**Submitted to the Workshop on Multimedia Technologies**

# Fast Fractal Image Encoding

Yi-Chun Wang and Shu-Yuan Chen [1]
Department of Computer Engineering and Science
Yuan-Ze University
135 Yuan-Tung Rd., Nei-Li, Chung-Li
Taoyuan, Taiwang, 320, R. O. C.

**Abstract** This thesis focuses on fast fractal image encoding. Fractal image encoding has a lot of advantages such as high speed decoding, competitive distortion-compression rate, and resolution independent decoding. However, it suffers from long search time to find a close match between a range block and a large pool of domain blocks. Furthermore, each pair of range-domain match involves time-consuming mean square error (MSE) calculation, since each calculation needs a lot of time to get contrast scaling by a complicated formula. In this study we propose two new techniques to reduce the number of MSE calculations and the time to accomplish an MSE calculation, which, in turn, speed up the encoding time.

First, MSE criterion is adopted to prune similar domain blocks in advance so that consequent searching time during encoding can be reduced dramatically. Note that when the size of the image increases, the size of domain pool increases exponentially. However, the proposed pruning method only increases the size of domain pool in nearly linear order. Second, a simple equation to approximate contrast scaling is derived on the basis of regression analysis so that the time taken to calculate each MSE can be dropped. Moreover, the estimated contrast scaling can be used to modify the adaptive search rule [1] so that the adaptive search rule can be performed more efficiently.

In a summary, the proposed method achieves good speed-up while maintaining the quality of compression. Various experimental results show that our method yields superior performance over other existing fractal encoding methods.

**Keywords:** fractal image encoding, fractal codes, image compression, domain pool reduction, adaptive search, classification.

[1]To whom all correspondence should be sent.
Tel: 886-3-463-8800 Ext. 357
Fax: 886-3-463-8850
Email: cschen@cs.yzu.edu.tw

# 1 Introduction

In this section, we describe the motivation of this study and introduce the method we proposed in this study. We also present state-of-the-art methods in fractal image encoding.

## 1.1 Motivation

Fractal image encoding have advantages like fast decoding, competitive distortion-compression rate, and resolution independent decoding. Though the long encoding time has been a major drawback to prevent it from practical usage. Therefore, a lot of papers about fractal image encoding focused on speeding up the encoding process.

In recent years, many encoding strategies were proposed to reduce the encoding time from hours to minutes. This is a great improvement, of course, but compared to state-of-the-art image encoding method like JPEG, which can encode an image in a few seconds, the gap is still considerably large.

Thus, the goal of this study is to propose new techniques to get fast encoding method while maintaining the PSNR value to an acceptable level.

## 1.2 Survey on Fractal Image Compression

Barnsley was the first one to propose the notion of *Fractal Image Compression* [2, 3, 4]. But Ref. [5] was the first paper to formalize a rigid procedure of encoding digital images based on fractal theory. In this paper, Jacquin assumed that "image redundancy can be efficiently exploited through self-transformability on a blockwise basis." Besides the baseline encoding procedure, to which the proposed method is compared, originates from this paper with slight changes in isometry transformation (rotation) and the partition scheme.

After Jaquin, fractal image encoding has gotten much attention among reseachers. In Ref. [6], different partition schemes are described to condense the encoding results as well as to improve the image quality, for example quadtree partition and HV partition.

However, long encoding time is a majore barrier of adopting fractal image compression as a practical usage. Thus, a lot of papers intended to speed up encoding process. In Ref. [7], they classify image blocks in frequency domain and achieve a decent result. They can encode the $256 \times 256$ Lenna image in less than four minites, yet with PSNR value oever 28. Another technique to classify image blocks in spatial domain [8] also has the same speedup rate and with higher PSNR value (over 30).

Nappi et. al. [9] classify the speed-up techniques into two categories: classification techniques and feature vector techniques. They also conclude

that state-of-the-art techniques can encode $512 \times 512$ gray level Lenna image in less then 60 seconds and achieve a PSNR value over 32.

The most effecient method seems to be the one proposed in Ref. [1] and it is the method we mentioned in the abstract. It is a hybrid approach by combining different effective techniques. The method can encode the $256 \times 256$ gray level Lenna image in less than 10 seconds and achive a PSNR value over 30.

### 1.3 Proposed Approach

In this study we propose two techniques to reduce the number of MSE calculations and the time to accomplish an MSE calculation, which, in turn, speed up the encoding time.

First, MSE criterion is adopted to prune similar domain blocks in advance so that consequent searching time during encoding can be reduced dramatically. Note that when the size of the image increases, the size of domain pool increases exponentially. However, the proposed pruning method only increases the size of domain pool in nearly linear order. Second, a simple equation to approximate contrast scaling is derived so that the time taken to calculate each MSE can be dropped. Moreover, the estimated contrast scaling can be used to modify the adaptive search rule [1] so that the adaptive search rule can be performed more efficiently.

