

Fast Fractal Image Compression Based on HV Partition

Guorui Jiang^{*a}, Yuzhuo Zhong^b, Shiqiang Yang^c, Bo Yuan

^aDepartment of Computer Science, Hebei Teacher's University, Shijiazhuang 050016

^{b,c}Department of Computer Science, Tsinghua University, Beijing 10084

ABSTRACT

This paper presents a new scheme to speed up fractal image compression based on HV partition. In this scheme, we propose a new approach based on character track that can speed up the process of searching appropriate domain blocks. In this technique we first abstract the range block's characters, then when we searching in the domain block pool, only domain blocks which characters are exactly fit with the range block could transact the process of matching. The experimental result shows that with this technique we can improve the speed of encoding more one hundred times than the original algorithm (without any accelerate technique).

Keywords: Fractal Image Compression, Partition, Character Track, Encode.

1. INTRODUCTION

With the development of multimedia technology, the three fields that are computer network, telecommunication and broadcasting will connect together, use of video and images is becoming increasingly popular. Because their data is very large, they require large channel bandwidth and storage space. The original image should be compressed for data handling, storage, and transmission. The demand for more and better images has intensified the research in image compression. The fractal image compression is an attractive encoding method which based on IFS(iterated function system)、self-similarity characteristics and the observation that all real-world image are rich in affine redundancy. M. F. Barnsley showed that the compression ratio of fractal image compression can arrive to very high¹⁻², but it needed manual work. A. E. Jacquin introduced first practical fractal block coding approach³⁻⁴, which implemented computer automating encoding, fractal coding had aroused a great deal of interest as a new promising image compression technique. In this technique, the original image is divided into N_r nonoverlapping range blocks and N_d overlapping domain blocks, where dimension of latter is greater. Then, for each range block, the most similar domain block is found using minimum mean square error criterion. However, because the intensive computation of search match of the fractal image coding is extremely high, its encoding time is very long. Therefrom, Many attempts have made to improve the encoding process⁵⁻¹⁸ and decoding process¹⁹⁻²⁰. For example, classification schemes have been developed to alleviate the heavy computation of full search and speed-up encoding, such as classification based on brightness levels, variance, vector quantization and archetype⁵, human visual system⁶, and incorporated with a fuzzy classifier⁷, the wavelet coefficients⁸, clustering process⁹ etc. The main idea of block classification is to categorize the range and domain blocks into different classes according to above features, the searching process is then restricted to the same category.

To enhance compression ratios, several ways to partition image have been introduced, such as quadtree partition, HV (horizontal or vertical) partition, Edge-based partition, Hexagonal partition⁵, triangular¹⁰⁻¹¹, diamond partition¹² and quadtree

* Correspondence: Email: guoruij@sj-user.he.cninfo.net; <http://www.hebtu.edu.cn> TEL: (86)0311 5076921.

recomposition¹³⁻¹⁵ etc. The most commonly used method is quadtree partition, for this kind of partition is easy to be carried out and the classification (general four levels according size) of domain blocks can be repeated use. But the experimental results show that HV partition is obviously better than quadtree partition, when the image's texture is smooth and the structure of image is not very complex. However, the size of range blocks created by HV partition is the more variable than quadtree partition, therefore we could hardly use the strategy of pre-classify and other pretreatment. This problem has restrained the practical application of HV partition for a long time.

This article present a fast algorithm of fractal image compression based on HV partition. In this approach, we use a method of character track while searching for the appropriate domain block. The three character standards we abstract are respectively based on the distributing of brightness, the gray difference and the gray ratio. The experimental results show that with this technique we may improve the encoding speed more than one hundred times than the original HV algorithm (without any acceleration).

The arrangement of this paper is as follows. In Sec. 2, we briefly describe the theory and application of fractal image compression. In Sec. 3, we introduce how HV partition work. Then, in Sec. 4, we present the algorithm based on character track. Next, in Sec. 5, we show the method of fast implement of character track. Finally, in Sec. 6, we give experimental results and conclude the paper.

2. THE THEORY AND APPLICATION OF FRACTAL IMAGE COMPRESSION

In this section, we briefly review the theory of fractal image compression and introduce automatic fractal encoding approach.

