Fractal image coding system based on an adaptive side-coupling quadtree structure

Chwen-Jye Sze\textsuperscript{a}, Hong-Yuan Mark Liao\textsuperscript{b,c,*}, Kuo-Chin Fan\textsuperscript{a}, Ming-Yang Chern\textsuperscript{c}, Chen-Kuo Tsao\textsuperscript{d}

\textsuperscript{a}Institute of Information Engineering, National Central University, Chung-Li, Taiwan
\textsuperscript{b}Institute of Information Science, Academia Sinica, Nanking, Taipei, Taiwan
\textsuperscript{c}Institute of Electrical Engineering, National Chung Cheng University, Chia-Yi, Taiwan
\textsuperscript{d}Institute of Information Engineering, National Chung Cheng University, Chia-Yi, Taiwan

Received 11 April 1995; revised 31 October 1995; accepted 31 October 1995

Abstract

A new fractal-based image compression system, based on a so-called Adaptive Side-Coupling Quadtree (ASCQ) structure, is proposed. The proposed system consists of three processes: a preprocessing, a compression and a decompression process. In the compression process, the original image is represented by an ASCQ structure. The set of Iterated Function System (IFS) codes, which is usually derived in the encoding process, can be calculated directly from this tree structure. Using these IFS codes, an image which is similar to the original one can be reconstructed. Unlike traditional methods, which have separate domain and range pools, the proposed ASCQ structure simultaneously contains the domain pool and range pool. Since the proposed ASCQ is an adaptive structure, the number of IFS codes will be variant depending on their corresponding original images. Experimental results show that the ASCQ structure is indeed an efficient structure for the fractal-based image compression system.

Keywords: Image coding system; Fractal; IFS codes; Quadtree structure

1. Introduction

In the past decade, fractal theory has been successfully applied to the description of natural shapes in many fields. A main branch of fractal theory, the IFS (Iterated Function System), is a powerful tool used in the image compression field. Using the IFS technique, the well-known Barnsely's fern can be generated by using an iterated method with only a few parameters [1,2]. Based on this concept, a good method which applies the IFS theory for efficient encoding of a real image was not developed until the fractal block coding method was proposed by Jacquin [3–5]. In Ref. [3], an image was partitioned into two-level hierarchical sub-blocks with fixed sizes for creating the domain and range pools. After this process, the contractive transformations of the IFS codes can be calculated between the elements in the range pool and those located in the domain pool. To derive the IFS codes, he made the contractive transformations the sum of some elementary block operations. These operations include absorption, gray level scaling and some isometries. The major drawback of Jacquin's approach is that the domain pool and range pool have only two fixed block sizes.

In Jacobs et al.'s [6] work, a quadtree partitioning method was applied to generate range blocks in the range pool. In their approach, the total number of domain blocks and the sizes of these blocks are fixed. At the beginning, the size of each range block is $32 \times 32$. Then, some blocks will be partitioned into four sub-blocks by using a quadtree scheme if no suitable domain blocks can be found in the domain pool. In other words, an original image is used to generate two separate pools, i.e. a domain pool and a range pool. The generated domain pool is a pool containing a finite number of domain blocks, and overlapping is allowed between these blocks. As to the range pool, it consists of a number of non-overlapping, not necessarily fixed-size range blocks generated by a quadtree partitioning scheme.

* Corresponding author. Email: liao@iis.sinica.edu.tw
In the above two methods, the number of domain blocks is fixed for any input image. Although in Jacobs et al.’s [6] method the range blocks were generated by a quadtree partitioning scheme, the generation process was in fact dependent on a predefined error function. Therefore, the two above-mentioned approaches did not make use of the characteristics of an input image at all. However, in the conventional approaches [3–6], the domain pool and range pool are both derived from the same input image. Therefore, we believe that a global search for mapping between these two pools is a waste, because some local relations may be important and useful but never used. From another viewpoint, if an input image contains a large number of smooth regions, and we use conventional methods to partition the image into fixed-size blocks, then the memory load as well as the slow searching efficiency will be the major drawbacks of these systems.