### 1.4 Organization of This Work

In Section 2, we describe the baseline fractal image encoding and decoding procedures. In Section 3, the proposed approach is described. Experimental results are included in Section 4. Section 5 gives conclusions and summarizes possible directions to further speed up the encoding time.

## 2 The Baseline Factal Image Encoding

In this section the basic concept of fractal image encoding is introduced. We also describe the baseline encoding and decoding procedures which do not use any speed up technique.

### 2.1 Introduction to Fractal Image Encoding

Fractal is a branch of mathematics, trying to extends traditional mathematical geometry to describe the geometry of nature. One important property of fractal is that you can produce visually complexed data by applying a simple rule over and over again. More detailed information about fractal can be referred to Ref. [10].

In fractal image encoding, we are dealing with the inverse problem. That is, we are try to find the simple rules that produce the visually complexed image. An important way to do this is described as follow: the target image is divided into blocks, referred to as range block, and for each range block, we find a larger block in the image, referred to as domain block, that looks like the range block by some predefined distance metric.

The domain block can be slightly adjusted (through geometic and intensity transformations) to match the range block. The coefficients of the transformation are the encoding results (fractal codes). On the other hand, to decode an image, those fractal codes are applied to any initial image and transform domains into their corresponding ranges. After a few iterations, the image would converge to an image that looks like the original one.

The above encoding process is first formulized by Jacquin [5] in 1992, and is disscussed throughly by Fisher and others [6] in 1994. The encoding procedure is outlined below with $R$ and $D$ as range and domain blocks, respectively.

**Procedure 1. The baseline fractal image encoding.**
Partition the image into range blocks.
Define domain pools, including size, shift amount.
For each range: $R$,
  Out of all possible domain blocks: $D$,
    For each possible rotation of $D$,
      Shrink $D$ to the same size of $R$.
      Find the best $\omega_i$ for $R$ that minimize the $MSE$ distance.
      $\omega_i$ is the fractal code.

Domain block's side length is often twice the range's. Domain shift is defined as the distance between two adjacent domains, which determines how many domains we have and impacts the encoding time and quality of the decoded image. The rotations of the domain blocks here follows definition given in Ref. [6] and includes eight rotations, 90 degree for four, then flip over for another four.

The $\omega_i$ in Procedure 1 is an affine transformation as shown in Equation 1, includes geometry translation, scaling and luminance scaling, offset.

$$\begin{bmatrix} x'_r \\ y'_r \\ z'_r \end{bmatrix} = \omega_i \begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ z_d \end{bmatrix} + \begin{bmatrix} e_i \\ f_i \\ o_i \end{bmatrix} \tag{1}$$

here $x_d$, $y_d$ and $z_d$ are coordinates and luminance of a pixel in domain block, $x'_r$, $y'_r$ and $z'_r$ are of the approximated ranges', $a_i$, $b_i$, $c_i$, $d_i$ are coefficients of geometry scaling transformation, $e_i$, $f_i$ are for geometry translation, and $s_i$, $o_i$ are contrast scaling and intensity offset coefficients, respectively.

In a summary, the baseline encoding process for a range block is to choose a domain block that minimize a predefined distance criterion. Mean square error (MSE) is usually used as the distance criterion and is computed by

$$d(R,D) = MSE(sD+o,R) = \frac{1}{N}\sum_{i=1}^{N}(sd_i + o - r_i)^2 \tag{2}$$

where $N$ is the number of pixels in the range block, $r_i$ is the $i$th pixel value in range $R$ , and $d_i$ is that of the domain $D$. Note that domain block is spatial contracted to the same size of the range.

Before we can evaluate the distance between $R$ and $D$, we must decide best contrast scaling ($s$) and intensity offset ($o$). Two partial differentiations on $s$ and $o$ in Equation (2) lead to Equations (3) and (4), as shown in Ref. [6].

$$\begin{aligned} s &= \frac{\left[N\sum_{i=1}^{N} d_i r_i - \sum_{i=1}^{N} d_i \sum_{i=1}^{N} r_i\right]}{\left[N\sum_{i=1}^{N} d_i^2 - (\sum_{i=1}^{N} d_i)^2\right]} \\ &= \frac{\frac{1}{N}\sum_{i=1}^{N} d_i r_i - \bar{D}\bar{R}}{Var(D)} \end{aligned} \tag{3}$$

$$\begin{aligned} o &= \frac{1}{N}\left[\sum_{i=1}^{N} r_i - s\sum_{i=1}^{N} d_i\right] \\ &= \bar{R} - s\bar{D} \end{aligned} \tag{4}$$

$$\begin{aligned} \bar{R} &= \frac{1}{N}\sum_{i=1}^{N} r_i \\ \bar{D} &= \frac{1}{N}\sum_{i=1}^{N} d_i \end{aligned}$$

where $\bar{D}$ and $\bar{R}$ are mean value of domain and range, respectively, and $Var(D)$ is the variance of the domain block. If $Var(D) = 0$, then $s = 0.0$ and $o = \bar{R}$. Further details about fractal image compression can be found in Ref. [6].

