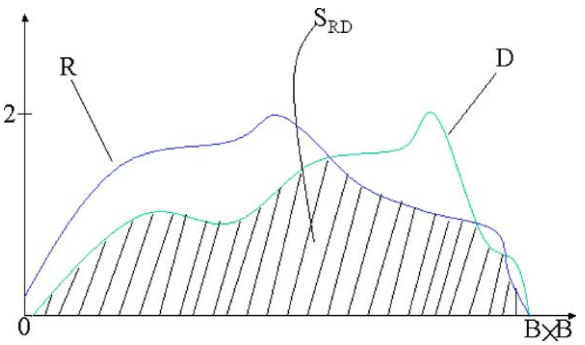

Fig. 2. Similar degree S_{RD} between range block R and domain block D.

maximum similar degree of range block R is

$$S_{\max} = \sum_{i=1}^{n} r_i, \tag{14}$$

we define the threshold value as $T = tS_{\max}$, here t is a factor and $t \in [0, 1]$. And

$$S_{RD} \geqslant T = tS_{\max} \tag{15}$$

gives the search condition. When $t = 0$, the search is full search. To check whether the similar degree condition works well or not, we incorporated condition (15) in Algorithm 2.1 and tested Algorithm 2.1 with Lena ($256 \times 256 \times 8$). Table 1 is the result.

The results show that, the PSNR of $t = 0.5$ is nearly the same as PSNR of $t = 0$. It means that the value of factor $t = 0.5$ would ensure that all range blocks and their best matching domain blocks satisfy this similar degree condition. In other words, we will achieve speed-up without suffering any loss of error. Increasing the tolerance t will further speed up the encoding, but at the expense of poorer matching since some of the best matching range block-domain block pairs will no longer satisfy the search condition.

3.3. No search scheme

According to Table 1, even similar degree condition is adopted, the encoding speed of fractal image compression cannot satisfy the real time processing requirement. Like some other fast methods, the similar degree method only reduces the factor of proportionality in the $O(N_D)$ complexity. We need a way to eliminate the search of domain pool to make the encoding speed of fractal image compression fast enough for real time task.

Table 1
Encoding results with different t

t	Encode time (s)	Speed up ratio (times)	PSNR
0	576		32.222
0.5	474	1.22	32.220
0.6	232	2.48	32.083
0.7	108	5.32	31.908
0.8	41	14.05	31.079

A scheme requiring no search has been described by Dudbridge [4], in which groups of four range blocks share their union as a common domain block. In that case encoding and decoding both have linear cost with image size, but the approximation quality is poor.

Different from the union of four domain blocks, we use the similar degree to induce the position of domain block corresponding to position of range block. Algorithm 3.2 gives the detail.

Algorithm 3.2. Fix the position of domain block by similar degree

(1) Separate the original image f to range blocks with size $B \times B$.
(2) For every range block, search the domain pool to find the domain block with similar degree greater than $tS_{\max}$. Record the position difference between best matching domain block and range block.
(3) If $B = B_{\min}$, count the position difference recorded for every range block, report the position difference with highest repeat times, stop.
Else, adjust B to $B/2$, go to step (1).

Suppose the position of range block is $(\mathrm{row}_R, \mathrm{col}_R)$, we check Lena ($512 \times 512 \times 8$), Girl ($512 \times 512 \times 8$), and Baboon ($512 \times 512 \times 8$) with Algorithm 3.2. The initial B is set as 16 and $B_{\min}$ is set as 2. For $t = 0.5, 0.6, 0.7, 0.8$, we count the position difference with highest repeat times, respectively. The statistic results show that, for most range blocks, the probability of high similar degree happening in the position of $(\mathrm{row}_R - B/2, \mathrm{col}_R - B/2)$ is larger than other positions. It means that if we fix the domain block position in $(\mathrm{row}_R - B/2, \mathrm{col}_R - B/2)$, we can get better reconstruction fidelity than other positions. Fig. 3 is the geometric description of the relation between range block R and domain block D. We compare this new position scheme with Dudbridge's scheme using Algorithm 3.1. Fig. 4 gives the comparison results and shows the advantage of new scheme. With the same *bpp*, proposed position