To solve the above-mentioned problems, we propose a new data structure, an Adaptive Side-Coupling Quadtree (ASCQ), to maintain both the range pool and domain pool. All the terminal nodes of this structure form the range pool of the system. Also, all the parent nodes on upper levels form the domain pool. The proposed ASCQ is different from conventional quadtree structures. In this structure, some of its nodes not only represent a single quadrant region, but also link with three side-coupling regions. These side-coupling regions, together with the body regions, form an overlapping domain pool which provides more opportunities for mapping to a best terminal node. The IFS codes can be calculated from the ASCQ structure directly. In our work, the encoding process is divided into two phases. They are the space-partitioning phase and the side-coupling region mounting phase. In the first phase, a quadtree structure is created by partitioning an original image into blocks with different sizes. As to the second phase, the side-coupling region mounting phase, we propose mounting the right, bottom, and diagonal side-coupling regions of each domain node on their corresponding domain nodes based on two pre-determined rules. The purpose of this step is to create an overlapping domain pool so that the subsequent searching process has more opportunities to obtain better results. Since the proposed ASCQ is an adaptive structure, the number of IFS codes will be variant depending on their corresponding original images.

The organization of the rest of this paper is as follows. In Section 2, some basic block operations which are useful in calculating the IFS codes will be introduced. Then, the proposed approach which includes both the compression and decompression processes are thoroughly discussed in Section 3. Some experimental results are reported in Section 4, and finally conclusions are presented in Section 5.

2. Contractive transformations

In this section, we will first introduce the set of elementary block operations proposed by Jacquin [3–5], which will be used in our method. The distortion measure and peak-to-peak signal-to-noise (PSNR) ratio, which can be used to evaluate the performance of an image coding system, will be discussed in Section 2.2.

2.1. Contractive transformations between blocks

In this section, some elementary block operations which are useful in defining the transformations between the range pool and the domain pool will be introduced. Since the square-block nature of the two pools in an ASCQ structure is similar to that of Jacquin’s scheme, the set of block operations proposed by Jacquin [3–5] can be directly applied to our work. In what follows, we shall summarize some useful blockwise contractive transformations [3–5].

(1) Scaling operator $f_s$

In general, a square image block with size $L \times L$ can be written as

$$I(x_0, y_0, L) = \{ (x, y, f(x, y)) | x_0 \leq x \leq x_0 + L, y_0 \leq y \leq y_0 + L \},$$

(1)

where $(x_0, y_0)$ is the coordinate of the left upper corner of this image block. When a square image block is represented as a quadtree structure, the ratio between the size of an arbitrary block and that of a smaller block located in the tree will be $(1/2)^n$, where $n$ represents the level difference between the two blocks. Since Jacquin’s scaling operator restricts the size of a block in the domain pool to four times the size of a block in the range pool, we have to extend the definition so that it can fit our needs. Let $D_0$ and $B$ be the sizes of the domain block and range block, respectively. $f_s$ consists of the discrete form of the blockwise contractive operator $S$, which maps image blocks from a domain block, $D = I(x_d, y_d, D_0)$, to a range block, $R = I(x_r, y_r, B)$. If we use the quadtree partition scheme, the size of $D$ is always equal to the size of $R$ multiplied by an integer $n$. The pixel values in the range block, $I(x_r, y_r, B)$, are the average values of $n^2$ pixels in the domain block:

$$f_s(I(x_r+i, y_r+j)) = \frac{1}{n^2} \sum_{p=0}^{n-1} \sum_{q=0}^{n-1} I(x_d+n-i+p, y_d+n-j+q),$$

for all $i, j \in \{0, 1, \ldots, B-1\}$. (2)

(2) Gray-level operator $f_g$

As to the gray level operator $f_g$, the absorption, luminance shift and contrast scaling are considered.
Therefore, $f_g$ can be defined as follows:

$$f_g(I(x,y)) = \alpha \cdot I(x,y) + \Delta g,$$

where $\alpha$ represents the contrast scaling factor, and $\Delta g$ indicates the luminance shift. For the purpose of processing a digital image, the parameters $\alpha$ and $\Delta g$ must be quantized to some special values. According to Jaquin's concept [3-5], the following conditions should be considered for $f_g$:

(i) when $f_g$ is absorption with respect to gray level,

$$f_g(I(x,y)) = \Delta g = \Delta g_{a},$$

where $\Delta g_{a} \in \{0, \ldots, G_{\max} - 1\};$ (4)

(ii) when $f_g$ is luminance by a constant,

$$f_g(I(x,y)) = I(x,y) + \Delta g = I(x,y) + \Delta g_{l},$$

where $\Delta g_{l} \in \{-G_{\max} + 1, \ldots, 0, \ldots, G_{\max} - 1\};$ (5)

(iii) when $f_g$ is contrast scaling by $\alpha$,

$$f_g(I(x,y)) = \alpha \cdot I(x,y).$$

(3) Reallocation operator $f_{R}$

The reallocation operation includes identity, reflection and rotation. Eight canonical isometries of a square block are listed as follows [3-5],