## 2.2 The Fractal Image Decoding

We have described the baseline encoding procedure in Section 2.1. To completeness, a brief description of the decoding process is included.

The fractal image decoding process consists of setting up an initial image and then updating each range block. The update is a transformation on the matched domain block according to the fractal codes $s$ and $o$ as specified by Equation (3) and (4), respectively. Thus, the transformation is described by

$$r_i = sd_i + o \tag{5}$$

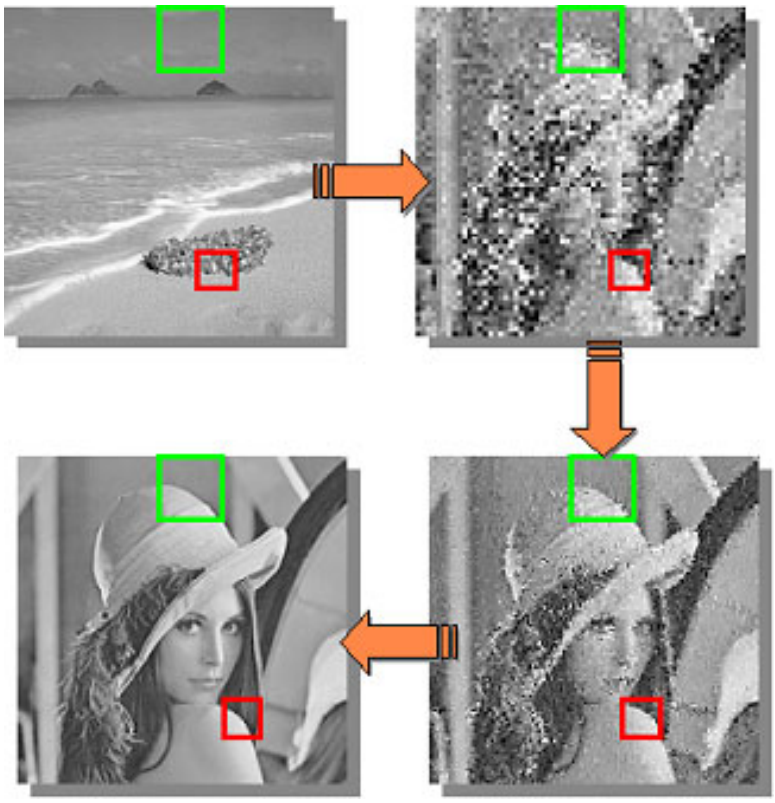

Figure 1: Illustration of the decoding process.

After performing iterations of such update, the image should converge to a stable image and the decoding process can be stoppted.

Figure 1 shows the intermediate results of decoded images during decoding process. For each iteration, we transform content of domain block (bounded by green rectangle) to range block (bounded by red rectangle). Note that the decoding process involves only spatial transformations while no transformation to frequency domain is needed.

## 3 Fast Fractal Image Encoding

As mentioned in Section 2, fractal image encoding suffers from long search time to find a close match between a range block and a large pool of domain blocks. Each pair of range-domain match involves time consuming mean square error (MSE) calculation since each calculation needs a lot of time to get contrast scaling by a complicated formula. In this section, two new techniques are proposed to reduce the number of MSE calculations and the time to accomplish an MSE calculation, which, in turn, speed up the encoding time.

More specifically, the proposed methods of reducing the number of MSE calculations can be divided into two categories: fixed pruning and adaptive pruning. The former reduces the domain pool for all the range blocks, while the latter prunes respective unqualified domain blocks for individual range blocks. In other words, the former is independent on range block, while the latter is dependent on range block. These two pruning methods are described in Sections 3.2 and 3.4, respectively. The adaptive pruning is achieved by the strategy of adaptive search that originates from Ref. [1] and is modified with estimated contrast scaling. Hence, a simple equation to approximate contrast scaling is first derived in Section 3.3, followed by the modified version of adaptive search in Section 3.4. On the other hand, the estimated contrast scaling can also be used to reduce time taken to calculate each MSE as described in Section 3.5. Some miscellaneous tips for fast encoding are introduced in Section 3.6. Finally the proposed encoding procedure is summarized in Section 3.7.

### 3.1 Mathedology of Fractal Encoding

The calculation of mean square error between two image blocks is the most important part of the fractal image encoding. In this section we introduce some mathematical terms first and then trasform the MSE distance into a new form that will be used in the later sections.

When calculating the MSE distance between a range block and a domain block, we allow the domain block adjusted by contrast scaling and intensity offset to approximate the range. Let $r'_i$ represent the intensity of $i$th pixel of the approximated range and $s$ and $o$ be the contrast scaling and intensity offset, respectively, i.e.,

$$r'_i = sd_i + o \tag{6}$$

Substituting Equation (4) into Equation (6), we then have

$$r'_i = sd_i + \bar{R} - s\bar{D} \tag{7}$$

where $\bar{R}$ and $\bar{D}$ are the pixel mean of range block and domain block, respectively.