2.1. The Theory of Fractal Image Compression

If (\mathbf{R}^n, d) is a Euclidean space, $X=\{I \mid I \text{ is a close set (digital image) in } \mathbf{R}^n \}$, d is a metric on \mathbf{R}^n , then we can get a derived metric h on X as follows. For arbitrary $A, B \in \mathbf{R}^n$, let

$$h(A, B) = \max \left\{ \sup_{x \in A} \inf_{y \in B} d(x, y), \sup_{y \in B} \inf_{x \in A} d(y, x) \right\}, \quad (1)$$

where *max* denote *maximum*, *sup* *supremum* and *inf* is *infimum*, then h is said to the *Hausdorff metric*. So the set X of digital images is endowed with a topological structure and become an image space (X, h) . It's a complete metric space.

Mathematical bases of fractal image compression are mainly Iterated function systems (IFS) theorem, Contraction mapping theorem and Collage theorem²¹.

An *iterated function system* consists of a complete metric space (\mathbf{R}^n, d) together with a finite set of contraction mappings $w_n: \mathbf{R}^n \rightarrow \mathbf{R}^n$. In complete metric space (X, h) , for arbitrary a contraction mapping $w: X \rightarrow X$, w has a unique fixed point (attractor), which is an image in the image space. Union of the contraction mappings in an iterated function system is just a contraction mapping in the image space. Therefore an iterated function system can determine an image. This provides mathematical base of decoding for fractal image compression. For a fixed image I , how can we find an IFS, whose attractor is approach to I ? The collage theorem gives a wonderful answer as follows:

If (X, h) is an image space, let $I \in X$, $\varepsilon \geq 0$ be given. Choose an IFS $\{ \mathbf{R}^n: w_1, w_2, \dots, w_n \}$ with contractivity factor $0 \leq s < 1$, such that

$$h \left(I, \bigcup_{i=1}^N w_i(I) \right) \leq \varepsilon, \quad (2)$$

Then

$$h(I, A) \leq \frac{\varepsilon}{1-s}, \quad (3)$$

where A is attractor of the IFS. This result shows that if union of images $w_i(I)$ ($i=1,2,\dots,N$) of contraction mappings approach image I , then attractor A of IFS correspondingly approach to I . This gives theoretical base of fractal block coding.

Fractal block encoding for a given image is chiefly to find one IFS whose attractor approaches original image and record their correlative parameters. Fractal image decoding is mainly to reconstruct IFS by the parameter and generate attractor of the IFS.

2.2. Conventional Fractal Image Encoding Approach

In fractal image coding, the most popular techniques are as follows:

2.2.1. Image partition.

For given image I , we first partition the image I by some approach^{5, 10-15} into many range blocks R_1, R_2, \dots, R_N , such that

$$I = \bigcup_{i=1}^N R_i. \quad \text{when } i \neq j \quad R_i \cap R_j = \phi. \quad (4)$$

We also partition the image I into some domain blocks, denoted by D_1, D_2, \dots, D_M which should be larger than the range block in order to fulfill the contraction condition. The set $\{D_1, D_2, \dots, D_M\}$ is said to *domain block pool*.

2.2.2. Metric on images

For an image space, we have known that there is a hausdorff metric. Of course, there are many metrics that measure the distance between two images. However in practical fractal image encoding, we should choose the simplest and the easiest computational a metric. For example⁴⁻⁵,

If X is a set of images, $f(x,y), g(x,y) \in X, (x,y) \in I^2$, then

$$d_{rms}(f, g) = \sqrt{\sum_{(x,y) \in I^2} [f(x,y) - g(x,y)]^2} \quad (5)$$

is a *root mean square (rms)* metric. And

$$d_{sup}(f, g) = \sup_{(x,y) \in I^2} |f(x,y) - g(x,y)| \quad (6)$$

is a *supremum* metric.

2.2.3. Search and match

During the encoding, for each R_i , we should seek from domain pool a suitable D_j that is the best-match block under some contraction mapping, namely, $d(w_i(D_j), R_i)$ is minimum or less than some given threshold. These self-similarity searches are very computationally intensive as compared to other algorithm, such as wavelet algorithm, JPEG algorithm etc. Now, classification schemes⁵⁻⁹ and fast search schemes^{16-18, 22} have accelerated the search-match process effectively, and decreased encoding time greatly.