(1) Identity:

$$f_{R_1}(I(x,y)) = I(x,y).$$

(2) Orthogonal reflection about the mid-horizontal axis $y = (B - 1)/2$:

$$f_{R_2}(I(x,y)) = I(x, B - 1 - y).$$

(3) Orthogonal reflection about the mid-vertical axis $x = (B - 1)/2$:

$$f_{R_3}(I(x,y)) = I(B - 1 - x, y).$$

(4) Orthogonal reflection about the first diagonal $x - y$ of block:

$$f_{R_4}(I(x,y)) = I(B - 1 - x, B - 1 - y).$$

(5) Orthogonal reflection about the second diagonal $x + y = B - 1$ of block:

$$f_{R_5}(I(x,y)) = I(B - 1 - y, B - 1 - x).$$
(6) Rotation around the center of the block, through +90°:

\[ f_{R_9}(I(x,y)) = I(y, B - 1 - x). \]  

(7) Rotation around the center of the block, through +180°:

\[ f_{R_1}(I(x,y)) = I(B - 1 - x, B - 1 - y). \]  

(8) Rotation around the center of the block, through +270°:

\[ f_{R_3}(I(x,y)) = I(B - 1 - y, x). \]

Based on the above operators, the set of contractive transformations between blocks can be calculated.

### 2.2. Distortion measure and PSNR

Since the performance of an image coding system has to be evaluated, the distortion measure and PSNR adopted in our work are defined as follows.

Let \( I_{\text{orig}} \) be an original image with size \( N \times N \) and \( I_{\text{re}} \) be a reconstructed image with the same size. Let \( I_{\text{orig}}(x,y) \) and \( I_{\text{re}}(x,y) \) be the pixels in \( I_{\text{orig}} \) and \( I_{\text{re}} \), respectively. Previously, the root-mean-square (rms) distortion measure was adopted most frequently to evaluate the performance of an image coding system. This measure is defined as follows [3–5]:

\[
\text{rms}(I_{\text{orig}}, I_{\text{re}}) = \sum_x \sum_y [(I_{\text{orig}}(x,y) - I_{\text{re}}(x,y))^2]^{1/2}.
\]  

If an image is reconstructed by a decoding process, its PSNR is usually used to evaluate the quality of a reconstructed image. A commonly used measure is defined as follows:

\[
\text{PSNR} = -20 \log_{10} \frac{\sum_x \sum_y [(I_{\text{orig}}(x,y) - I_{\text{re}}(x,y))^2]^{1/2}}{\sum_x \sum_y I_{\text{orig}}(x,y)}.
\]

In the next section, we shall introduce in detail the procedure for the proposed system.

### 3. Proposed approach

A fractal image coding system based on the proposed ASCQ structure contains two main processes: a compression process and a decompression process. The block diagram of these processes is shown in Fig. 1. In our work, there are four steps involved in the compression process. They are: preprocessing, ASCQ construction, fractal codes construction, and quantization. In what follows, the details of the proposed system will be clearly reported.

#### 3.1. Compression process

**3.1.1. Preprocessing**

Usually, when an image is represented by a quadtree structure, its partition rule most of the time is guided by the characteristics of the image. Whether a node must be further divided depends on the features of its corresponding image block. Previously, Ramamurthi and
Gersho [7] proposed an image classification method. In their method, an image block could be one of the following three types: a shade block, a midrange block or an edge block. A shade block is smooth with no significant gradient. A midrange block has a mordant gradient but no definite edge. An edge block presents a strong change of intensity across a curve — often a piece of object boundary — which runs across the block. From the above discussion, it is obvious that the distribution of gradients is an important indicator for deciding the type of an image block.

Let \((x, y, f(x, y))\) be a pixel in an image and \((x + 1, y, f(x + 1, y)), (x, y + 1, f(x, y + 1)), (x + 1, y + 1, f(x + 1, y + 1))\) be its east, south-east and south neighboring pixels, respectively. The difference values \(d_0, d_1\) and \(d_2\) can be defined as follows:

\[
d_0 = f(x + 1, y) - f(x, y),
\]

\[
d_1 = f(x, y + 1) - f(x, y),
\]

\[
d_2 = f(x + 1, y + 1) - f(x, y).
\]

A differentiated pixel which corresponds to pixel \((x, y, f(x, y))\) is denoted as \((x, y, f^d(x, y))\), where \(f^d(x, y)\) can be represented as

\[
f^d(x, y) = d_{\max},
\]

where \(d_{\max} \in \{d_0, d_1, d_2\}\) and \(|d_{\max}| = \max\{|d_0|, |d_1|, |d_2|\}\). After an image is differentiated, the type of an image block (i.e. shade, midrange or edge) can be determined by analyzing its gradient-based histogram.

