1998 International Computer Symposium
Workshop on Image Processing and Character Recognition
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

SPEED UP THE EXHAUSTIVE SEARCH ON FRACTAL IMAGE COMPRESSION

M. H. Hung, C. H. Hsieh, J. H. Jeng, and J. D. Sun

Department of Information Engineering, I-Shou University
Tahsu, Kaohsiung, Taiwan, R.O.C
Email:{m863002m,hsieh}@csa500.isu.edu.tw

ABSTRACT

The encoding process of fractal image compression is extremely time consuming. This paper presents two fast algorithms which aim to reduce the redundant computations of the exhaustive search method. The simulation results indicate that the computations of the two presented algorithms reduces the encoding time over 17%.

1. INTRODUCTION

The most popular fractal compression technique is the so-called block-based compression presented by Jacquin, who used transformations defined for image blocks to encode image data [1]-[4]. In the block-based approach, two operation data sets called called range pool and domain pool are used. The input image is divided into nonoverlapping blocks with a fixed size, say 8 x 8. The collection of these blocks forms the range pool. By partitioning the image into overlapped blocks with another fixed size, say 16 x 16, we obtain the domain pool. Each domain block D is shrunk to 8×8 first, and rotated with degree 90^0, 180^0 and 270^0 together with an additional flip with respect to the vertical center line to constitute 8 orientations. The encoding process is to search the best match domain block from the domain pool for a given range block in the range pool. The encoding with an exhaustive search is extremely time consuming.

A number of fast algorithms such as block classification [2],[3],[4], and DCT-domain block matching [5],[6] have been proposed to the speed up the exhaustive search process. This paper presents two fast algorithms which reduce the redundant computations of the exhaustive search algorithm. They can be combined together and may also be applied into the existing fast algorithms to further reduce the computation. The analysis of the computations for the exhaustive search method is first presented. The proposed fast algorithms are then described. Finally, the simulation results are given.

2. ANALYSIS OF COMPUTATION

In the matching process, the domain block is shrunk to the same block size as the range block. Assume that the block size is N×N, the shrunk domain block is represented as D=($a_{0,0}$, $a_{0,1}$, ..., $a_{N-1,N-1}$), and the range block is R=($b_{0,0}$, $b_{0,1}$, ..., $b_{N-1,N-1}$). For a given R, the search process is to find a D from the domain pool, the contrast factor s and the brightness factor o minimizing (1)

$$\varepsilon = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(s\cdot a_{i,j} + o - b_{i,j})^2 \tag{1}$$

up to the 8 orientations stated above. Taking the partial derivatives of (1) with respective to s and o, and setting zero, one obtains

$$s = \frac{N^2\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}a_{i,j}b_{i,j} - \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}a_{i,j}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}b_{i,j}}{N^2\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}a_{i,j}^2 - (\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}a_{i,j})^2} = \frac{N^2\cdot P_{ab} - S_a\cdot S_b}{N^2\cdot Q_a - S_a^2} \tag{2}$$

$$o = \frac{1}{N^2}\left[\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}b_{i,j} - s\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}a_{i,j}\right] = \frac{1}{N^2}[S_b - s\cdot S_a] \tag{3}$$

An substitution of (2) and (3) into (1), (1) becomes

$$\varepsilon = \frac{1}{N^2}\left[\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} b_{i,j}^2 + s\left(s\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} a_{i,j}^2 - 2\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} a_{i,j}b_{i,j} + 2\cdot o\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} a_{i,j}\right) + o\left(N^2\cdot o - 2\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} b_{i,j}\right)\right]$$

$$= \frac{1}{N^2}\left[Q_b + s(s\cdot Q_a - 2\cdot P_{ab} + 2\cdot o\cdot S_a) + o(N^2\cdot o - 2\cdot S_b)\right] \quad (4)$$

There are 8 computations of (2), (3) and (4) corresponding to the 8 orientations. The amount of computations of each term are listed in Table 1. It is noted that $\Sigma\Sigma b_{i,j}$ and $\Sigma\Sigma b_{i,j}^2$ are neglected since they are calculated only once for each range block. For N=8, the computations required are 704 mul/div + 704 add/sub operations for each range block.

Table 1. Computational complexity for the matching of R and D with 8 orientations

Term	Computations
$S_a=\Sigma\Sigma a_{i,j}$	N^2 add
$Q_a=\Sigma\Sigma a_{i,j}^2$	N^2 mul+ N^2 add
$P_{ab}=\Sigma\Sigma a_{i,j}b_{i,j}$	8×(N^2 mul+ N^2 add)
Eq.(2)	8×(5 mul/div+2 add/sub)
Eq.(3)	8×(2 mul/div+ 1 add/sub)
Eq.(4)	8×(9 mul/div+ 5 add/sub)
Total	(128+9 N^2) mul/div + (64+10 N^2) add/sub

Note:add(addition operator), sub(subtraction operator), mul(multiplication operator), div(division operator)

3. FAST ALGORITHMS

3.1 Algorithm I

This algorithm reduces the redundant computations of $\Sigma\Sigma a_{i,j}$ and $\Sigma\Sigma a_{i,j}^2$. Assume that the block size is 8×8 and the neighboring domain blocks are overlapped with one-pixel apart. Figure 1 shows the relation between the computed parts of current block $B_{k,l}$ and the previous neighbor block in vertical direction, $B_{k-1,l}$, and the pervious neighbor block in horizontal direction, $B_{k,l-1}$.