2.2.4. Contraction mapping

In search-match process, we use contraction mappings w_i ($i=1,2,\dots,N$) which are three-dimensional transformations:

$$W_i \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} w_{1i} & w_{2i} & 0 \\ w_{3i} & w_{4i} & 0 \\ 0 & 0 & s_i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix} \quad i=1,2,\dots,N \quad (7)$$

where s_i , o_i denote the contrast factor and intensity shift respectively. The 3D transformations include a 2D spatial transformation and a 1D grey level transformation. In practice, the 2D spatial mappings are usually contracting ($0, \pi/2, \pi, 3\pi/2$) rotations and reflections (in x -axis, y axis, line $y=\pm x$), which determine the shape and position of attractor of IFS. The computation of optimal values for s_i and o_i is as follows:

For given two image blocks containing n pixel intensities, a_1, a_2, \dots, a_n (sampling pixel intensities or average pixel intensities from D_i), b_1, b_2, \dots, b_n (from R_i),

$$\text{if } n \sum_{j=1}^n a_j^2 - \left(\sum_{j=1}^n a_j \right)^2 = 0, \quad \text{then } s_i = 0, \quad o_i = \frac{1}{n} \sum_{j=1}^n b_j$$

$$\text{if } n \sum_{j=1}^n a_j^2 - \left(\sum_{j=1}^n a_j \right)^2 \neq 0, \quad \text{then}$$

$$s_i = \frac{\left[n \sum_{k=1}^n a_k b_k - \sum_{k=1}^n a_k \sum_{k=1}^n b_k \right]}{\left[n \sum_{k=1}^n a_k^2 - \left(\sum_{k=1}^n a_k \right)^2 \right]} \quad (8)$$

$$o_i = \frac{1}{n} \left(\sum_{k=1}^n b_k - s_i \sum_{k=1}^n a_k \right)$$

2.2.5. Distortion measure

Distortion of image in fractal image encoding is measured by the *Peak signal-to-noise ratio* (PSNR). It is defined as⁵

$$PSNR = 20 \log_{10} \left(\frac{l}{d} \right) \quad (9)$$

where l is the largest possible value of the signal (typically 255 for grey image), and d is the *rms* difference between two images, namely,

$$d = rms = \sqrt{\sum_{k=1}^n (a_k - b_k)^2} \quad (10)$$

where a_1, a_2, \dots, a_n (from $w_i(D_i)$) and b_1, b_2, \dots, b_n (from R_i) are pixel intensities.

3. THE HV PARTITION APPROACH

In quadtree partition, during the search-match, range block R_i , without to find appropriate domain block to match, is always to be partitioned into four same sub-blocks, when the size of range block R_i could not reach the minimum set in advance. However, observing self-similarity characteristics, many of these range blocks needn't always to be partitioned into four same sub-blocks, sometimes they need only to be partitioned two sub-blocks, which can match with appropriate domain blocks. Obviously, in this case the quadtree algorithm may produce some redundancy of range blocks. Actually, HV partition may conquer this disadvantage, because we only partition range block by HV partition into two parts horizontally or vertically based on the image texture. For each block R_i that we want to partition to two sub-blocks, how can we choose partition line? We should consider three factors at the same time: 1) partition line parallel to the short side of block R_i ; 2) partition line is close to the center of R_i ; 3) partition line is drawn along some tangent of boundary. Y. Fisher⁵ has given a algorithm of HV partition as follows:

If r_{ij} for $0 \leq i < N$ and $0 \leq j < M$ denote the pixel values in the block R_i , where M, N denotes the height, width of the block R_i respectively. Then $\sum_k r_{k,l}$, $\sum_l r_{k,l}$ is the summation of pixel values in l th line, k th column respectively. Set

$$h_l = \frac{\min(l, M-l-1)}{M-1} \left(\sum_k r_{k,l} - \sum_k r_{k,l+1} \right)$$

$$v_k = \frac{\min(k, N-k-1)}{N-1} \left(\sum_l r_{k,l} - \sum_l r_{k+1,l} \right)$$
(11)

and let

$$H(L) = \max\{h_l / l : l=1, 2, \dots, M\}$$

$$V(K) = \max\{v_k / k : k=1, 2, \dots, N\}$$

If $H(L) > V(K)$, then we make the partition at the $No. L$ line;

If $H(L) < V(K)$, then we make the partition at the $No. K$ column;

If $H(L) = V(K)$, then we make the partition parallel the narrow side of the block R_i .

For the two sub-blocks of block R_i , if we could find appropriate domain blocks to match with them, then the partition of the block R_i is finished. We could record the encoding data of two sub-blocks only instead of four sub-blocks, so HV partition enhances the encoding efficiency. However about classification scheme, it is easy to implement, and rate of repeating use of domain block classes is very high in quadtree partition. But HV partition has not this advantage. How can we speed up the fractal block coding in the HV partition in order to use the higher encoding efficiency? Now we pose a new idea about it.

