REGION-BASED FRACTAL COMPRESSION FOR STILL IMAGE

Yung-Ching Chang, Bin-Kai Shyu, Jia-Shung Wang

Department of Computer Science, National Tsing Hua University,
Hsinchu, Taiwan 30043, Republic of China
e-mail: dr804339@cs.nthu.edu.tw (Yung-Ching Chang)
jswang@cs.nthu.edu.tw (Jia-Shung Wang)

ABSTRACT

Fractal image coding is a novel and attractive technique for still image compression. By utilizing the characteristic of self-similarity, an iterated function system can automatically converts an image into a set of affine transformation coefficients. However, the conventional block-based segmentation methods inadequately satisfy the natural image property and thus can't achieve an efficient performance. In this paper, we propose a thorough fractal image compression system to approach the target of very low bit-rate. To more efficiently utilize the property of natural images, an image dependent region-based segmentation technique is proposed. This region-based process consists of two steps: First, we improve the performance of quadtree decomposition by utilizing the adaptive threshold method. Second, a merging scheme is introduced to the result of quadtree decomposition that combines several similar blocks into a small number of regions. We also provide a quadtree-based segmented chain code to efficiently record the contours of the regions. Moreover, a post-processing algorithm is applied according to region-based segmentation to eliminate the blocking artifact. The experimental results indicate that the proposed method has the potential to achieve comparable extreme low bit rate among the existing method at the same level of quality.

Keywords: fractal image compression, region-based method, very low bit-rate.

1. INTRODUCTION

Fractal image compression, which is based on the *iterated function system* (IFS), is a novel and attractive approach to image coding [Barns88a] [Fishe94a]. Fractal image compression encodes an image by specifying the parameters of a *contraction mapping* with a unique *fixed point*, which approximates the original image. To recover the approximation, the mapping is iterated from an arbitrary initial point until converges to the fixed point. The typical fractal image coding methods [Jacqu92a] initially partition the original image into some non-overlapped range blocks, and then approximate each range block with a contractive affine transformation of a domain block, which are selected from the same image with different size. Under the assumption that there has self-similarity between range and domain blocks in an image, the encoding problem essentially reduces to a block matching problem. For each range block, a search is performed within a pool of candidate domain blocks to obtain the one that yields the best approximation. The coefficients of the affine transformation for each range block are then quantized and stored. At the decoding side, an arbitrary initial image is partitioned as done at the encoding size, and then the decoded coefficients of the affine transformation are applied iteratively for each range block. Base on the constraint of contractivity, the decoded image is gradually similar to the original one.

A number of factors can alter the performance of the fractal image compression method. The most critical one is the segmentation of image. Because there is a set of coefficients of the affine transformation for each range block, the segmentation result that can maintain the same level of distortion with fewer range blocks thus achieves better performance. Second, because some of the coefficients are real, an efficient quantization design is required.

By observing the behavior of some fractal image coding methods, we realize that the uniform block-wise segmentation of image inadequately satisfies the property of natural images. Therefore, the image segmentation with various block size, include quadtree decomposition, is introduced into

fractal image compression in some publications [Jacks97a]. With the quadtree decomposition, the finer range blocks can achieve better match of the rough area of image while larger range blocks can match the smooth area of image with lower bit rate. Unlike other researches fixing the threshold for each level, an adaptive threshold scheme [Shust94a] is utilized here to obtain performance improvement.

In addition to the quadtree decomposition, we attempt to construct a region-based segmentation scheme to satisfy the characteristics of natural images. This region-based process consists of two steps: First, the image is segmented by quadtree decomposition with adaptive threshold. Second, a merging scheme is introduced to the result of quadtree decomposition that combines several similar blocks into a small number of regions. Because there is now only a set of coefficients for a region instead of some sets of coefficients for the separated blocks, the bit rate can be further reduced. Finally, the coefficients are quantized and then entropy coded to reduce bit rate further.

The trade-off of the region-based segmentation is the requirement of extra bit rate to record the shape boundary of each region. Here, we provide a so-called quadtree-based segmented chain code to efficiently record the contours of the regions. At the decoder, because of the significantly degradation of image quality with high compression ratio, we also provide a post-processing algorithm according to region-based segmentation to eliminate the possible blocking artifacts.

In this paper, the brief introduction of fractal block coding is succinctly presented in section 2. In section 3, we introduce the quadtree decomposition scheme and the process of adaptive quadtree decomposition based fractal coding. Section 4 describes the region-based fractal image coding. The post-processing for region-based fractal image compression is provided in section 5. Section 6 summaries some experimental results. Finally, some conclusions are drawn in section 7.