Accordingly, the MSE distance between $R$ and $D$ can then be derived as follows

$$\begin{aligned} MSE(sD+o,R) &= \frac{1}{N}\sum_{i=1}^{N}(r'_i - r_i)^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}[(sd_i + \bar{R} - s\bar{D}) - r_i]^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}[s(d_i - \bar{D}) - (r_i - \bar{R})]^2 \end{aligned} \tag{8}$$

$$
\begin{aligned}
&= s^2\frac{1}{N}\sum_{i=1}^{N}(d_i-\frac{1}{N}\bar{D})^2+\frac{1}{N}\sum_{i=1}^{N}(r_i-\bar{R})^2-2s\frac{1}{N}\sum_{i=1}^{N}(d_i-\bar{D})(r_i-\bar{R}) \\
&= s^2Var(D)+Var(R)-2s\frac{1}{N}\sum_{i=1}^{N}(d_i-\bar{D})(r_i-\bar{R}) \qquad (9)
\end{aligned}
$$

where $Var(D)$ and $Var(R)$ are pixel variance of domain and range, respectively.

Further substituting contrast scaling ($s$) in Equation (3) to Equation(9) leads to

$$
\begin{aligned}
MSE(sD+o,R) &= \frac{(\frac{1}{N}\sum_{i=1}^{N}d_ir_i-\bar{D}\bar{R})^2}{Var(D)^2}Var(D)+Var(R) \\
&\quad -2\frac{(\frac{1}{N}\sum_{i=1}^{N}d_ir_i-\bar{D}\bar{R})}{Var(D)}[\frac{1}{N}\sum_{i=1}^{N}(d_i-\bar{D})(r_i-\bar{R})] \\
&= Var(R)-\frac{(\frac{1}{N}\sum_{i=1}^{N}d_ir_i-\bar{D}\bar{R})^2}{Var(D)} \qquad (10)
\end{aligned}
$$

Equation (10) can be rearranged to be

$$
MSE(sD+o,R)=Var(R)-s^2Var(D) \qquad (11)
$$

### 3.2 Reduction of Domain Pool

To find a close match for a range block from a large pool of domain blocks during encoding is time expensive. However, for most images, a fraction of domain blocks are similar to each other so some domain blocks can be pruned and only representative blocks are preserved. In this way, the number of domains in the search space can be reduced and search time can be saved. Thus, a simple, yet effective method is proposed to achieve the goal mentioned above. Note that this reduction is performed before encoding.

The following propositions used for reduction are stated first.

**Proposition 1** *If two domain blocks $A$ and $B$ are equal, then $MSE(s_aA+o_a,R)=e_1$ implies $MSE(s_bB+o_b,R)=e_1$, where $s_a$, $o_a$, $s_b$, $o_b$ are calculated by Equations (3) and (4), respectively.*

**Proof 1** *Two blocks $A$ and $B$ are equal means that $a_i=b_i$ for all pixels. Thus, we can replace $a_i$ by $b_i$ in Equations (3), (4) and (8) and get*

$$
MSE(s_aA+o_a,R)=MSE(s_bB+o_b,R) \qquad (12)
$$

□

This proposition can be used to prune equal domains. However, the reduction rule can not achieve high pruning rate.

If we can allow one domain block to be adjusted by an intensity offset and ensure that the property of Proposition 1 still holds, the pruning rate can be improved to a considerable level. So, we derive Proposition 2 on the basis of a new definition.

**Definition 1** *Two (domain) blocks A and B are called similar if* $MSE(B+t, A) = 0$, *where* $t = \bar{A} - \bar{B}$ *with* $\bar{A}$ *and* $\bar{B}$ *as the mean values of A and B, respectively.*

**Proposition 2** *If two domain blocks are similar, then* $MSE(s_aA+o_a, R) = e_1$ *implies* $MSE(s_bB + o_b, R) = e_1$.

**Proof 2** *From Equation (10), we have*

$$MSE(s_aA + o_a, R) = Var(R) - \frac{(\frac{1}{N}\sum_{i=1}^{N} a_i r_i - \bar{A}\bar{R})^2}{Var(A)} \tag{13}$$