4. THE ALGORITHM BASED ON CHARACTER TRACK

Because sizes of range blocks R_i are varieties in the HV partition, if we categorize the domain blocks as in quadtree partition, then we would meet great computation. It isn't practical. So we propose a new scheme which be called "character track".

4.1. The Basic Idea of Character Track

During the search-match in the HV partition, for each range block R_i , first some characters of the range block R_i are abstract by some kinds of standards we set in advance. Then the search is done through the domain block pool, step by step, according to the characters. Finally only those domain blocks whose characters are fit with the range block's characters exactly could be matched with the range block R_i .

To minimize the complexity of the character abstracting and avoid the unnecessary search, there are three aspects as follows that must be considered.

- 1) Characteristic standards are divided into several levels, its computational intension is from low to high. For example, first characteristic level must be the simplest to compute; and second characteristic level is simpler to compute than third; third characteristic level is simple to compute etc. And computation of next level uses enough the front computation results.
- 2) So-called "character track" mean that using the characteristic standards tracks domain blocks with same characters from domain block pool. It isn't to categorize domain block pool.
- 3) The search process is just the track process. Set

$$T_i = \{D_l \mid D_l (l \in A) \text{ are domain blocks with } i\text{th level, } A \text{ is an index set}\}, \quad i=1, 2, \dots, N.$$

For each range block R_k , first, using 1 th level excludes a great deal domain blocks in domain block pool from match and gets set T_1 . Next, using 2 th level finds set T_2 from the set T_1 . Thirdly, using 3 th level gains set T_3 from the set T_2 , ... Lastly,

using N -1th level obtains set T_N from the set T_{N-1} . Then the range block R_k matches only with $D_l \in T_N$.

Using the scheme of characteristic track will decrease greatly computational intension during the search-match. Therefore it speed up the fractal encoding based on HV partition. Next we give its a concrete and simple example.

4.2. An Algorithm of Characteristic Track

For characteristic standards, there are many ways to choose. Here, we propose three kinds of character criterions with different complexities (from low to high) as follows.

4.2.1. Three characteristic levels

First characteristic level:

We suppose that the image block R_i is made up of four parts (Left-Up, Right-Up, Left-Bottom, Right-Bottom), then calculate the summation of pixels' value as *brightness*. Let B_1, B_2, B_3, B_4 denotes the gray value summation of Left-Up, Right-Up, Left-Bottom, Right-Bottom respectively. And we could always regard the Left-Up part as the brightest part through transform. Then we regard the sequence of B_i ($i=1,2,3,4$) as the first character.

Second characteristic level:

Suppose $B_1 > B_2 > B_3 > B_4$, where B_i is the gray value summation in one of the four parts. Let $V_1 = B_1 - B_2, V_2 = B_2 - B_3, V_3 = B_3 - B_4$, then regard the sequence of V_1, V_2, V_3 as the second character.

Third characteristic level:

Suppose $B_1 > B_2 > B_3 > B_4$. Let $\lambda = B_1 / B_4, \mu = B_2 / B_3$. If either B_3 or B_4 is equal to zero, then the value of (λ, μ) is not exist, we can transact it separately. If the values of (λ, μ) exist, then they form the third character.

4.2.2. The process of character track

For each range block R_i to be matched, first its three characteristic levels are computed, next domain blocks with same characters are tracked, level by level, according to the three levels, then the process of matching are carried out.

In those three standards, the fore two levels are easy to be calculated. Using them to track can greatly reduce the scope of domain blocks to match. The third standard is based on the character of self-similarity scale. It's different from the first and second standard in the way of application. We first get rid of the mean-value of the sub-block, then calculate (λ_R, μ_R) as the character of range block $R_i, (\lambda_D, \mu_D)$ as the character of domain block D_l .

Given $\delta, \varepsilon > 0$. Let $\lambda_R = t \lambda_D, \mu_R = s \mu_D$. If $s, t \in [1 - \delta, 1 + \varepsilon]$, then we may regard the character of range block and domain block as same in this level. We could adjust the value of δ, ε to fulfil the application's request. $[1 - \delta, 1 + \varepsilon]$ may be explained as a kind of glide window.

5. THE METHOD OF FAST IMPLEMNET OF

CHARACTERISTIC TRACK

In practical application of the HV partition, to speed up encoding, we use the algorithm of characteristic track with the following techniques.