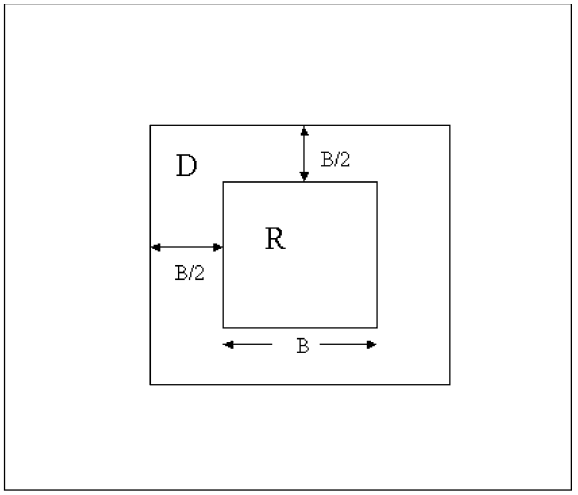

Fig. 3. Position relationship between range block R and domain block D.

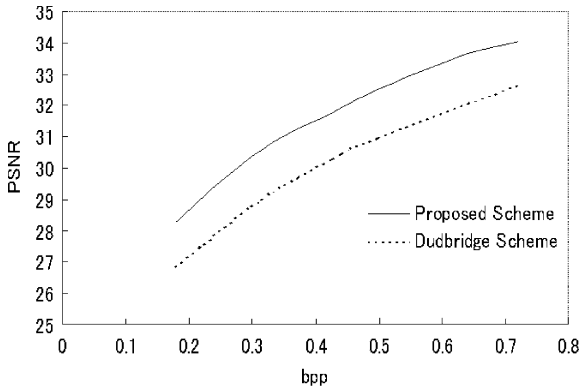

Fig. 4. Comparison of DudBridge position scheme with proposed position scheme.

scheme can get much higher PSNR than Dudbridge scheme.

In summary, our proposed no search scheme is different from Dudbridge's scheme in such aspects:

(1) Similar degree is used to induce the position of domain block and get better results (Fig. 4).
(2) Incorporating the Quadtree structure of range blocks to the no search scheme can improve the coding fidelity significantly. The adoption of adaptive tolerance can further improve the coding fidelity (Fig. 1).
(3) Instead of (5), (8) is used as the fractal encoding problem. It can make the encoding process much faster. The adoption of technique defined in (10) can speed up the error calculation thus further speed up the encoding process.

4. Experiment

With all the above discussion, we suggest the following no search algorithm to be used for fractal image coding.

Algorithm 4.1. No search fractal image coding

- Give one threshold error T, separate the original image to range blocks with size B_{ini} and mark all range blocks uncovered.
- While there are uncovered range blocks R_i, do
  {
  - Suppose the position of R_i is $(\text{row}_R, \text{col}_R)$, size of R_i is $B \times B$, take the domain block in position $(\text{row}_R - B/2, \text{col}_R - B/2)$ and calculate the error_c defined in (10).
  - If $\text{error}_c < T$ or $B = B_{\min}$, use (9) to search for the best scaling parameter a_i and store the a_i and $\bar{r}$, mark R_i as covered.
  - If $\text{error}_c \geqslant T$, separate the range block into four equally sized sub-blocks, increase the threshold value T corresponding to such blocks, mark these sub-blocks as uncovered range blocks and remove R_i from the list of uncovered range blocks.

  }

In Algorithm 4.1, B_{ini} means the initial block size of range block. We test the no search scheme with standard Lena image ($512 \times 512 \times 8$). For every range block, we use 3 bits to store the scaling parameter a_i and 1 byte to store the mean of range block $\bar{r}$. In the Quadtree structure, we considered four levels, starting at 16×16 blocks and finishing in 2×2 blocks, i.e. $B_{\text{ini}} = 16$ and $B_{\min} = 2$. The *rms* error threshold T will be changed by $T_{n+1} = 2T_n + 1$, $n = 1, 2, 3$, here T_n means the error threshold of the n level. In the IFS code, we need

to spend other 2 bits to save the quadtree level information. For those range blocks with position $\mathrm{row}_R < B/2$ and $\mathrm{col}_R < B/2$, we choose the domain block in $(\mathrm{row}_R, \mathrm{col}_R)$. For those range blocks with position $\mathrm{row}_R < B/2$ or $\mathrm{col}_R < B/2$, we choose the domain block in $(\mathrm{row}_R, \mathrm{col}_R - B/2)$ or $(\mathrm{row}_R - B/2, \mathrm{col}_R)$.