2. FUNDAMENTAL OF FRACTAL BLOCK CODING

Before describing the delicate variation of fractal image coding scheme, we give a typical example to introduce the process of fractal block coding, which is the essential implementation of the fractal image coding.

In fixed-size block segmentation, an $N \times N$ image is partitioned into several uniform non-overlapped square block of size $r \times r$, where r is a factor of N. These blocks are called *range* blocks. We define an affine transform w of the from:

$$w\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & s \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e \\ f \\ o \end{pmatrix} \quad (1)$$

is a contractive mapping with contractivity factor $|s|$ and $0 \le |s| < 1$. For a range block R_i, find a block with size $2r \times 2r$ from other parts of the same image, we name it as *domain* block D_i, to minimize the distortion $d(w(D_i), R_i)$. For each range block, there are four *fractal coefficients* to be decided: s is the contrast factor, o is the brightness factor, e and f is the position of the best-matched domain block. In generally, the position distribution of the best-match domain block, the center position of the range block shows a slightly higher probability among all the positions. Therefore, we assign a search range $(L+2r) \times (L+2r)$ centered at the range block, and the horizontal and vertical step size is δ_h and δ_v, respectively, as illustrated in Fig. 1. The coefficient e and f are now used to represent the relative position between R_i and D_i. Fig. 2 depicts an example of fractal block coding.

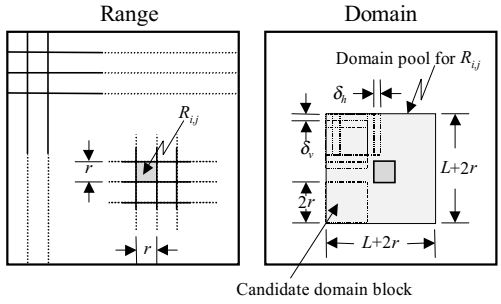


Fig. 1. An example of domain pool

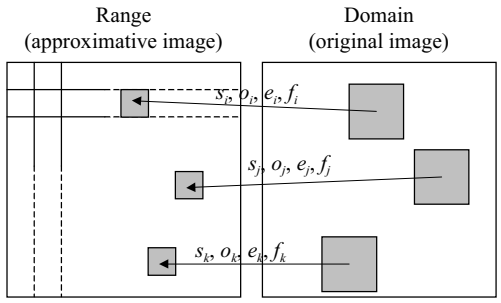


Fig. 2. The coding process of a fractal block coder

3. ADAPTIVE QUADTREE DECOMPOSITION FOR FRACTAL IMAGE CODING

The limitation with fixed sized square range blocks is that it does not take advantage of the contents of image. In typical images, there are range blocks for which it is impossible to find matching domain blocks within a certain distortion tolerance when the fixed range block size is used. Similarly, there are regions where larger range blocks could be used and their matching domain blocks could still be found.

This implies that higher compression performance may be achieved if variable size range blocks are used. Quadtree decomposition is a common method to partition image into variable size of block, since its flexibility and less overhead. There is some studies [Jacks97a] utilized quadtree segmentation into fractal image coding and obtained somewhat improvement. However, as that will be discussed in this section, a unique distortion threshold for various block sizes wouldn't be an intelligent choice.

3.1. Quadtree representation and coding

A *quadtree* is a tree in which each node potentially has four subnodes. Quadtree *decomposition* is an image represented by a *quadtree*: the root of this tree is the original image, and each node of the tree, except root, corresponds to a square that is a quadrant of its parent's square. Fig. 3 (a) is an example of quadtree decomposition, and Fig. 3 (b) is the corresponding quadtree representation.

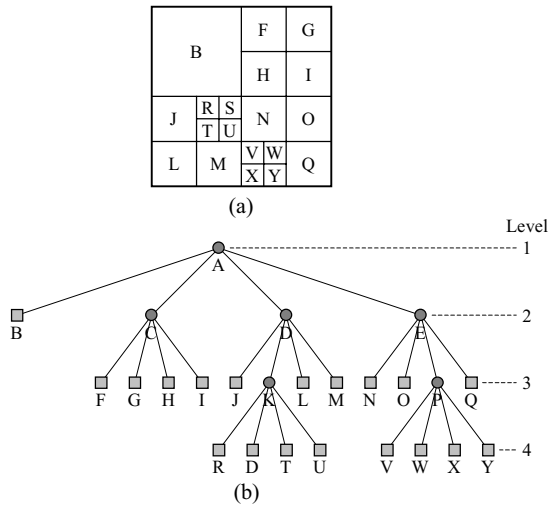


Fig. 3. (a) An example of quadtree decomposition, and (b) the corresponding quadtree representation

To store and transmit the quadtree structure, one simple way is to traverse the tree through depth-first search. Mark "1" if a leaf node is visited, and mark "0" for an internal node. Considering Fig. 3, the traversal of the quadtree by depth-first search is

10100001010000001100100000.