Fig. 2 shows a set of gradient-based histograms of different image models, where \(\Delta h\) represents the gradient obtained by taking the derivative with respect to intensity.

3.1.2. Generating ASCQ structure

In this section, we shall describe how an ASCQ structure is generated. Basically, an ASCQ is a new data structure which can keep both the range pool and domain pool in it. All the terminal nodes of this structure form the range pool of the system. Also, all the parent nodes on the upper levels form the domain pool. The proposed ASCQ is different from conventional quadtree structures. In this structure, some of its nodes not only represent a single quadrant region, but also link with three side-coupling regions (Fig. 3(a)).

As shown in Fig. 3(b), the side-coupling regions include the right side, bottom side and diagonal side regions. The black part of each side coupling region belongs to a body region, and the white part belongs to the body region's neighboring region. These side-coupling regions, together with the body regions, form an overlapping domain pool which provides more opportunities for mapping to the best terminal node.

Basically, the side-coupling regions are not always mounted on a body node, for example, when a body node is a terminal node or when it and its neighboring nodes are all non-edge nodes. The partition rule of an ASCQ is based on the geometric features contained in each body node. In this paper, we use the means and variances of differentiated image blocks as the geometric features.

The proposed procedure for creating an adaptive side-coupling quadtree structure can be divided into two phases. They are explained in detail below.

Space-partition phase

The purpose of this phase is to create an adaptive quadtree structure from an original image. This image, which is organized based on the requirement of an ASCQ structure, will be represented by a number of square sub-image blocks with different sizes. The partition rule is basically the same as that of other quadtree structures. However, two different terminal conditions are proposed to decide whether a body node should be further divided or not. In what follows, the two terminal conditions are described:

- (1) If the size of a body node is less than or equal to the default minimum size.
- (2) If a body node is non-edge, and any body node at its upper level is also non-edge.

Side-coupling region mounting phase

We have stated that the terminal nodes of an ASCQ form the range pool, i.e. these nodes correspond to a
set of target blocks which are to be matched with a set of domain blocks (contained in the parent nodes). Since an overlapping domain pool is required, three types of aforementioned side-coupling regions will be mounted on some nodes of the quadtree structure. Basically, it is not necessary for every node on an ASCQ to be mounted with side-coupling regions. The following rules can be used to decide whether a body node should be mounted with side-coupling regions or not.

1. If a body node corresponds to an image block which touches the right boundary or the bottom boundary of the image, it is not mounted with side-coupling regions.
2. If the size of a body node is less than or equal to a default minimum size, this node does not need to be mounted with side-coupling regions.
3. If a body node and its neighboring body nodes are all shade blocks or midrange blocks, this node does not need side-coupling regions.

3.1.3. Calculating fractal codes
After an ASCQ structure is created from an image, the next step is to calculate the fractal codes directly from this structure. Fig. 4 is a flowchart which shows how the fractal codes are calculated correctly. The input to this procedure is an ASCQ structure. There are three types of directed paths contained in the flowchart. The wide arrows, middle arrows, and thin arrows represent, respectively, the transmission paths of image blocks, the contractive transformations and the control signals. After the contractive transformations of all the
terminal nodes are recorded, these results will be sent to the next procedure — the quantization procedure.

In what follows, we will explain how the contractive transformations between the range pool and the domain pool can be calculated directly from an ASCQ structure (see Fig. 5). First, let \( I_R \), be a terminal node in a constructed ASCQ structure. The subscript \( R \) means that the node is a member of the range pool, and the superscript \( l \) is the depth (level number) of the node. Let \( D_R \) be the differentiated result of \( I_R \) (i.e., differentiate block \( I_R \) with respect to intensity). Then, by calculating the means and variances from \( I_R \) and \( D_R \), we can determine which category \( I_R \) belongs to. If the mean and variance of the intensity of block \( I_R \) are both close to zero, then \( I_R \) can be identified as a shade block and directly sent to the shade processor. If other conditions occur (e.g., midrange or edge), then \( I_R \) and its corresponding features are sent to a feature comparator for further consideration. The comparator is designed to decide whether a node in the domain pool is similar to the terminal node \( I_R \) in the range pool. When a terminal node \( I_R \) at level \( l \) is selected, all the body nodes and side-coupling nodes on the constructed ASCQ with level values less than \( l \) must be searched. Let \( k \) be a level number and \( k < l \). As shown in Fig. 5, let \( I_D \) be an input block node (in the domain pool) at level \( k \). If node \( I_R \) was previously identified as not being a shade block, and the features derived from node \( I_R \) are very close to those of node \( I_D \), then \( I_D \) will be sent to the midrange or edge processors, respectively, based on the feature types of \( I_R \). In our design, an overlapping threshold range between the midrange block and the edge block is allowed. Therefore, it is possible for both \( I_R \) and \( I_D \) to be sent to the midrange and the edge processors simultaneously.