From the statistic data, we could find that the nearing space around a range block has more probability to contain the appropriate domain block. So we adopt a kind of screwy extend search way, which center is the range block R_i (see Figure 1, solid line rectangle denotes R_i , another D_i ,

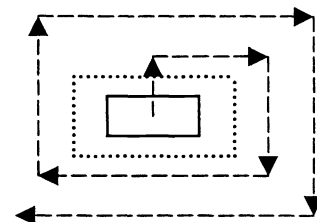


Fig. 1 Search way

arrowhead direction is the search way). In this way, threshold (error) E that is usually variable dependent on the area of R_i must be appointed. During search, once match error arrives to less than the threshold, the corresponding D_i is record, and the search for the R_i finished. The step size of search, that is a constant value or a variable value from small to big gradually, is set in advance. We use four kinds of spatial transformations: rotations of $0, \pi$ and reflections in X-axis, Y-axis.

6. EXPERIMENTAL RESULTS AND CONCLUSIONS

Using the algorithm of the three standards above, we have done some experiments, representative results is following.

6.1. Experimental Result

Program running environment is PC computer: Cyrix 133MHz, 32M memory. For the convenience of compare, all the data of code are stored in its original form, without doing any kind of noiseless coding.

6.1.1. An experiment using HV partition and quadtree partition

We take a frame (Fig. 2(a)) from VCD film. Its texture is moderate smooth and its structure isn't very complex. For this image, we use both methods, HV partition (propose algorithm) and quadtree partition⁵, to compress it respectively. Compression results are following.

Table 1 Compression ratio, PSNR, range blocks for HV and quadtree partition

NO.	Partition Algorithm	Compression Ratio	PSNR	Range Blocks
Fig. 2(b)	Quadtree	22: 1	36.8	3139
Fig. 2(c)	HV	30: 1	36.0	1883

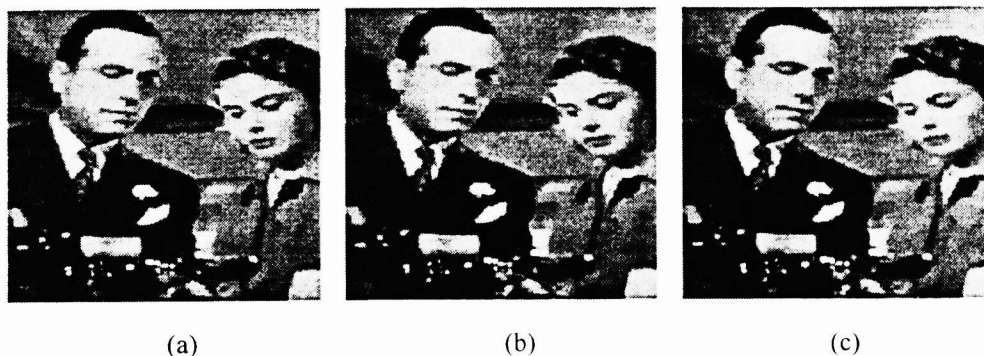


Fig. 2 (a) Original image, (b) Quadtree compression, (c) HV compression.

The experimental results show that the efficiency of HV partition is obviously higher than quadtree partition for texture smooth and structure simple image.

6.1.2. The compression time comparison

Compression time of image is dependent on computer, program and parameter. For an image, using same computer and same program, its compression times are different for different parameters. Several experiments show that encoding an image spends a few minutes in the proposed algorithm, almost ten hours in sufficient search. Of course, the proposed algorithm encoding reduces PSNR from 1 to 2 dB and exchange more one hundred times speed-up than sufficient search.

Why is the coding time difference so large? We explain it by statistic data from coding a 512×512 lena image. In our proposed algorithm, the image is partitioned to 1639 blocks, the step size of search is 4, and match times is 310000. The average area of a sub-block is 160. The average match times of each sub-block are 189. For a sub-block whose area is 160,

suppose it is a square, by step size 4 and four transformations with shrinking factor 2, its average match times for sufficient search is about 59200, which is more than 300 times than 189. However the characteristic abstract is computation simple, it spends little time. This is the reason that the character track encoding enhances speed more one hundred times than the sufficient search algorithm.

6.1.3. Another experiment in the HV partition

By character track algorithm based on HV partition, we record following data for the 512×512 Lenna gray image under two different threshold parameters.