This code is usually denoted as *quadtree code*.

An image can be represented by a quadtree decomposition of any given quadtree size, in this paper, the size is determined by the coding quality concerned. We adjust the quadtree size by using two parameters: *minimum level* and *maximum level*. The assigning of minimum level means that the image at least has to be decomposed to this level, and therefore we have only to record the quadtree code from the minimum level but not from the root. The assigning of maximum level means that the image will not be decomposed into smaller blocks than the corresponding block size of maximum level, and therefore we can ignore the quadtree code for the maximum level. By given the minimum level = 2 and maximum level = 4, the reduced quadtree code for Fig. 3 becomes

0100001010010010.

3.2. Quadtree decomposition algorithm with adaptive threshold

In our implementation of employing the quadtree decomposition into fractal image coding, the first step is to decompose the image to the minimal level of the quadtree (i.e. the largest leaf node). For each leaf node (a range block), find the best fractal coefficients with the minimal MSE over its domain pool. If this minimal error is larger than a pre-assigned threshold e and if the level of the node is less than the maximum level, we split this leaf node into four quadrant subnodes, and repeat the above process until the maximum level is augmented.

In conventional compression methods that employee quadtree decomposition scheme, the distortion threshold is given uniquely over the all levels. Shusterman and Feder [Shust94a] proposed a scheme of compressing image via quadtree decomposition. They proved that if we use adaptive thresholds on quadtree levels, the coding quality would be better than that of a fixed threshold at the same bit rate. If we choose an initial threshold e_1 on the first level, a suboptimal threshold for the next level is determined as below:

$$e_2 = 2 \cdot e_1, \qquad (2)$$

where e_i is the threshold of level i. If the *initial threshold* on level 1 is known, we can rewrite (2) as follows

$$e_i = 2^{i-1} \cdot e_1. \qquad (3)$$

We call the threshold on level 1 as *initial threshold*. Suppose that the quadtree threshold is of the form

$$e_i = k \cdot e_{i-1}. \qquad (4)$$

When k =1, it is the conventional situation that the threshold is fixed for all levels; when k = 2, it is the adaptive threshold that proposed by Shusterman and Feder.

The experimental results for various threshold assignment indicate that the adaptive threshold method with the factor k = 2 is the best one over various bit rate. In general, if the allocated bits for each leaf nodes remain the same over different levels, the assignment of factor k = 2 can achieve the best image quality.

4. REGION MERGING SCHEME

In addition to the quadtree decomposition proposed in the previous section, we consider a more flexible

segmentation of the original image, which can take advantage of the contents of image better than square blocks. However, an arbitrary region-based segmentation very tough to implement and require a large amount of overhead on coding the shape of regions. In this section, we propose a two-step technique to approach the region-based concept. First, an image is quadtree decomposed by utilizing the adaptive threshold method. Next, we design a merging scheme to the resulting quadtree decomposition for combining several similar blocks into a small number of regions. This technique maintains the balance between the image quality and coding bit rate.

4.1. Merging scheme

Suppose I is an image of size $N \times N$, and I is segmented by quadtree to a set of one or several blocks $Q = \{B_1, B_2, \ldots, B_n\}$. We call R a *quadtree-based region* (or *region* in short) if R is the union of a subset of Q with the constraints that R is connected.

In region-based fractal image compressing, the range of each affine transformation w in the iteration function system is a region. The mapping w between the domain and range of is illustrated in Fig. 4.

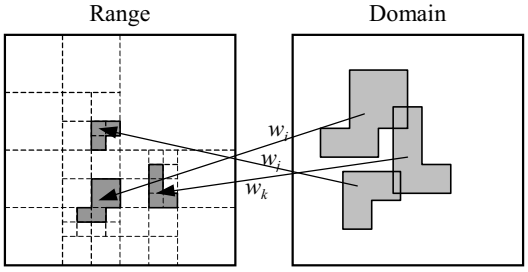


Fig. 4. The mapping of region-based affine transformations

Let $R = \{R_1, R_2, \ldots, R_n\}$ be a set of regions, which is denoted the segmentation of original image. For each region-pair (R_i, R_j) in $R \times R$, if R_i and R_j are adjacent, let R_m the region merged by R_i and R_j geometrically, denoted by $R_M = (R_i, R_j)$, we can find the coefficients of iteration function system and minimal error ε_M. If there is no any domain cell corresponding R_M, or if R_i and R_j are not adjacent, we regard the minimal error ε_M as infinity. The total mean square error ε_T of $R \times R$ is the sum of all error ε_{Mi} of R_{mi}, except infinity value, in $R \times R$.