*Due to Definition 1, we have*

$$a_i = b_i + t = b_i + \bar{A} - \bar{B} \tag{14}$$

*Substitute Equation (14) into Equation (13), we have*

$$\begin{aligned}
MSE(s_aA + o_a, R) &= e_1 = Var(A) - \frac{(\frac{1}{N}\sum_{i=1}^{N} a_i r_i - \bar{A}\bar{R})^2}{Var(R)} \\
&= Var(A) - \frac{(\frac{1}{N}\sum_{i=1}^{N} (b_i + \bar{A} - \bar{B}) r_i - \bar{A}\bar{R})^2}{Var(R)} \\
&= Var(A) - \frac{(\frac{1}{N}\sum_{i=1}^{N} (b_i r_i + \bar{A} r_i - \bar{B} r_i) - \bar{A}\bar{R})^2}{Var(R)} \\
&= Var(A) - \frac{(\frac{1}{N}\sum_{i=1}^{N} b_i r_i + \bar{A}\frac{1}{N}\sum_{i=1}^{N} r_i - \bar{B}\frac{1}{N}\sum_{i=1}^{N} r_i - \bar{A}\bar{R})^2}{Var(R)} \\
&= Var(A) - \frac{(\frac{1}{N}\sum_{i=1}^{N} b_i r_i + \bar{A}\bar{R} - \bar{B}\bar{R} - \bar{A}\bar{R})^2}{Var(R)} \\
&= Var(A) - \frac{(\frac{1}{N}\sum_{i=1}^{N} b_i r_i - \bar{B}\bar{R})^2}{Var(R)}
\end{aligned} \tag{15}$$

*On the other hand, we have* $Var(A) = Var(B)$, *since* $a_i = b_i + t$ *for all i. Thus, we can substitute the equality into Equation (15) to get*

$$\begin{aligned}
MSE(s_aA + o_a, R) &= Var(B) - \frac{(\frac{1}{N}\sum_{i=1}^{N} b_i r_i - \bar{B}\bar{R})^2}{Var(R)} \\
&= MSE(s_bB + o_b, R) = e1
\end{aligned} \tag{16}$$

□

On the basis of Proposition 2, a new reduction rule is proposed as follow. For a new domain block $B$, the MSE distance between $B$ and each domain block $A_j$ is denoted by $MSE(B+t_j, A_j)$, where $t_j = \bar{A}_j - \bar{B}$. If there exists any $j$ so that $MSE(B + o_b, A_j) = 0$, then $B$ is excluded from the pool. In other words, if $MSE(B + t_j, A_j) > 0$ for all $j$, then $B$ is added to the domain pool. In practice, we give a threshold value for the MSE distance in the above rule. If there exists any $j$ so that $MSE(B + t_j, A_j)$ is less than a threshold value, $B$ is excluded from the pool.

To speed up the reduction procedure, all the domain blocks are preclassified into proper classes. The reduction rule is then individually applied to each class rather than the whole domain pool so that execution time can be degraded. The classification strategy is based on the conjecture that two blocks with diverse variances are not likely to have low MSE distance. So, classify domain blocks into proper groupes according to their variances, then each new block $B$ is compared only to those domain blocks from the same class as shown in Figure 2. Experimental results show that performing classification technique will not sacrifice the quality of reduced domain pool.

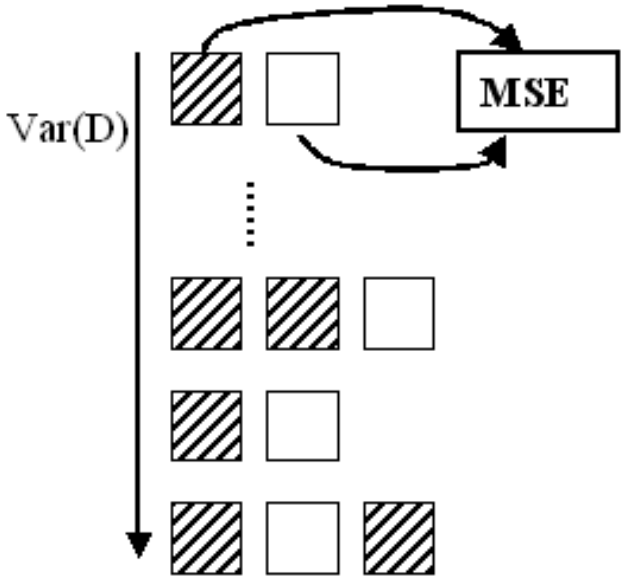


Figure 2: Illustration of domain pool reduction. Each row includes those domain blocks with the same variance $Var(D)$. The MSE distance between each pair of domain blocks in computed as MSE. Squares with slashes denote preserved blocks.

### 3.3 Estimation of Contrast Scaling

Because the calculation of contrast scaling (Equation (3)) is complicated, instead of direct calculation of $s$, Ref. [1] tried to calculate MSE for each possible case of $s$. However, the execution time can be further speeded up if

we can estimate $s$ in an easier way and the estimated value does approximate the optimal one. Then, we can evaluate MSE just once for each domain rather than the number of quantized values of $s$ (for examples four in Ref. [1]).

In this section, we first describe the basic idea of estimating contrast scaling, followed by some experimental results to demonstrate that the estimated value approximates the optimal one and does speed up the encoding time. Finally, a mathematical explanation is included to show why the estimated value works.