During the calculation of spatial contractive transformation of domain block, there is some repeat calculation. To eliminate the repeat calculation, before encoding of range blocks, we preprocess the original image f (with size $M \times N$) to image f_p with the same size of f. The method is to express every pixel $f_\mathrm{p}(i,j)$ by the mean of neighboring four pixels in the corresponding position of f. It is that, for $i = 1, 2, \ldots, M;\ j = 1, 2, \ldots, N$

$$f_\mathrm{p}(i,j) = \frac{1}{4}\sum_{m=0}^{1}\sum_{n=0}^{1} f(i+m, j+n). \tag{16}$$

During the encoding of range block, supposing the position of domain block is $(\mathrm{row}_D, \mathrm{col}_D)$, we only need to take out the contractive domain block from f_p with the following method:

$$D(i,j) = f(\mathrm{row}_D + 2i, \mathrm{col}_D + 2j),\quad i = 1, 2, \ldots, B;\ j = 1, 2, \ldots, B. \tag{17}$$

This technique eliminates the repeat calculation successfully.

In experiment, we use the Dell Pentium IV 2.8 GHz PC without special image processing hardware to test our method. The operating system is Windows 2000 and the programming language is Visual C++ 6.0. Table 2 (column 5 is the encoding time) is the results of Lena ($512 \times 512 \times 8$). We see that, the no search fractal coding is very fast, when PSNR > 36, it only needs 0.078 s, if PSNR = 34, it only spends 0.062 s to finish the encoding. It also shows that, with the present standard personal computer, the speed of the proposed method is enough for some real time applications of fractal image compression. Fig. 5 is the original image of Lena, Fig. 6 is the

Table 2
Encoding results of Lena ($512 \times 512 \times 8$)

T	PSNR	bpp	Time (s) Pentium II 450 M	Time (s) Pentium IV 2.8 G
3	36.04	1.38	0.515	0.078
7	35.30	0.97	0.438	0.062
16	34.02	0.67	0.359	0.062
26	33.07	0.54	0.328	0.046
39	32.03	0.43	0.313	0.046

Fig. 5. Original image of Lena ($512 \times 512 \times 8$).

Fig. 6. Decoding result of Fig. 5, PSNR = 34.02 dB, bpp = 0.67 b/p, encoding time = 0.062 s.

reconstruction result of proposed method with 0.062 s encoding time. Comparing Fig. 6 with Fig. 5, we know that the proposed method works well.

To compare our method with the updated results of fast method reported by Tong and Wong [20], we use the same test environment of Tong and Wong: a PC with an Intel Pentium II 450 MHz CPU and 128 MB memory. Column 4 of Table 2 lists the encoding time with this Pentium II 450 MHz PC.

Tong and Wong improved the algorithm designed by Saupe [15] and gave the results for Lena ($512 \times 512 \times 8$) and Baboon ($512 \times 512 \times 8$). In their results, compression ratio was reported and we change the compression ratio to *bpp* to make the comparison convenient. The results for Lena are listed in Fig. 7. From Fig. 7 we find that, with the same *bpp*, our no search scheme can improve the speed of Tong and Wong's scheme by nearly 22 times with a little loss of PSNR (less than 0.55). Compared with Saupe's algorithm, our scheme improves the *bpp* from 0.947 to 0.676 and speeds up the encoding by nearly 108 times with 0.55 dB loss of PSNR.

Table 3 is the comparison of Tong and Wong, Saupe and our method for Baboon ($512 \times 512 \times 8$). The experiment environment is PC Pentium II 450 MHz. Table 3 shows that, with 1.6 dB loss of PSNR and 0.37b/p loss of *bpp*, we speed up the Tong and Wong's scheme from 8 to 0.658 s. And for Saupe's scheme, we speed up the encoding by 91 times with 1 dB loss of PSNR.

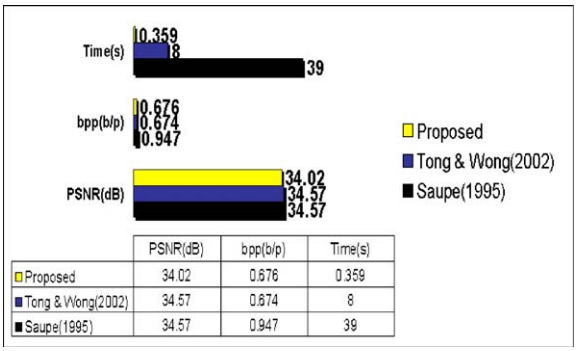

Fig. 7. Comparison of Tong and Wong scheme, Saupe scheme with proposed scheme.

