AN EFFECTIVE FRACTAL IMAGE CODING METHOD WITHOUT SEARCH

Shen Furao † *and* *Osamu Hasegawa* †‡

†Imaging Science and Engineering Lab., Tokyo Institute of Technology (TIT)
‡PRESTO, Japan Science and Technology Agency (JST)

ABSTRACT

Fractal image coding is a computationally expensive method. This paper does some improvement for traditional no search method and Quadtree method, then combines the improved no search method with improved Quadtree method to speed up the fractal encoding process greatly and hold high reconstruction fidelity. Some other time-consuming parts of fractal coding are redesigned and accelerated with new techniques. Experiments on standard images show that the proposed scheme can reduce encoding time greatly with only a little loss of approximation quality.

1. INTRODUCTION

Fractal image coding has been used in many image processing applications such as feature extractions, image signatures, and texture segmentation [1]. 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 [2]. These advantages make fractal image coding a very attractive method for applications in multimedia [3].

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 can not 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 [4].

The drawback of fractal image coding is that the block matching process is very time-consuming. Plenty of research focused on how to improve the speed of fractal image coding [1][5] but cannot satisfy some real time requirement. The no search scheme proposed by Dudbridge [6] is very fast, but it suffers from the very poor reconstruction quality.

In this paper, we shall do some improvement for the no search scheme in [6] and Quadtree scheme in [7], and then combine these two methods to speed up the fractal image coding with high reconstruction quality. Some other time-consuming parts of fractal coding are also redesigned and accelerated with new techniques.

2. FRACTAL IMAGE CODING

2.1. Basic scheme

The basic fractal compression scheme separates an image f into non-overlapping range blocks. For each range block $R = (r_{ij})$, we search the domain pool to find the domain block $D = (d_{ij})$ and affine transform W such that $W(D)$ provides the best matching for R. The domain pool Ω is extracted by sliding a window from the top left corner of image f in horizontal or vertical directions. The transform W consists of a spatial contraction map C followed by gray-level transform G that is the composition of a contrast scaling a and a luminance shift b, viz. $W = G \circ C$, $G(x) = ax + b$. 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 will be constrained in $[-1, 1]$ and both 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. The fractal encoding problem will be searching the domain pool Ω to find $\tilde{D}$, a_k and b_l to make

$$E(R, \tilde{D}) = \min_{D \in \Omega} \min_{i,j} ||a_i D + b_j I - R||^2 \qquad (1)$$

Here, I is a block whose elements are all ones.

The minimization problem defined by (1) 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 and b_j. It depends on how many bits allocated to a and b. (2) Error Calculation: The calculation of

0-7803-8554-3/04/$20.00 ©2004 IEEE.

$||a_iD+b_jI-R||^2$. It depends on the size of range block. (3) Number of range blocks. (4) Domain Search: The search of domain pool. It depends on the number of domain blocks in the domain pool.

2.2. Improve the speed

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.

Different from $G(x) = ax + b$, we adopt the gray-level transform G

$$G(D) = a_i(D - \overline{d}I) + \overline{r}I \quad (2)$$

advocated by [1][5]. Where $\overline{r}$ is range block mean, $\overline{d}$ is domain block mean. With (2), the fractal encoding problem (1) will be

$$E(R, \tilde{D}) = \min_{D\in\Omega} \min_i ||a_i(D - \overline{d}I) - (R - \overline{r}I)||^2 \quad (3)$$

In (3), the IFS parameters becomes a_i and $\overline{r}$. Instead of searching the $a_i, i = 1, 2, \ldots, m$ and $b_j, j = 1, 2, \ldots, n$ in (1), 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.

With the gray-level transform (2), the "Error Calculation" part becomes

$$error = ||a_i(D - \overline{d}I) - (R - \overline{r}I)||^2 \quad (4)$$

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

$$error_c = ||a_i(D_{\frac{1}{2},1} - \overline{d}I) - (R_{\frac{1}{2},1} - \overline{r}I)||^2 \quad (5)$$

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.

There are regions that could be covered well with larger range blocks and regions that are difficult to cover well this way. Quadtree scheme will be improved in Section 3.1 to reduce the number of range blocks.

For Domain Search, we will improve the no search scheme in [6] to speed up the encoding process greatly. The detail will be discussed in Section 3.2.