As mentioned in Section 2.1, the solution of optimal contrast scaling (Equation (3)) is derived from a partial differentiation on the MSE. It is the optimal value to minimize the MSE distance between two image blocks. An intuitive thought on the function of the contrast scaling is to adjust standard deviation of a block so that it can match that of the other block. In this way, it seems that $\frac{Std(D)}{Std(R)}$ is a good estimated value, where $Std(D)$ is the standard deviation of the adjusted block and $Std(R)$ is that of the matched block. Therefore, we estimate the contrast scaling $s'$ by

$$s' = \frac{Std(D)}{Std(R)} \tag{17}$$

Table 1 shows encoding results of using optimal and estimated contrast scaling to calculate MSE. Conventional method tries all possible cases (four in our experiment) and find the optimal one, while our method tries to get the estimated value using Equation (17) and then quantizes it. The encoding time is lowered while PSNR values do not decrease seriously.

Table 1: Comparison of conventional and estimated contrast scaling.

| | $256 \times 256$ Lenna | | $512 \times 512$ Lenna | |
|---|---|---|---|---|
| | Time (sec.) | PSNR | Time (sec.) | PSNR |
| trials of $s$ | 36 | 29.225 | 604 | 35.757 |
| estimation of $s$ | 24 | 29.330 | 406 | 35.628 |

To explain why estimated contrast scaling works, we divide image blocks into two types, uniform and non-uniform blocks, according to variance of the pixel values in the block. Figure 3 shows that uniform ranges tend to find uniform domains as their closest match. That means the $Var(R)$ and $Var(D)$ are small in this case. Accordingly, from Equation (9) we can conclude that the MSE distance is small no matter what value the contrast scaling ($s$) is. Note that the experiment is applied to a $256 \times 256$ Lenna image.

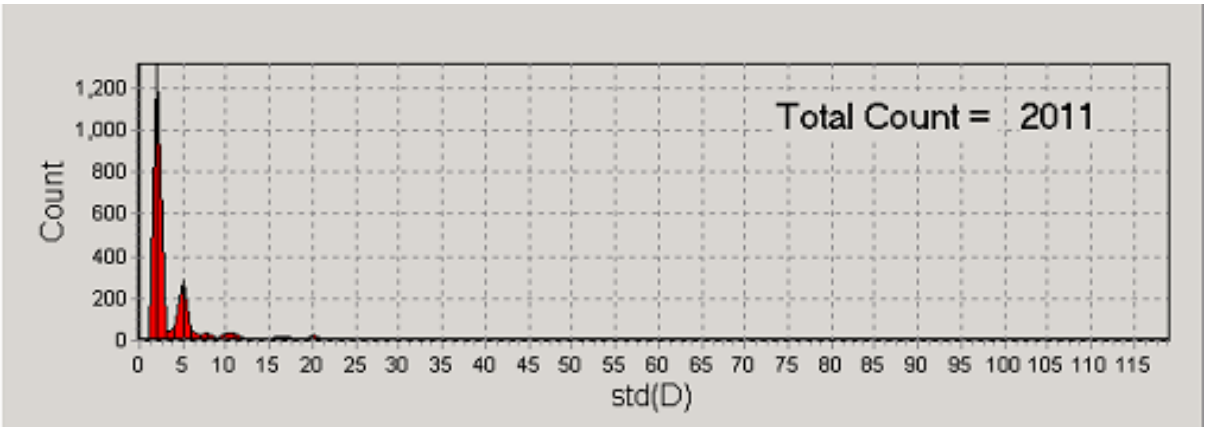


Figure 3: Variance distribution of closest matched domains for uniform ranges.

For non-uniform ranges, Equation (11) should be reexpressed by

$$\frac{MSE(sD+o,R)}{Var(D)} = \frac{Var(R)}{Var(D)} - s^2 \tag{18}$$

Substituting Equation (17) into the above equation, we can get

$$\frac{MSE(sD+o,R)}{Var(D)} = (s')^2 - s^2 \tag{19}$$

If a domain $D$ is a match of a non-uniform range $R$, the MSE distance $MSE(sD+o,R)$ is low. In addition, experimental result shows that the variance of this domain, $Var(D)$, is large (see Figure 4). Since $MSE(SD+o,R)$ is small and $Var(D)$ is large, from Equation (19), we can conclude that $(s')^2 - s^2$ approaches zero, that means estimated contrast scaling ($s'$) approaching the optimal one ($s$). The relationship of $s$ and $s'$ of all matched range-domain pairs with respect to uniform and non-uniform ranges are shown in Figure 5. The nodes labeled by color blue and red denote uniform and non-uniform ranges, respectively.

Above explanation leads to the following conclusion. If a domain is the match of a range by optimal contrast scaling, we would not miss it by using the estimated contrast scaling. On the other hand, if a domain does not match a range by the optimal contrast scaing, the estimated value leads to larger MSE distance because $s$ is the optimal value to minimize the MSE distance. Therefore, we would not find the wrong domain as the closest match even using the estimated contrast scaling.