The basic consideration for region merging is to split the image to finer segmentation with quadtree decomposition, then employee a merging scheme to merge adjacent blocks with similar property into regions. For each merging action, the total error increase and the bit rate decrease. It seems that we can assign a target PSNR or bit rate to indicate the completion of merging. However, as we will describe in the following section, the total bit rate consist of quadtree code, fractal coefficients and region boundary code, it is more difficult to achieve a pre-assigned target bit rate than PSNR. Therefore, in our implementation of the merging algorithm, the region is merged until the total MSE grow beyond a pre-assigned threshold.

Before the merging start, we assign a threshold T, and segment the image with adaptive quadtree decomposition to an initial segmentation R with MSE smaller that T. The first step of merging process is to find the error for each pair (R_i, R_j). Next, find out the pair $R_M = (R_i, R_j)$ in $R \times R$ with minimal error and then update R by removing R_i and R_j from R and adding R_m into R, recalculate the MSE of $R \times R$. Third, repeat the last step until the MSE ε_T is great than threshold T. We call the threshold T as *merging threshold*. For convenient, we will assign a PSNR value as the target of merging, though this value can be easily converted to MSE.

The following algorithm is implemented in our experiment:

Step 1. Do quadtree decomposition and let $R = \{R_1, R_2, \ldots, R_n\}$ is the decomposition result and E is the total error of the decomposition.

Step 2. Convert the target PSNR P to total error T. If $E > T$, exit the procedure.

Step 3. For each region R_i in R, find out the set of adjacent regions $A_i = \{R_a, R_b, \ldots\}$. Find the minimal error ε_{ix} for each region pair (R_i, R_x), $x = a, b, \ldots$. Let ε_i and ε_x are the error of R_i and R_x, respectively. Calculate the error increment $\Delta\varepsilon_{ix} \leftarrow (\varepsilon_{ix} - \varepsilon_i - \varepsilon_x)$.

Step 4. Find the region pair (R_k, R_k') with mininal error increment $\Delta\varepsilon_{kk'}$ over all region pairs.

Step 5. Let $E \leftarrow (E - \varepsilon_k - \varepsilon_{k'} + \varepsilon_{kk'})$. If $E > T$, exit the procedure.

Step 6. Let $R_M \leftarrow R_k + R_k'$.

Step 7. Remove R_k and R_k' from R, and add R_M into R.

Step 8. For each region R_{Ak} in A_k (i.e. the region that is adjacent to R_k but is not R_k'), find the minimal error ε_{Akm} for region pair (R_{Ak}, R_m). Calculate the error increment $\Delta\varepsilon_{Akm} \leftarrow (\varepsilon_{Akm} - \varepsilon_{Ak} - \varepsilon_m)$.

Step 9. For each region R_{Ak}' in A_k' (i.e. the region that is adjacent to R_k' but is not R_k), find the minimal error $\varepsilon_{Ak'm}$ for region pair (R_{Ak}', R_m). Calculate the error increment $\Delta\varepsilon_{Ak'm} \leftarrow (\varepsilon_{Ak'm} - \varepsilon_{Ak'} - \varepsilon_m)$.

Step 10. GOTO Step 4.

After the merging process, some quadtree blocks with similar property are merged into regions. If three blocks combine a region, we have only to record a set of fractal coefficients instead of three sets. The fractal coefficients are further encoded by

the adaptive arithmetic coding. Moreover, in the region-based implementation, we have not only the extra bits to represent the structure of quadtree decomposition, but also another bits to record the combination of blocks for each region.

4.2. Adaptive arithmetic coding for fractal coefficients

Utilizing entropy coding can further save the bit rate for the fractal coefficients of regions. The *Arithmetic Coding* [Rubin79a] is a more efficient entropy coding tool than Huffman coding. In this section, the *Adaptive Arithmetic Coding* [Witte87a] is employed and modified to encode the coefficients.

For a set of possible alphabet **S** = {s_1, s_2, s_3, …, s_n}, the first-order entropy coding algorithms, like Huffman coding and arithmetic coding, allocate bits for each symbol according to their occurrence probability. That is, for a symbol s_i with probability p_i, the bit b_i allocated for this symbol can be written as:

$$b_i = -\log_2 p_i . \tag{5}$$

For example, if symbol s_i has the probability 0.25, then the allocated bit b_i is 2.