3. WITHOUT SEARCH SCHEME

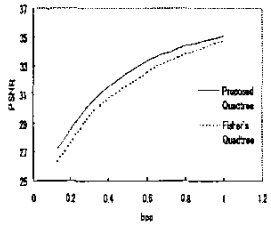
3.1. Quadtree scheme

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 Fisher's quadtree scheme [7] used same constant rms tolerance T for every quadtree level. Here, we propose the adaptive rms tolerance T_n for different quadtree level n. 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 satisfy compression ratio. It means that, for large range blocks, the rms tolerance must be stricter than small range blocks. We compared the constant tolerance and adaptive tolerance scheme with Lena ($512 \times 512 \times 8$) in Fig.1. And the rms tolerance is adapted by

$$T_{n+1} = 2T_n + 1 \quad (6)$$

From Fig.1 we know that this proposed adaptive rms tolerance technique could get better compression results than traditional constant rms tolerance.



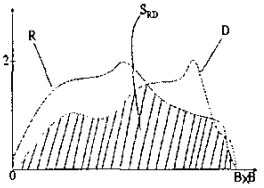


Fig.1: Comparison results **Fig.2**: Similar Degree

3.2. No search scheme

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. Thus, to make (4) minimum, the block $D_{error} = D - \overline{d}I$ and block $R_{error} = R - \overline{r}I$ 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^{norm}_{error} and R^{norm}_{error}. There are $n = B \times B$ pixel intensities, $d_1, \ldots, d_n$ (from D^{norm}_{error}) and $r_1, \ldots, r_n$ (from R^{norm}_{error}). We define the Similar Degree as $S_{RD} = \sum_{i=1}^{n} \min(r_i, d_i)$. 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.

A scheme requiring no search has been described by Dudbridge [6], 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:

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 a threshold t. Record the position difference between such domain blocks 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 step1.

here, B_{min} means the allowance minimum size of range block. We checked Lena ($512 \times 512 \times 8$), Girl ($512 \times 512 \times 8$), and Baboon ($512 \times 512 \times 8$) with the above algorithm. The initial B is set as 16 and B_{min} is set as 2. Suppose the position of range block is (row_R, col_R), the statistic results show that, for most range blocks, the probability of high Similar Degree happening in the position of $(row_R - B/2, col_R - B/2)$ is larger than other positions. It means that if we fix the domain block position in $(row_R - B/2, col_R - B/2)$, we can get better reconstruction fidelity than fix in other positions.

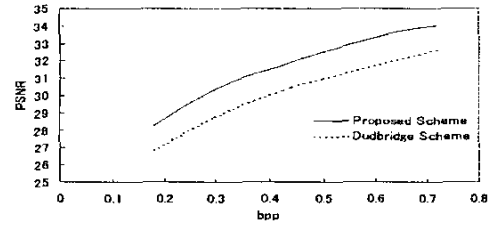


Fig.3: Comparison: Dudbridge with Proposed scheme

We compare this new position scheme with Dudbridge's scheme for Lena ($512 \times 512 \times 8$). Fig.3 gives the comparison results and shows the advantage of new scheme. With the same *bpp*, proposed position scheme can get much higher PSNR than Dudbridge scheme.

3.3. Algorithm

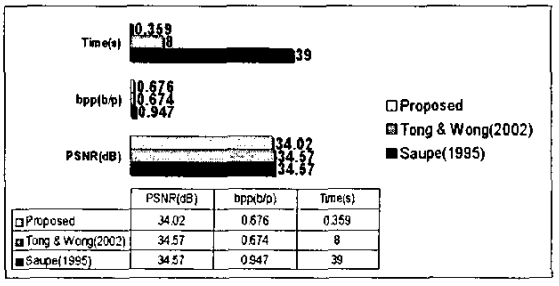
With all the above discussion, we suggest the following no search algorithm to be used for fractal image coding.

1. Give one *rms* tolerance T, separate the original image to range blocks with size B_{ini} and mark all range blocks uncovered.
2. While there are uncovered range blocks R_i, do
   (a) Suppose the position of R_i is (row_R, col_R), size of R_i is $B \times B$, take the domain block in position $(row_R - B/2, col_R - B/2)$ and calculate the $error_c$ defined in (5).
   (b) If $error_c < T$ or $B = B_{min}$, use (4) to search for the best scaling parameter a_i and store a_i and $\bar{r}$, mark R_i as covered.
   (c) If $error_c \geq T$, separate the range block into four equally sized sub-blocks, adapt the *rms* tolerance T with (6), mark these sub-blocks as uncovered range blocks and remove R_i from the list of uncovered range blocks.

Here, B_{ini} means the initial block size of range block. This algorithm combines improved no search scheme with improved Quadtree method, and considers some other accelerating techniques.