In a shade processor (see Fig. 6(a)), contractive transformation is directly derived by calculating the average gray level of the pixels contained in node \( I_R \). In fact, this value does not need to be calculated because the mean value of \( I_R \) has already been calculated in a previous step. Therefore, the output of the shade processor is the mean of input \( I_R \) directly. When both \( I_R \) and \( I_D \) are sent to the midrange processor (see Fig. 6(b)), the contractive transformations can be derived by composing a scaling, a contrast scaling, and a luminance shift [3–5]. The equation of this composition can be represented as follows:

\[
\text{fr}(f_r(I_R(x, y))) = \alpha_m \cdot f_r(I_R(x, y)) + \Delta g_l,
\]

where \( \Delta g_l = \text{mean}(I_R) - \alpha_m \cdot \text{mean}(I_D) \). \( \alpha_m \) is a midrange contrast scaling factor chosen from one of the eight quantized values illustrated in Table 1 with a minimum error distortion measure between \( I_R \) and \( f_r(I_R(x, y)) \). The derived contractive transformation as well as the error value will be sent to the CT buffer shown in Fig. 4.

When an image block is identified as an edge block, the contractive transformation is the composition of a scaling, a contrast scaling, a luminance shift, and some reallocations (see Fig. 6(c)). This operation can be expressed as follows:

\[
\text{fr}(f_r(S_D(x, y))) = \text{fr}(\alpha_e \cdot S_D + g_t).
\]

where \( S_D(x, y) = f_r(I_D(x, y)) \). \( \Delta g_l \) and \( \alpha_e \) can be calculated by the following equations, respectively:

\[
\Delta g_l = \frac{I_R}{\text{mean}(I_D)}
\]

\[
\alpha_e = \frac{\text{mean}(I_R)}{\text{mean}(I_D)}.
\]

When the contrast scaling factor, \( \alpha_e \), is out of the range (0.3, 1.9), block \( I_D \) must be abandoned. If \( \alpha_e \) is within the above range, then the one among the eight isometries \( \{f_r_{\alpha_e}\} \) that minimizes the distortion measure is selected. Table 1 shows different ranges of these parameters for different image models.

3.1.4. Quantization procedure

When the set of contractive transformations is extracted from an ASCQ, every parameter of the codes must be transformed into a bit sequence by using a
Fig. 6. Processors of three different image models. (a) Shade processor, (b) midrange processor, (c) edge processor.

The purpose of this step is to simplify the format of the compressed data. Quantization is a many-to-one mapping and, therefore, is fundamentally lossy. In order to avoid this major loss in the encoding process, Jacquin [3-5] only uses a few quantized values in the IFS codes construction process. That is, the quantization process is embedded in the IFS code construction process. In our design, the parameters

<table>
<thead>
<tr>
<th>Image Block Type</th>
<th>Parameters of contraction transformation</th>
<th>Corresponding ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shade block</td>
<td>Absorption</td>
<td>$\Delta g_e \in {0, \ldots, G_{max}}$</td>
</tr>
<tr>
<td>Midrange block</td>
<td>Midrange Contrast scaling</td>
<td>$\alpha_m \in {2, 4, 5, 7, 8, 9, 1, 1.1}$</td>
</tr>
<tr>
<td></td>
<td>Luminance shift</td>
<td>$\Delta g_l \in {-G_{max}, \ldots, G_{max}}$</td>
</tr>
<tr>
<td>Edge block</td>
<td>Edge Contrast scaling</td>
<td>$\alpha_e \in {3, 4, 5, \ldots, 1.5, 1.6}$</td>
</tr>
<tr>
<td></td>
<td>Luminance shift</td>
<td>$\Delta g_l \in {-G_{max}, \ldots, G_{max}}$</td>
</tr>
<tr>
<td></td>
<td>Isometries ${f_R}$</td>
<td>$0 \leq i \leq 7$</td>
</tr>
</tbody>
</table>
the IFS codes are quantized in a separate quantization process rather than in the IFS code construction process.