The problem of the entropy coding methods is how to transmit the probability distribution of the alphabet to the decoding side. Some "dynamic" coding algorithms, like the adaptive arithmetic coding, provide a smart solution. The probability distribution is automatically reconstructed at decoder by receiving each coded symbol; therefore, the transmitting of probability distribution is omitted.

The adaptive arithmetic coding algorithm [Witte87a] works with a *symbol occurrence pool* and a *window size*. Initially, the symbol occurrence pool contains an occurrence for each symbol. The algorithm calculates the probability of each symbol and used it to encode the new-coming data. This new data is added into the symbol occurrence pool, and the probability of each symbol is re-calculated. This process is repeated, until the total occurrence of the symbols in the symbol occurrence pool reaches the window size. At this time, the occurrence of each symbol is divided by two, and then continues the coding process. In short, if we segment the input data into chunks with a half of window size, before encoding the first data of chunk c_j, the occurrence o_i of each symbol in the pool can be written as:

$$o_i = \frac{1}{2} o_{i,j-1} + \frac{1}{4} o_{i,j-2} + \frac{1}{8} o_{i,j-3} + \ldots , \tag{6}$$

where $o_{i,j}$ is the occurrence of symbol s_i at chunk j. The probability of each symbol after encoding each new data is re-calculated by:

$$p_i = \frac{o_i}{\sum_{k=1}^{n} o_k} . \tag{7}$$

The decoder can reconstruct the symbol occurrence pool by decoding each data, updating the pool, and used it to decode the new data.

In the original design [Witte87a], the initial symbol occurrence pool contains one occurrence for each symbol. Before the occurrence converges to the distribution of input data, there is some bit rate wasted to encode the beginning portion of input data. In this paper, we propose a modification to reduce the wasting bits. A fixed probability distribution is stored in the encoder and decoder. Before encoding, half window size of occurrence is generated according to the fixed probability distribution and then added into the symbol occurrence pool. If the probability distribution of the data to be encoded is similar to the fixed one, the symbol occurrence pool can be converged with fewer steps and then reduce the wasted bit rate. This fixed probability distribution is trained by some typical test images to achieve a universal usability.

4.3. The representation of region boundary

In region-based fractal image coding, we have to record the information of range regions, including locations and shapes (contours) of these regions. One common way to represent the shape of a region is by using the *chain code* scheme.

By inspecting the region structure obtained after the merging process, we can realize that the original image is split into a number of adjacent regions. If we encode each region separately, most of the region boundary will be coded twice, as Ebrahimi mentioned in his paper [Ebrah94a]. The region contour tracing algorithm introduced by Ebrahimi is employed in our implementation.

We also modify the *segmented chain code* [Lu91a] according to the quadtree structure to record the region contour. In the *quadtree-based segmented chain code* (QBSCC in short), the step for each link is no longer unique but depending on the distance between two end points on quadtree segmentation.

The region contour tracing procedure composed with QBSCC is listed here:

Step 1. Index the points on quadtree structure if its degree is great then 2.

Step 2. Erase the outer boundary of the image.

Step 3. Select a not yet encoded boundary from left to right and from top to down. An end point with odd degree has higher priority than the points with even degree. Record the difference of current starting point index to the previous one and the direction of the start link and the run in number of link until the direction changed.

Step 4. Record the direction of the change (0: turn right, 1: turn left) and the run in number of link for this segmented chain, until the next

change of the direction.

Step 5. If encounter the end of the contour, signal it by the termination code (a special segmented code that the run of link is zero). Otherwise, GOTO Step 4.

Step 6. If there are not yet encoded boundaries, GOTO Step 3. Otherwise, STOP.

In this algorithm, there are 3 cases for which the coding bits can be saved. The first case is that the direction of the starting boundary can be ignored (in Step 3) if there are only one selection. The second case is that the direction of the following segmented chain can be ignored (in Step 4) if there are only one selection. The final case occurs when there is no possible following chain to be coded, the termination code can be ignored (in Step 5). Fig. 5 demonstrates an example for tracing and encoding the region boundary, including the three ignore cases. Finally, the difference of starting position and the length of runs are entropy encoded.