Table 1 Two results under different parameter for 512×512 Lenna

No.	CR	PSNR	Range Blocks	Time(sec)
Fig. 1	24.5: 1	28.6	2317	750
Fig. 2	40.3: 1	27.0	1406	520



Fig. 3 The 512×512 Lenna (here 2:5 display), (a) Original image, (b), (c) compression with CR=24.5, 40.3.

6.2. Conclusion

In the article, we propose the new idea about character track based on the HV partition, and implement an its three character standards algorithm. The experimental results show that using the proposed algorithm may improve the encoding speed more one hundred times, and bear lost PSNR from 1 to 2 dB cost than the original algorithm. Meanwhile we also see that the advantage of HV partition in partition efficiency is evident. Although our given algorithm enhances the encoding speed in the HV partition greatly, its encoding speed is little lower than quadtree algorithm with classification scheme.

REFERENCES

1. M. F. Barnsley, *Fractals Everywhere*, Academic Press, New York, 1988.
2. M. F. Barnsley and A. D. Sloan, "A better way to compress images," *Byte*, 1, pp. 215-223, 1988.
3. A. E. Jacquin, "Image coding based on a fractal theory of iterated contractive image transformations," *IEEE Tran. Image Processing*, 1, pp. 18-30, 1992.
4. A. E. Jacquin, "Fractal image coding: a review," *Proceedings of the IEEE*, 81(10), pp. 1451-1465, 1993.
5. Y. Fisher, *Fractal Image Compression: Theory and Application*, Springer-Verlag, New York, 1995.
6. Sunil Kumar and R.C.Jain, "Low complexity fractal-based image compression technique," *IEEE Transactions on Consumer Electronics*, 43(4), pp. 987-993, 1997.
7. K.F. Loe.; W. G. Gu.; K.H.Phua "Speed-up fractal image compression with a fuzzy classifier," *Signal Processing: Image Communication*, 4(10), pp. 303-311, 1997.

8. Y. Zhang and L. M. Po, "Speeding up fractal image encoding by wavelet-based block classification," *Electronics Letters*, 32(7), pp. 2140-2141, 1996.
9. C. J. Wein, I. F. Blake, "On the Performance of fractal compression with clustering," *IEEE Transaction on Image Processing*, 53, pp. 522-526, 1996.
10. F. Davoine, E. Bertin, A. Chassery, "An adaptive partition for fractal image coding," *Fractals*, 15, pp. 243-256, 1997
11. D. Franck, etc., "Fractal image compression based on Delaunay triangulation and vector quantization," *IEEE Transactions on Image Processing*, 2, pp. 338-346, 1996.
12. Huaqiu Deng; Wenhua Weng; Li Li; Yinglin Yu, "An efficient fractal image compression method," *IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, 5, 4204-4206, 1997
13. D. J. Jackson, W. Mahmoud, W. A. Stapleton, P. T. Gaughan, "Faster fractal image compression using quadtree recomposition," *Image Vision Computing*, (15), pp. 759-767, 1997.
14. D. J. Jackson, W. Mahmoud, "Parallel pipelined fractal image compression using quadtree recomposition," *The Computer Journal*, 139(1), pp. 1-13, 1996.
15. Yigang Wang, Yiweng Jin, Qunsheng Peng, "Merged quadtree fractal image compression," *Optical Engineering*, 1.37(37), pp. 2284-2289, 1998.
16. L. Cieplinski, C. Jedrzejek, T. Major, "Acceleration of fractal image compression by fast nearest-neighbor search," *Fractals*, 5, pp. 231-241, 1997.
17. C. K. Lee, W. K. Lee, "Fast fractal image block coding based on local variances," *IEEE Transactions on Image Processing*, 7 (6), pp. 888-891, 1998.
18. M. Ramkumar, G. V. Anand, "An FFT-based technique for fast fractal image compression," *Signal Processing*, 63, pp. 263-268, 1997.
19. Yong Ho Moon, Yoon Soo Kim, Joe Ho Kim, "Fast fractal decoding algorithm with convergence criteria," *Optical Engineering*, 136(7), pp. 1992-1999, 1997.
20. H. Lin, A. N. Venetsanopoulos, "Fast fractal image compression using pyramids," *Optical Engineering*, 37(6), pp. 1720-1731, 1998.
21. M. F. Barnsley and L. P. Hurd, *Fractal Image Compression*, AK Peters, Ltd., Wellesley, Ma., December 1992.
22. D. Saupe, S. Jacob, "Variance-based quadtrees in fractal image compression," *Electronics Letters*, 33(1), pp. 46-48, 1997.