Table 3
Comparison results of Baboon ($512 \times 512 \times 8$)

	PSNR	bpp	Time (s)
Proposed	24.2	1.7	0.658
Tong and Wong	25.82	1.33	8
Saupe	25.19	1.69	60

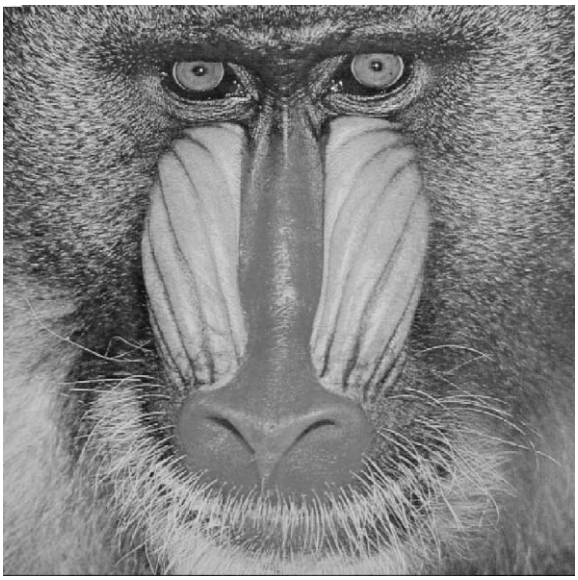

Fig. 8. Original image of Baboon ($512 \times 512 \times 8$).

When we do the test on Dell Pentium IV 2.8 GHz PC, the encoding of Baboon only spends 0.1 s (PSNR = 24.2 dB, *bpp* = 1.7 b/p). Fig. 8 is the original image of Baboon, Fig. 9 is the decoding results. From Fig. 9 we find that, for Baboon, even the PSNR is not so high, it is difficult for the human eye to detect the difference between original image and decoding image.

We also test our algorithm on some other well-known gray level images such as Girl and Gold-Hill. For the images with constant regions, our scheme works well (such as Lena); but for images with too much details (such as Baboon), the proposed scheme cannot get good compression quality (in the mean of PSNR), and nearly all fractal coders cannot work well for such images. For both kinds of images, the speed of proposed method is very fast.

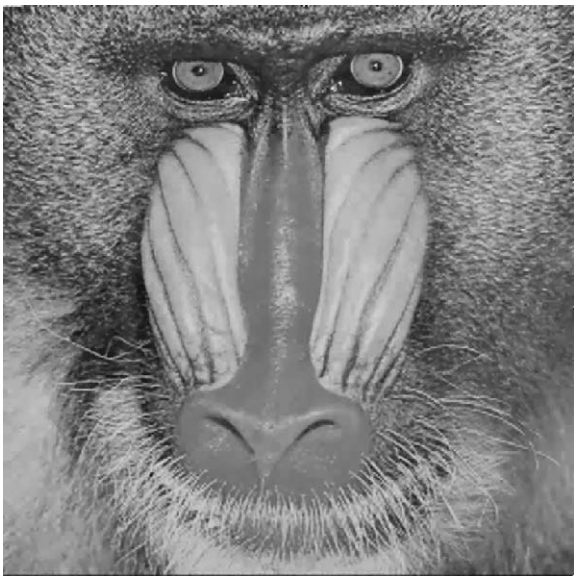

Fig. 9. Decoding of Fig. 8, PSNR = 24.2 dB, bpp = 1.7 b/p, encoding time = 0.1 s.

5. Conclusion

In this paper, we have examined the basic fractal image compression method and discussed some methods to improve every part of the fractal image encoding. We designed many new techniques to improve some traditional fractal methods and compounded these techniques to form the proposed no-search scheme. In the proposed no-search scheme, a simple but very efficient similarity measure method was introduced to determine the domain block's position; one technique was designed to reduce the calculation of *rms* error; and we gave an adjusted quadtree scheme and incorporated it to the no-search scheme to improve coding fidelity significantly. Experiments showed that this method could get the fastest speed of fractal image coding up to the present and maintain high reconstruction fidelity. Thus, this research supplied a strong support for the fractal image compression using for real time products.