Every IFS code extracted from the previous process should include the image model type, range node allocation, corresponding domain node allocation, absorption $\Delta g_a$, luminance shift $\Delta g_l$, midrange contrast scaling $\alpha_m$, or edge contrast scaling $\alpha_e$, and isometry $f_R$. Because the order of an IFS code sequence is independent of the decoded results, we can sort the order of IFS codes based on their range node allocation $(R_x, R_y)$. Under these circumstances, the allocation of each ordered range block can be decided only by ordering of the block sizes. Therefore, the information of range block allocation can be neglected when the IFS codes have been sorted.

In what follows, we will summarize the formats of three different image models:

(i) the format of a shade image block

\[ M_t R_s \Delta g_a \]

(ii) the format of a midrange image block

\[ M_t R_s D_x \Delta g_l \alpha_m D_x \]

(iii) the format of an edge image block

\[ M_t R_s D_x \Delta g_l \alpha_e D_y f_R \]

These formats are enough to represent all the kinds of IFS codes.

Let the size of an input image be $2^n \times 2^n$ and the minimum default size of a range node be $2^m \times 2^m (m < n)$. Further, let the maximum grey level, $G_{\text{max}}$, be $2^l - 1$, and the midrange contrast scaling, $\alpha_m$, be quantized to eight different values (0.2, 0.4, 0.5, 0.7, 0.8, 0.9, 1.0 and 1.1). Further, let the edge contrast scaling, $\alpha_e$, be quantized to 16 different values (0.3, 0.4, 0.6, ..., 1.6, 1.7, 1.8). If the
Default minimum block size: 4 × 4

Fig. 8. The number of domain blocks and range blocks versus the edge threshold.

Fig. 9. PSNR versus edge threshold with default minimum sizes 4 × 4 and 8 × 8.

The size of a domain block is less than that of an input image (2^n × 2^n) and is larger than the minimum default size of a range block (2^m × 2^m), then (n - m) bits are required to represent the allocations of a domain block. The possible sizes of a range block and a domain block are limited to the ranges (2^n-2, 2^n-3, ..., 2^n) and (2^n-1, 2^n-2, ..., 2^m+1), respectively. Therefore, the total number of bits required to represent both D_s and D_t is \( \left\lfloor \log_2(n - m - 1) \right\rfloor + 1 \) bits at least. In sum, the total number of constructed IFS codes is \( \mathcal{P} = \mathcal{P}_s + \mathcal{P}_m + \mathcal{P}_e \), where \( \mathcal{P}_s, \mathcal{P}_m \) and \( \mathcal{P}_e \) represent, respectively, the number of shade blocks, midrange blocks and edge blocks. The total number of these IFS codes and the corresponding compression ratio can be calculated by the following formula:

\[
\text{Total bits} = \mathcal{P}_s \cdot [\text{Bit}(M_s) + \text{Bit}(R_s) + \text{Bit}(D_s)] + \mathcal{P}_m \cdot [\text{Bit}(M_m) + \text{Bit}(R_m) + \text{Bit}(D_m)] + \mathcal{P}_e \cdot [\text{Bit}(M_e) + \text{Bit}(R_d) + \text{Bit}(D_e)] + \text{Bit}(\alpha_m) + \text{Bit}(\alpha_e) + \text{Bit}(f_{R_s}) + \text{Bit}(f_{R_d}) + \text{Bit}(f_{D_s}) + \text{Bit}(f_{D_e}) - \mathcal{P}_s \cdot [2 + \left\lfloor \log_2(n - m - 1) \right\rfloor + 1] + \mathcal{P}_m \cdot [2 + 2 \cdot \left\lfloor \log_2(n - m - 1) \right\rfloor + 3 + (l + 1) + 2 \cdot \left\lfloor \log_2(n - m - 1) \right\rfloor + \mathcal{P}_e \cdot [2 + 2 \cdot \left\lfloor \log_2(n - m - 1) \right\rfloor + 4 + (l + 1) + 2 \cdot \left\lfloor \log_2(n - m - 1) \right\rfloor + 3], \tag{25}
\]

where the function \( \text{Bit}(x) \) returns the least number of bits required for representing the value x, and

\[
\text{compression ratio} = \frac{2^{2n} \times l}{\text{Total bits}}. \tag{26}
\]

Table 2

Performance of the proposed method on the set of test images with edge threshold 4.0 and default minimum size 4 × 4