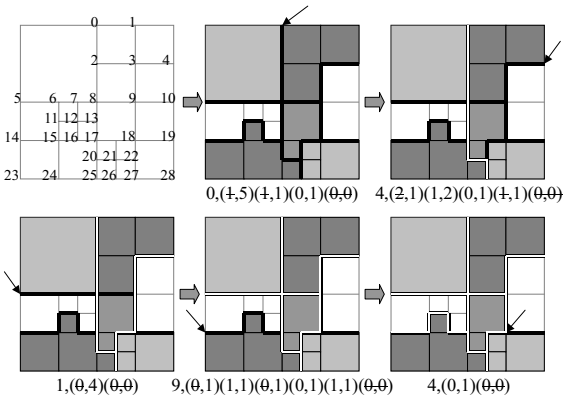


Fig. 5. An example for tracing and encoding of region boundary

5. POST-PROCESSING FOR REGION -BASED FRACTAL IMAGE CODING

Highly visible artifacts become annoying when the coding bit rate is low. If an image is coded by a lossy coding method, the lower the allowed bit rate assigned, the more the artifact become noticeable due to loss information. The coding algorithms also influence the visual quality. The block-based coding algorithms usually produce blockiness artifacts near the block boundary, particularly located in flat areas of an image. The region-based fractal image compression method we proposed produces significant blocking effect at the region boundary with the PSNR lower than 30dB. Therefore, in this section, we propose a filtering solution for the removal of blocking effect.

There are a large amount of studies has been done to improve the reconstructed image quality at low bit rate [Ramam86a]. Although some studies focus on the removal of annoying DCT artifacts at the block boundary, the techniques for reducing the blocking artifacts based on the local content of image can be employed in our region-based fractal image coding [Hu97a] [Shen98a].

5.1. Determine the pixels to be smoothed

By inspecting the merging result of the region-based fractal image compression, we can realize that if there is a large region exists, the image area covered by this region is usually flat. Moreover, some large quadtree blocks instead of small quadtree blocks usually combine a large region. Base on this observation, we expect to eliminate the significant blocking effect at flat area by applying a large smoothing filter over more pixels near the boundary of large quadtree blocks. At the case of small quadtree blocks, applying a small smoothing filter over the few pixels adjacent to boundary would be a better consideration for not over-smoothing those pixels.

Because of the non-regular segmentation, we define a simple method to determine which pixels to be smoothed. Fig. 6 gives a brief description of this method. For each segment of vertical boundary, find the distances from this boundary to the nearest boundaries to the left and to the right. We call this distance as the *horizontal visual space*. For each segment of horizontal boundary, this distance named as the *vertical visual space*. For a larger *visual space*, there would be a larger flat area near this boundary, and therefore more pixels have to be smoothed. Here, for each visual space value, we pre-define an *influence range* to control how many pixels to be smoothed. As demonstrated in Fig. 6 (b), the pixels within the influence range of each segment of boundary have to be filtered.

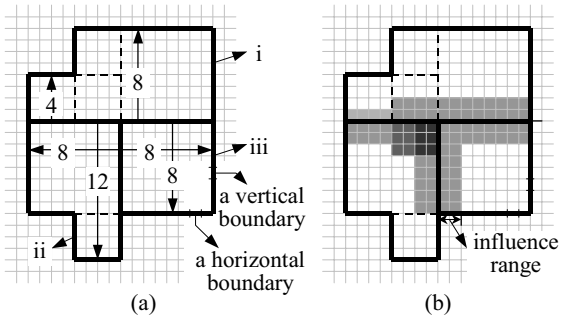


Fig. 6. Determine which pixels to be smoothed

5.2. Smoothing filter with variable size

The next step is to determine what smoothing filter for each visual space. In generally, we should apply a smoothing filter with a larger size over the flat area and a smoothing filter with a smaller size over the rough area. Base on this requirement, we design a filter that can be control by the desired size. The horizontal and vertical elements of the filter are given in Eq. (8), where $2k+1$ or $2l+1$ are the

horizontal and vertical size of the filter, respectively. If a pixel is only within the influence range of one boundary segment, a filter with the same horizontal and vertical size is used, and the corresponding boundary segment determines the filter size. If a pixel is within the influence range of a vertical boundary and a horizontal boundary simultaneously, the vertical visual space determines the vertical filter size and the horizontal visual space determines the horizontal filter size, respectively. The final 2-D filter is the linear combination of the horizontal and vertical elements, as given in Eq. (9). Fig. 7 (b) illustrates an example of the filter size for each pixel to be smoothed.

$$h(n) = 1 - \cos\left(\pi + \frac{2\pi}{2k+1} \cdot n\right), \; n = -k, -k+1, \ldots, k \qquad (8)$$

$$v(n) = 1 - \cos\left(\pi + \frac{2\pi}{2l+1} \cdot n\right), \; n = -l, -l+1, \ldots, l$$

$$f(i,j) = v(i) \cdot h(j), \quad i = -m, -m+1, \ldots, m; \quad j = -p, -p+1, \ldots, p. \qquad (9)$$