Furthermore, we can combine the fractal-coding method with some frequency methods such as DCT or wavelet transform to improve the compression ratio and get better results than pure fractal coders and pure transform coders. For example, Zhao and Yuan [22] partitioned the original image to range blocks and domain blocks, then did the DCT for every block (range block and domain block). After that, they did the fractal coding in the frequency space and saved the fractal code as the results. This scheme gained better results than JPEG. Our next research plan is to use this proposed no search scheme to improve the speed of hybrid coders that based on fractal coders and transform coders so as to improve the performance of hybrid coders.

References

[1] B. Bani-Eqbal, Speeding up fractal image compression, Proc. SPIE: Still-Image Compression 2418 (1995) 67–74.
[2] M. Barnsley, Fractal image compression, in: Blackledge J.M. (Ed.), Image Processing: Mathematical Methods and Applications, Clarendon Press, Oxford, 1997, pp. 183–210.
[3] M. Barnsley, A. Sloan, A better way to compress images, BYTE, January (1988) 215–223.
[4] F. Dudbridge, Least squares block coding by fractal functions, in: Y. Fisher (Ed.), Fractal Image Compression: Theory and Application to Digital Images, Springer, New York, 1994, pp. 231–244.
[5] Y. Fisher, Fractal image compression, SIGGRAPH'92 Course Notes 12 (1992) 7.1–7.19.
[6] S. Furao, O. Hasegawa, A fast and less loss fractal image coding method using simulated annealing, Proceedings of Seventh Joint Conference on Information Sciences 2003, Cary, North Carolina, USA, September (2003).
[7] B. Hurtgen, C. Stiller, Fast hierarchical codebook search for fractal coding of still images, SPIE Visual Commun. PACS Med. Appl. (1993) 397–408.
[8] E.W. Jacob, Y. Fisher, R.D. Boss, Image compression: a study of the iterated transform method, Signal Processing 29 (1992) 251–263.
[9] A.E. Jacquin, Image coding based on a fractal theory of iterated contractive image transformations, IEEE Trans. Image Process. 1 (1) (1992) 18–30.
[10] A.E. Jacquin, Fractal image coding: a review, Proc. IEEE 81 (10) (1993) 1451–1465.
[11] J. Li, C.-C. Kuo, Fractal wavelet coding using a rate-distortion constraint, Proceedings of the IEEE International Conference on Image Processing, Lausanne, Switzerland, Vol. 2, 1996, pp. 81–84.
[12] D.M. Monro, F. Dudbridge, Approximation of image blocks, in: Proceedings of the International Conference on Acoustics, Speech, Signal Processing, Vol. 3, 1992, pp. 4585–4588.
[13] D.M. Monro, P.D. Wakefield, Zooming with implicit fractals, Proceedings of the ICIP-97 IEEE International

Conference on Image Processing, Washington, DC, October (1997).
[14] G.E. Øien, S. Lepsøy, A class of fractal image coders with fast decoder convergence, in: Y. Fisher (Ed.), Fractal Image Compression: Theory and Applications, Springer, New York, 1995, pp. 153–174.
[15] D. Saupe, Fractal image compression via nearest neighbor search, in: Conference on Proceedings of NATO ASI Fractal Image Encoding and Analysis, Trondheim, Norway, 1995.
[16] D. Saupe, Lena domain pools for fractal image compression, in: Conference on Proceedings of SPIE Electronic Imaging'96, Still Image Compression II, Vol. 2669, San Jose, CA, 1996.
[17] J. Shapiro, Embedded image coding using zerotrees of wavelet coefficients, IEEE Trans. Signal Process. 41 (12) (1993) 3445–3462.
[18] C.S. Tong, M. Pi, Fast fractal image encoding based on adaptive search, IEEE Trans. Image Process. 10 (9) (2001) 1269–1277.
[19] C.S. Tong, M. Pi, Analysis of a hybrid fractal-predictive-coding compression scheme, Signal Processing: Image Communication 18 (2003) 483–495.
[20] C.S. Tong, M. Wong, Adaptive approximate nearest neighbor search for fractal image compression, IEEE Trans. Image Process. 11 (6) (2002) 605–614.
[21] B. Wohlberg, G.d. Jager, A review of the fractal image coding literature, IEEE Trans. Image Process. 8 (12) (1999) 1716–1729.
[22] Y. Zhao, B. Yuan, A hybrid image compression scheme combining block-based fractal coding and DCT, Signalling Processing: Image Communication 8 (1996) 73–78.