<table>
<thead>
<tr>
<th>Image</th>
<th>PSNR (dB)</th>
<th>Compression ratio</th>
<th>No. of domain blocks</th>
<th>No. of range blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>34.68</td>
<td>5.48</td>
<td>4430</td>
<td>3364</td>
</tr>
<tr>
<td>Peppers</td>
<td>31.95</td>
<td>5.01</td>
<td>4667</td>
<td>3598</td>
</tr>
<tr>
<td>Mandrill</td>
<td>25.20</td>
<td>4.27</td>
<td>5166</td>
<td>4048</td>
</tr>
<tr>
<td>Fractal-generated clouds</td>
<td>36.83</td>
<td>4.41</td>
<td>5085</td>
<td>3967</td>
</tr>
</tbody>
</table>

Edge threshold \( Et = 4.0 \) and default minimum size \( Ms = 4 \times 4 \).
3.2. Decompression process

In the previous sections, we have explained how the IFS codes can be constructed from an original image and quantized to the digital sequences. When these compressed data are received from the network or a storage disk, an original image can be reconstructed after performing a dequantization process followed by an iterated reconstruction process using these data. Because a dequantization procedure is an inverse process of a quantization procedure, it can be derived easily. A set of IFS codes can be extracted from its corresponding digital file after the dequantization procedure. The original image can be reconstructed by these IFS codes in the image reconstruction procedure. From the inequality equation in Refs. [1,2], we know that the initial input image is independent of the final reconstructed image if the iteration number is large. Thus, the initial image in the decoding procedure can be any image. For a general iterated decoding process, all the pixels in the current image are generated by the pixels of the previous image calculated using all of the IFS codes. After several iterations, the terminal condition can be detected by calculating the differences between the generated image and the previous one. If the change of the differences after each iteration is less than a threshold, then the image is considered to be completely reconstructed.

![Graph](image)

Fig. 10. Compression ratio versus edge threshold with default minimum sizes 4 × 4 and 8 × 8.

![Images](image)

Fig. 11. Reconstructed images with edge threshold 4.0 and default minimum size 4 × 4.
Table 3
Performance of the proposed method on the set of test images with edge threshold 8.0 and default minimum size $4 \times 4$.

<table>
<thead>
<tr>
<th>Image</th>
<th>PSNR (dB)</th>
<th>Compression ratio</th>
<th>No. of domain blocks</th>
<th>No. of range blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>32.85</td>
<td>8.28</td>
<td>3026</td>
<td>2152</td>
</tr>
<tr>
<td>Peppers</td>
<td>31.06</td>
<td>6.47</td>
<td>3716</td>
<td>2743</td>
</tr>
<tr>
<td>Mandrill</td>
<td>25.06</td>
<td>4.71</td>
<td>4730</td>
<td>3652</td>
</tr>
<tr>
<td>Fractal-generated</td>
<td>33.40</td>
<td>14.74</td>
<td>1915</td>
<td>1183</td>
</tr>
</tbody>
</table>

* Edge threshold $E_t = 8.0$ and default minimum size $M_s = 4 \times 4$.

4. Experimental results

A number of test images, shown in Fig. 7, were used to test the effectiveness of the proposed method. All the test images were of size $256 \times 256$. We used several measures to evaluate the performance of the proposed system. These measures included the number of domain blocks, the number of range blocks, PSNR, and the compression ratio. The number of domain blocks and range blocks was counted in the compression process. After a test image was reconstructed from the IFS codes, PSNR and the compression ratio were calculated and used to evaluate the performance of the system. As described in Section 3.1.2, the default minimum size $M_s$ and the edge threshold $E_t$ have to be predetermined before a test image is input to the encoding system. As to this part, we have used the Lena image to analyze the characteristics of these parameters.

In a fractal-based coding system, the bottleneck of computation time is in deriving the IFS codes. Before
Table 4
Performance of the proposed method on the set of test images with edge threshold 4.0 and default minimum size 8 × 8a

<table>
<thead>
<tr>
<th>Image</th>
<th>PSNR (dB)</th>
<th>Compression ratio</th>
<th>No. of domain blocks</th>
<th>No. of range blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>30.67</td>
<td>21.79</td>
<td>1175</td>
<td>948</td>
</tr>
<tr>
<td>Peppers</td>
<td>28.01</td>
<td>20.71</td>
<td>1202</td>
<td>979</td>
</tr>
<tr>
<td>Mandrill</td>
<td>22.62</td>
<td>19.41</td>
<td>1245</td>
<td>1024</td>
</tr>
<tr>
<td>Fractal-generated</td>
<td>34.60</td>
<td>19.68</td>
<td>1245</td>
<td>1024</td>
</tr>
</tbody>
</table>

* Edge threshold $E_t = 4.0$ and default minimum size $M_s = 8 \times 8$.