4. EXPERIMENT

In the experiment, we compare the algorithm in section 3.3 with a fast method reported by Tong & Wong [1], we use the same test environment of [1]: a PC with an Intel Pentium II 450 MHz CPU and 128 MB memory. 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. $B_{ini} = 16$ and $B_{min} = 2$.



	PSNR(dB)	bpp(b/p)	Time(s)
Proposed	34.02	0.676	0.359
Tong & Wong(2002)	34.57	0.674	8
Saupe(1995)	34.57	0.947	39

Fig.4: Comparison of Tong & Wong [1], Saupe [8] with proposed scheme (with PII 450Mhz PC)

Tong & Wong improved the algorithm designed by Saupe [8] and gave the results for Lena ($512 \times 512 \times 8$) and Baboon ($512 \times 512 \times 8$). The comparison results for Lena are listed in Fig.4. From Fig.4 we find that, with the same *bpp*, our scheme can improve the speed of Tong & 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 0.55dB loss of PSNR.

Table1 is the detail of encoding results of our method. Column 4 lists the encoding time with Pentium II 450MHz

PC. Column 5 is the encoding time with Dell Pentium IV 2.8 GHz PC. It shows that, with the present standard PC, the speed of 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 reconstruction result of proposed method with 0.062 seconds encoding time (PSNR=34.02dB, *bpp*=0.676b/p). Comparing Fig.6 with Fig.5, we know that the proposed method works well.

Table1: Encoding results of proposed method

T	PSNR	bpp	Time(s) PII 450M	Time(s) PIV 2.8G
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 Lena

Fig.6: Decoded image

Table2 is the comparison of Tong & Wong, Saupe and our method for Baboon ($512\times512\times8$). The experiment environment is PC Pentium II 450MHz. With 1.6dB loss of PSNR and 0.37b/p loss of *bpp*, we speed up the Tong & Wong's scheme from 8 seconds to 0.658 seconds. And for Saupe's scheme, we speed up the encoding by 91 times with 1dB loss of PSNR.

Table2: Comparison results of Baboon

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

Fig.7: Original Baboon

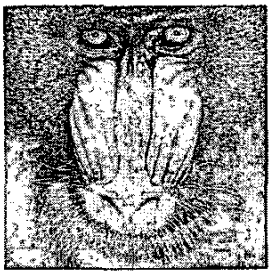


Fig.8: Decoded image

When we do the test on Dell Pentium IV 2.8GHz PC, the encoding of Baboon only spends 0.1 seconds (PSNR = 24.2dB, *bpp* = 1.7b/p). Fig.7 is the original image of Baboon, Fig.8 is the decoding results. From Fig.8 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.

5. CONCLUSION

This paper improved the no search scheme and Quadtree scheme, then combined these two methods to do fractal image coding. Some other techniques were also used to accelerate the speed. Experiments showed that this method could get very fast speed and maintain high reconstruction fidelity. Thus, this research supplied a strong support for the fractal image compression using in real time products.

6. REFERENCES

[1] C.S. Tong, M. Wong, "Adaptive Approximate Nearest Neighbor Search for Fractal Image Compression," IEEE Trans. on Image Processing, vol.11, No.6, pp. 605-614, 2002.

[2] D.M. Monro, P.D. Wakefield, "Zooming with Implicit Fractals," Proceeding of the ICIP-97, Washington DC, October 1997.

[3] M. Barnsley, "Fractal Image Compression," in Image Processing: Mathematical Methods and Applications, Clarendon Press, Oxford, pp. 183-210, 1997.

[4] B. Wohlberg, G.d. Jager, "A Review of the Fractal Image Coding Literature," IEEE Trans. on Image Processing, vol.8, No.12, pp. 1716-1729, 1999.

[5] S. Furao, O. Hasegawa, "A Fast and Less Loss Fractal Image Coding Method Using Simulated Annealing," Proceeding of the 7th JCIS 2003, Cary, North Carolina, USA, Sept. 2003.

[6] F. Dudbridge, "Least Squares Block Coding by Fractal Functions," in Y. Fisher (Eds.), Fractal Image Compression: Theory and Application to Digital Images, New York: Springer-Verlag, pp. 231-244, 1994.

[7] Y. Fisher, "Fractal Image Compression," SIGGRAPH'92 Course Notes, vol.12, pp. 7.1-7.19, 1992.

[8] D. Saupe, "Fractal Image Compression via Nearest Neighbor Search," in Conf. Proc. NATO ASI Fractal Image Encoding and Analysis, Trondheim, Norway, 1995.