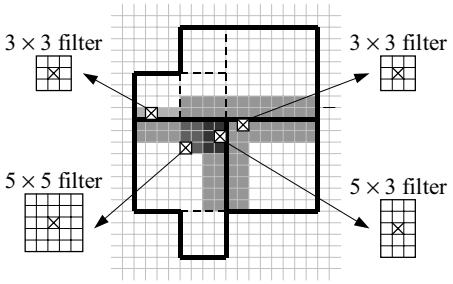


Fig. 7. The adaptive filter size

6. EXPERIMENTAL RESULTS

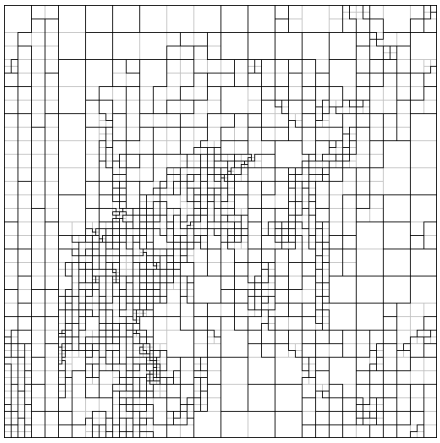
In this section, some experiments for evaluating the proposed algorithm is provided. The test image is 8-bit gray level 512×512 "Lena" image. The maximum and minimum block sizes are 32×32 and 4×4, respectively. The search range parameter L is 16 and the horizontal/vertical step size δ_h and δ_v is 1, thus the bits for encoding the location of best-match domain block is 4 + 4. The contrast factor s_i and brightness factor o_i is uniformly quantized and coded by 4 bits and 6 bits, respectively.

Table 1 lists the simulation results for "Lena" at 27, 29, and 31dB. When the target PSNR is 29dB, the compression ratio achieves 106. The region structure and decoded image for PSNR at 29dB are illustrated in Fig. 8.

Quadtree code	QBSCC	Fractal coefs.	Total bits	Bit rate (bit/pixel)	Decoded PSNR	Smoothed PSNR
1628	3671	6240	11539	0.0440	26.66	27.03
2036	4587	13145	19768	0.0754	28.61	29.00
2864	8710	23191	34765	0.133	30.57	31.02

Table 1. Some simulation results for "Lena"

(a)



(b)

Fig. 8. (a) The decoded image and (b) the region structure with PSNR = 29.00 dB and bit rate = 0.0754 bpp

Base on our poor knowledge, there are no other fractal image compression methods can achieve the low bit rate listed in Table 1. Most researches based on the structure of Jacquin's [Jacqu92a] with only two levels of segmentation, causes them to remain in high bit rate. Only a fairly few methods can work well while bit rate is lower than 0.2 bpp. Fisher and Lawrence [Fishe92a] provide a result of PSNR = 29.2 dB with 0.21 bpp. Chang and Kuo [Chang95a] can achieve 29.2 dB with 0.19 bpp.

Recently, the wavelet transformation with efficient coefficient coding [Shapi93a] [Said96a] achieves an excellent performance. The structure of multi-dimensional wavelet transformation, like the quadtree decomposition, has the same potential to encode image with very low bit rate. In Shapiro's article, the "Lena" image can be encoded with 0.0078125 bpp, although the PSNR is as low as 21.69 dB that no virtual usage. The following researches [Said96a] [Xiong98a] [Khanh97a] further improve the performance of wavelet coding that cause wavelet coding significantly outperforms other image coding methods. A post-processing algorithm [Xiong98a] can reconstruct the blurred edge for low

bit rate wavelet coding. Comparing these rivals with our proposed fractal coding method, it seems that the wavelet coding exhibits slightly better performance than our method. Table 2 summaries some comparison results sorted by descending PSNR.

Author name	Bit rate (bit/pixel)	PSNR (dB)	Bibliography
Xiong *et al.*	0.20	33.34	[Xiong98a]
Fan *et al.*	0.15	32.02	[Xiong98a]
Khanh *et al.*	0.1333	31.20	[Khanh97a]
Ours	0.133	31.02	
Fan *et al.*	0.1	30.45	[Xiong98a]
Fan *et al.*	0.08	29.60	[Xiong98a]
Fisher *et al.*	0.21	29.2	[Fishe92a]
Chang *et al.*	0.19	29.2	[Chang95a]
Ours	0.0754	29.00	
Khanh *et al.*	0.0615	28.22	[Khanh97a]
Ours	0.0440	27.03	

Table 2. The comparisons of different image coding methods at low bit rate

7. CONCLUSION

In this paper, the region-based fractal image compression algorithm is proposed. From the experimental results presented above, we can realize that the proposed algorithm can achieve excellent performance at low bit rate. The adaptive quadtree decomposition provides the flexibility for coding images with a wide range of different bit rate. The region-merging scheme can further eliminate redundant bit rate with a better utilization of image property. The modified adaptive arithmetic coding can efficiently store the fractal coefficients. The post-processing can eliminate the blocking effect while coding images at low bit rate.