The sum of entries in the blocks are given as

$$sum(B_{k,l}) = \sum_{i=0}^{7}\sum_{i=0}^{7} a_{i,j},\ sum(R_i) = \sum_{j=0}^{7} a_{i,j},\ sum(C_j) = \sum_{i=0}^{7} a_{i,j}$$

where $sum(R_i)$ and $sum(C_i)$ denote the sum of elements for the i^{th} row and for j^{th} column, respectively. Since $sum(B_{k,l-1})$ and $sum(B_{k-1,l})$ are already computed, the quantity $sum(B_{k,l})$ can be calculated from the previous horizontal neighbor block as

$$sum(B_{k,l}) = sum(B_{k,l-1}) - sum(C_{-1}) + sum(C_7)$$

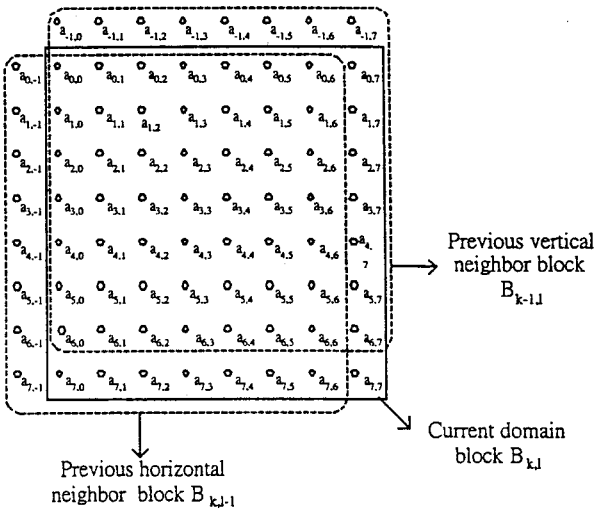

Figure 1. The relation between the current domain block and the neighbor domain blocks

Also, it can be computed from the previous vertical block as

$$sum(B_{k,l}) = sum(B_{k-1,l}) - sum(R_{-1}) + sum(R_7)$$

Similarly, for square sum, one can derive the formulas based on the previously computed quantities as

$$sqr\_sum(B_{k,l}) = sqr\_sum(B_{k,l-1}) - sqr\_sum(C_{-1}) + sqr\_sum(C_7)$$
$$sqr\_sum(B_{k,l}) = sqr\_sum(B_{k-1,l}) - sqr\_sum(R_{-1}) + sqr\_sum(R_7)$$

The new formulas which re-use the previous horizontal neighbor blocks or vertical neighbor blocks need only 16 additions and 16 additions+16 multiplications for the calculations of $\Sigma\Sigma a_{i,j}$ and $\Sigma\Sigma a_{i,j}^2$. The origional formulas required of 64 additions and 64 additions+64 multiplications. Therefore, the reduction of computations for additions and multiplications for each range block are 96 and 48, respecrively. This corresponds to the reduction ratios of 96/704=13.6% for additions and 48/704=6.8% for multiplications.

3.2 Algorithm II

This algorithm reduces the redundant computation of $\Sigma\Sigma a_{i,j}b_{i,j}$. In the comparison of R and D, R is fixed while there are 8 orientations for D. Figure 2 shows the relation of the entries for each orientation using 4×4 block as an example.

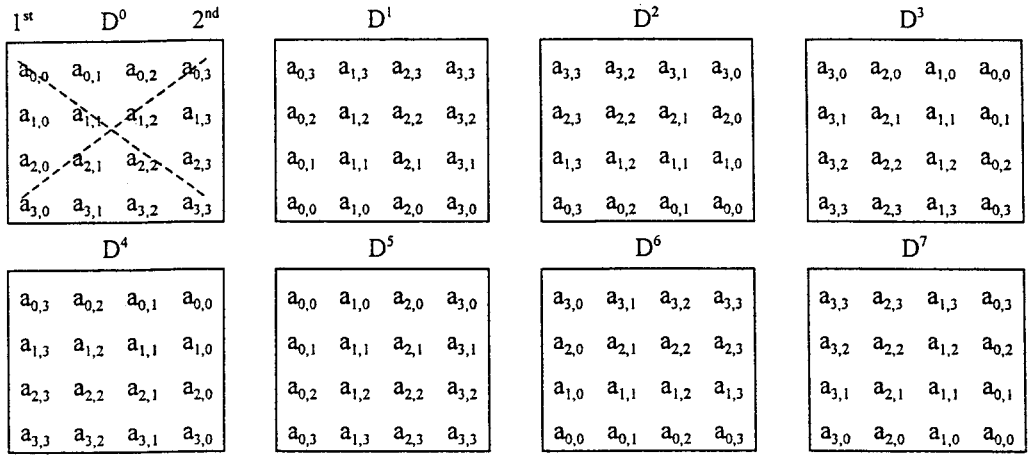

Figure 2. The relation of the entries for each orientation of domain block

In D^0, D^1, D^2 and D^3, the first diagonal entries appear again in D^5, D^4, D^7 and D^6 in the same positions, respectively. Similarly, the second diagonal entries of D^0, D^1, D^2 and D^3 also appear again in D^7, D^6, D^5, and D^4, respectively. Table 2 shows all the repeated portions.