### 3.4 Adaptive Search of Modified Version

The adaptive search rule is proposed in Ref. [1]. It's a simple yet effective technique. When searching a domain for the encoding range, only the

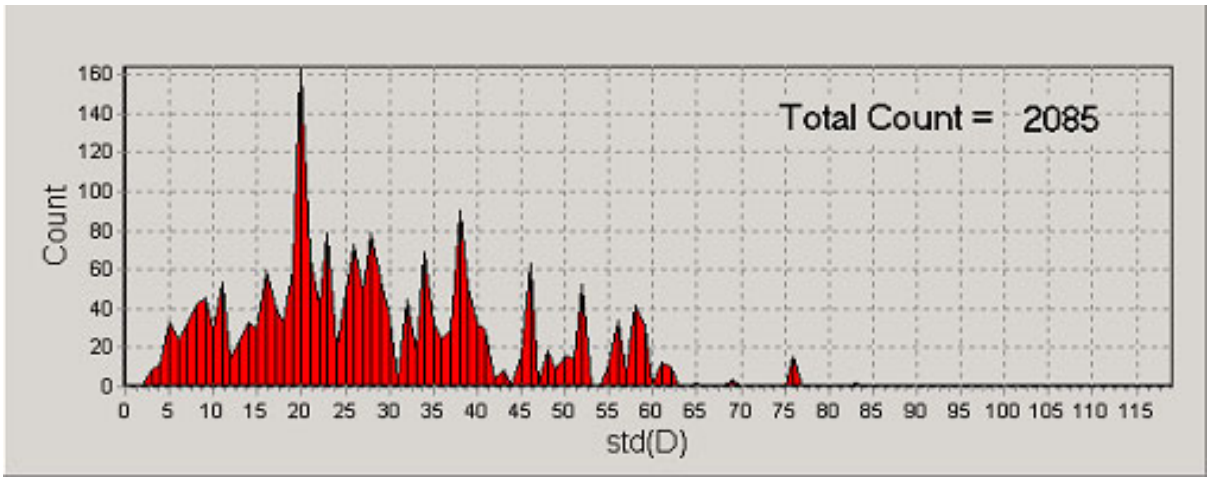


Figure 4: Variance distribution of closest matched domains for non-uniform ranges.

domain satisfing the following equation is considered.

$$|\sigma_r - s'\sigma_d| \leq T_1 = Std(R)\theta \tag{20}$$

where $T_1$ is a threshold value and it is adaptive because it consists of a constant $\theta$ and the standard deviation of the range $Std(R)$. Note taht the adaptive search in Ref. [1] is modified by replacing $s$ by $s'$ in this study. Thus the criterion can be checked only once rather than four times in this study.

### 3.5 Time Reduction for MSE Calculation

In practice, we calculate the MSE distance by Equation (9) and replace $s$ by the estimated contrast scaling $s' = \frac{Std(R)}{Std(D)}$. Thus, the MSE calculation can be expressed by

$$MSE(sD + o, R) = (s')^2 Var(D) + Var(R) - 2(s')\frac{1}{N}\sum_{i=1} N(d_i - \bar{D})(r_i - \bar{R}) \tag{21}$$

Note that $\bar{D}$, $\bar{R}$, $Var(D)$ and $Var(R)$ can be calculated in advance.

### 3.6 Miscellaneous Tips to Speed-up

#### 3.6.1 Rotation on Range

To find a closeest domain for a range, each domain can be rotated in eight orientations. However, rotation of domain is time consuming and cannot be

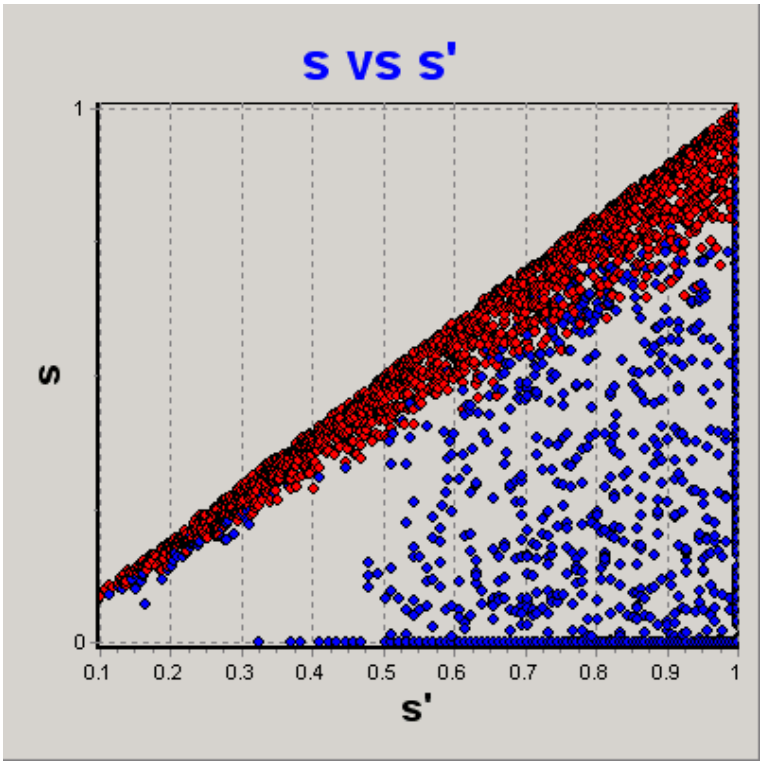


Figure 5: Relationship of optimal ($s$) and estimated ($s'$) contrast scaling of all matched range-domain pairs. The node labeled by color blue and red denote uniform and non-uniform ranges, respectively.

done in advance since storage of rotation results for all domains would take too much memory space.