Comparing to other fractal image compression methods, the proposed region-based method can outperform other methods in a wide range of different bit rate. However, the excellent wavelet coding methods are still slightly ahead of us. We believe that there are still some improvement space at the selection and encoding of fractal coefficients. A better post-processing algorithm may be another hope.

REFERENCES

[Barns88a] Barnsley,M.: Fractals Everywhere, *Academic Press, San Diego*, 1988.

[Chang95a] Chang,H., Kao,C.: Fractal block coding using simplified finite-state algorithm, *Proceeding of SPIE*, Vol. 2501, pp. 536-544, 1995.

[Ebrah94a] Ebrahimi,T.: A new technique for motion field segmentation and coding for very low bitrate video coding applications, *Proceeding of ICIP 1994*, pp. 433-437.

[Fan98a] Fan,G., Cham,W.-K., Liu,J.: Model-based edge reconstruction for low bit-rate wavelet-based image coding, *Proceeding of ICIP 1998*, pp. 2561-2564.

[Fishe92a] Fisher,Y., Lawrence,A.: Fractal image compression for mass storage applications, *Proceeding of SPIE*, Vol. 1662, pp. 244-255, 1992.

[Fishe94a] Fisher,Y.: Fractal image compression - theory and application, 1994.

[Hu97a] Hu,J., Sinaceur,N., Li,F., Tam,K.-W., Fan,Z.: Removal of blocking and ringing artifacts in transform coded images, *Proceeding of ICASSP 1997*, Vol. 4, pp. 2565-2568.

[Jacks97a] Jackson,D., Mahmoud,W., Stapleton,W., Gaughan,P.: Faster fractal image compression using quadtree recomposition, *Image and Vision Computing*, Vol. 15, pp. 759-767, 1997.

[Jacqu92a] Jacquin,A.: Image coding based on a fractal theory of iterated contractive image transformations, *IEEE Trans. on Image Processing*, Vol. 1, No. 1, pp. 18-30, Jan. 1992.

[Khanh97a] Khanh,N.-P., Hans,W.: DWT image compression using contextual bitplane coding of wavelet coefficients, *Proceeding of ICIP 1997*, pp. 2969-2972.

[Lu91a] Lu,C.-C., Dunham,J.: Highly efficient coding schemes for contour lines based on chain code representations, *IEEE Trans. on Communications*, Vol. 39, No. 10, pp. 1511-1514, Oct. 1991.

[Ramam86a] Ramamurthi,B., Gersho,A.: Nonlinear space-variant post processing of block coded image, *IEEE Trans. on Acoustic, Speech, and Signal Processing*, Vol. ASSP-34, pp. 1258-1268, Oct. 1986.

[Rubin79a] Rubin,F.: Arithmetic stream coding using fixed precision registers, *IEEE Trans. on Information Theory*, Vol. IT-25, No. 6, pp. 672-675, Nov. 1979.

[Said96a] Said,A., Pearlman,W.: A new, fast, and efficient image codec based on set partitioning in hierarchical trees, *IEEE Trans. on Circuits and Systems for Video Tech.*, Vol. 6, No. 3, pp. 243-250, June 1996.

[Shapi93a] Shapiro,J.: Embedded image coding using zerotrees of wavelet coefficients, *IEEE Trans. on Signal Processing*, Vol. 41, No. 12, pp. 3445-3462, Dec. 1993.

[Shen98a] Shen,M.-Y., Kuo,C.-C.: Review of postprocessing techniques for compression artifact removal, *Journal of Visual Communication and Image Representation*, Vol. 9, No. 1, pp. 2-14, 1998.

[Shust94a] Shusterman,E., Feder,M.: Image Compression via Improved Quadtree Decomposition Algorithms, *IEEE Trans. on Image Processing*, Vol. 3, No. 2, Mar. 1994.

[Witte87a] Witten,I., Neal,R., Cleary,J.: Arithmetic coding for data compression, *Comm. of ACM*, Vol. 30, No. 6, pp. 520-540, June 1987.

[Xiong98a] Xiong,Z., Ramchandran,K., Orchard,M.: Wavelet packet image coding using space -frequency quantization, *IEEE Trans. on Image Processing*, Vol. 7, No. 6, pp. 892-898, June 1998.