These codes can be derived, each range block has to search for a very 'similar' domain block in the domain pool. Therefore, the computation time is definitely dependent on the number of domain blocks and range blocks. Fig. 8 shows how the number of domain blocks and range blocks changed with respect to different edge thresholds. The default minimum block size was set to $4 \times 4$ and $8 \times 8$ in Figs. 8(a) and (b), respectively. From Fig. 8, it is obvious that when the edge threshold increased, both the number of domain blocks and that of range blocks decreased. As to the compression ratio, since this ratio is dependent on the number of IFS codes, and the number of IFS codes is dependent on the number of range blocks, the compression ratio is dependent on the number of range blocks. Figs. 9 and 10 illustrate the PSNR versus the edge threshold and the compression ratio versus the edge threshold, respectively. Both figures contain two different curves which, respectively, correspond to two different default minimum block sizes (i.e. $4 \times 4$ and $8 \times 8$). From Fig. 9, it is seen that when the edge threshold increases, the value of PSNR decreases for both minimum block sizes. As to the compression ratio, when the edge threshold increases, both curves in Fig. 10 climb with different rates (the $8 \times 8$ mini-

Fig. 13. Reconstructed images with edge threshold 4.0 and default minimum size 8 × 8.
Table 5
Performance of the proposed method on the set of test images with edge threshold 8.0 and default minimum size 8 x 8a

<table>
<thead>
<tr>
<th>Image</th>
<th>PSNR (dB)</th>
<th>Compression ratio</th>
<th>No. of domain blocks</th>
<th>No. of range blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>29.89</td>
<td>26.61</td>
<td>977</td>
<td>760</td>
</tr>
<tr>
<td>Peppers</td>
<td>27.77</td>
<td>23.15</td>
<td>1102</td>
<td>871</td>
</tr>
<tr>
<td>Mandrill</td>
<td>22.59</td>
<td>20.16</td>
<td>1208</td>
<td>985</td>
</tr>
<tr>
<td>Fractal-generated clouds</td>
<td>33.26</td>
<td>30.63</td>
<td>881</td>
<td>658</td>
</tr>
</tbody>
</table>

a Edge threshold $E_t = 8.0$ and default minimum size $M_s = 8 \times 8$.

From the results shown in Figs. 9 and 10, it is obvious that the selection of a best edge threshold is, in fact, a compromise between PSNR and the compression ratio. If one wants a high compression ratio, then the value of PSNR will be sacrificed simultaneously. On the other hand, an encoding scheme with high PSNR will lose part of its compression capability.

In the experiments, we used two different edge thresholds, 4 and 8, to evaluate the performance of the system. The default minimum sizes of Tables 2 and 3 were both set to $4 \times 4$. In Tables 4 and 5, the default minimum size was set to $8 \times 8$. The edge threshold was set to 4.0 for all the examples in both Tables 2 and 4. Further, the edge threshold was set to 8.0 for all the examples in Tables 3 and 5. Figs. 11-14 are the reconstructed images which correspond to the data shown in Tables 2-5, respectively. It is obvious that the compression ratio for the cases shown in Tables 4 and 5 is higher than those shown in Tables 2 and 3. On the other hand, the number of domain blocks or range blocks shown in Tables 4 and 5 is always less than...
the examples shown in Tables 2 and 3. From these experiments, we found that the results were coincident with what was expected.

5. Concluding remarks

In this paper, we have proposed a fractal image coding system based on an adaptive side-coupling quadtree structure. In this structure, both the range blocks and domain blocks are included. This kind of arrangement is totally different from previous approaches. When an image is input to this encoding system, both the range blocks and the domain blocks are adaptively created by the characteristics of the input image. The number of range blocks is dependent on the number of domain blocks in an ASCQ structure. The advantage of the proposed scheme is that, when an input image is a smooth one, the number of IFS code will be significantly reduced. In the ASCQ construction process, the edge threshold as well as the default minimum size are the most important parameters. Together, they can influence the number of domain nodes and range nodes in an ASCQ structure. Therefore, two performance measures, i.e. PSNR and compression ratio, are dependent on the two parameters. The drawback of the system is that the two above mentioned parameters cannot be determined on line.

References