Table 2. Repeated data in the 8 orientations

D^j	diagonal line	appear again in D^j
D^0	$a_{0,0}\, a_{1,1}\, a_{2,2}\, a_{3,3}(1^{st})$	D^5
D^1	$a_{0,3}\, a_{1,2}\, a_{2,1}\, a_{3,0}(1^{st})$	D^4
D^2	$a_{3,3}\, a_{2,2}\, a_{1,1}\, a_{0,0}(1^{st})$	D^7
D^3	$a_{3,0}\, a_{2,1}\, a_{1,2}\, a_{0,3}(1^{st})$	D^6
D^0	$a_{0,3}\, a_{1,2}\, a_{2,1}\, a_{3,0}(2^{nd})$	D^7
D^1	$a_{3,3}\, a_{2,2}\, a_{1,1}\, a_{0,0}(2^{nd})$	D^6
D^2	$a_{3,0}\, a_{2,1}\, a_{1,2}\, a_{0,3}(2^{nd})$	D^5
D^3	$a_{0,0}\, a_{1,1}\, a_{2,2}\, a_{3,3}(2^{nd})$	D^4

It indicated that the partial results of $\Sigma\Sigma a_{i,j}b_{i,j}$ for D^0, D^1, D^2 and D^3 can be used for the computation of $\Sigma\Sigma a_{i,j}b_{i,j}$ for D^4, D^5, D^6 and D^7. For example, in the computation of R and D^0, the term $(a_{0,0}b_{0,0}+a_{1,1}b_{1,1}+a_{2,2}b_{2,2}+a_{3,3}b_{3,3})$ is already computed, but it will be computed again for D_i^5. So if one saves these computed partial results for the computations of R to D^0, D^1, D^2 and D^3, then re-uses the quantities for the computations of R to D^4, D^5, D^6 and D^7, one can thus remove all the redundant computations. For 8×8 blocks, 16×4 additions+16×4 multiplications will be removed, which corresponds to 64/704 = 9.09% reductions for both additions and multiplications. This algorithm can be combined with the Algorithm I to further reduce the computational complexity.

4. SIMULATION RESULTS

The simulation is conducted on the picture "Lena" of size 256×256 and 256 gray levels. The size of image block is 16×16 for elements from the domain pool and 8×8 for that from the range pool. The program is written in Borland C++ of 32-bit window application run on Pentium-133 computer. Table 3 shows the encoding time using the exhaustive search and the proposed fast algorithms. The experimental results approximately match the analysis of the computational complexity in the previous section.

Table 3. Comparison of encoding times for various algorithms

Algorithms	Encoding time	Average search time per range block	Percentage of reduction
Exhaustive Search	5600 sec	5.47 sec	N/A
Fast Algorithm I	4992 sec	4.88 sec	10.86%
Fast Algorithm II	5216 sec	5.09 sec	6.86%
Fast Algorithm I + II	4608 sec	4.50 sec	17.71%

5. CONCLUSIONS

The encoding process for fractal image compression is time consuming due to the exhaustive search for similarities. The two algorithms proposed in this paper remove all the redundant computations in the search process. From the results of the simulations, it is proved that the fast algorithms saves a lot of time in the encoding phase. In addition, Algorithm II may be combined with any existing fast algorithms such as block classification to further speed up the encoding.

6. REFERENCES

[1] Fisher, Y., Fractal Image Compression –Theory and Application, Springer-Verlag, New York, 1994.

[2] Kim I. K. and R. H. Park, "Still image coding based on vector quantization and fractal approximation, " IEEE Trans. Image Processing, 5(4), pp. 587-597, Apr. 1996.

[3] Jacquin, A. E., "Fractal Image Coding: A Review", Proceedings of The IEEE, Vol.81 ,No. 10, pp.1451-1465, Oct 1993.

[4] Jacquin, A. E., "Image coding based on a fractal theory of iterated contractive image transformations", IEEE Trans. Image Processing, 1(1), pp.18-30, Jan. 1992.

[5] Wohlberg B.E. and G. de Jager, " Fast image domain fractal compression by DCT domain blcok matching", Electronic Letters, 31(11), pp.869-870, May 1995.

[6] Zhao Y. and B. Yuan, Image compression using fractals and discrete cosine transform", Electronic Letters, pp. 474-475, Mar 1994