Obviously, rotations on range have the same effect as rotation on domain. For example, rotating domain 180 degrees yields the same results as rotating range 180 degrees. Flipping the domain over and rotating 90 degrees clockwise has the same effect as done on range. There are two exceptions, rotating domain 90 degree *clockwise* is the same as rotating range 90 degree *counterclockwise*, and vice versa.

According to the observation mentioned above, we rotate the range in eight possible orientations and store them all. Here, when calculating MSE, only the term of $\sum(d_i - \bar{d})(r_i - \bar{r})$ in Equation (21) needs recalculation for different pre-rotated range and no more rotations on domain are needed.

### 3.6.2 The Revised Fractal Code

In this section, revised fractal code is introduced to facilitate fast encoding. The idea is to encode range block intensity mean value, $\bar{R}$, instead of the

intensity offset, $o$. Note that the former encoding is expressed by Equation (7), while the latter is by Equation (6). Actually, this idea is not new. It has been mentioned in Ref. [1] and many other papers.

This change also affects the decoding process. Since the code $\bar{R}$ indicates the mean value of each range block of the target image, the initialized image for decoding can be set to the mean values instead of random values. This results in fast decoding with fewer iterations. The number of iterations needed for the $256 \times 256$ Lenna image to converge is listed in Table 2. However, there is one drawback of using revised fractal code, i.e., we have to calculate $\bar{D}$ in Equation (7) during decoding. Fortunatelly, this additional computation can be compensated by the reduction of number of iterations.

Table 2: Comparison of number of iterations using different fractal code to Lenna image.

| | traditional fractal code | new fractal code |
|---|---|---|
| initial image | 0.0 | 23.115 |
| itr. 1 | 13.839 | 27.845 |
| itr. 2 | 20.383 | 29.228 |
| itr. 3 | 26.017 | *30.009 |
| itr. 4 | 29.301 | |
| itr. 5 | *30.401 | |
| itr. 6 | 30.692 | |

### 3.7 The Encoding Procedure

The proposed encoding procedure is summarized in this section. We describe parameters needed by the encoding system first. The range side length, domain side length and domain shift amount are needed to produce range and domain pool. We have introduced two threshold values in this section. $T_0$ represents root mean square error between different domains and is used to reduce domain pool. $T_1$ is the adaptive search threshold and is used to decide what domains are worth trying.

Here we introduce the third threshold, which is commonly used by most fractal image encoding approaches, i.e., the stop threshold. If there is a domain whose root mean square error between it and the encoding range is less than $T_2$, the encoding result is accepted and the encoding is terminated for that range.

The summarized encoding procedure is listed below.

**Procedure 2. The oroposed fractal image encodingn.**
Partition the image into range blocks.

Procude the reduced domain pool ($T_0$ is used here).
For each range: $R$,
Quantize $\bar{r}$
Rotating the range and store the rotating results as: $R_1, \ldots R_8$
Out of all possible domain blocks: $D$,
    Calculate $s'$ by Equation (17).
    If $D$ satisfies Equation (20), proceed, else find next $D$
      Contrast $D$ to the same size of $R$.
      Calculate $MSE$ against $R_1, \ldots R_8$ by Equation (21).
      Update best domain according to $MSE$.
      If $MSE < (T_2)^2$, stop encoding for $R$
    Store the fractal code.

# 4 Experimental Results

Experiments in this study are conducted on an computing environment described in Table 3. In addition, only the uniform partition scheme is concerned in this study.

Some notations used in this section are defined first. 'RS' stands for 'range side length', 'DS' stands for 'domain side length' and 'DSh' for 'domain shift amount'. Four threshold values introduced in previous section are also included, $T_0$ is used by domain pool reduction, $T_1$ is used by adaptive search, and $T_2$ is used to determine whether to stop searching or not. $Avg.\#MSE$ denotes the averaged number of mean square error calculated for each range block, and $\#D$ is the number of blocks left in the domain pool. Besides, N/A stands for not available.

Table 3: Computing environment.

| | |
|---|---|
| CPU | Intel Celeron 1000 MHz |
| Memory | $128 \times 3 = 384$ MB (PC133) |
| Operating System | Windows 2000 |
| Programming Language | C++ |
| Compiler | gcc 2.95.3-5 (for encoder)<br>Borland C++ 5.5 (for decoder) |

Tables 4 and 5 list encoding results of $256 \times 256$ and $512 \times 512$ Lenna images, respectively. Baseline encoding encodes the image without performing domain pool reduction and contrast scaling estimation, but with the miscellaneous techniques including revised fractal code and rotation on range. The reduced domain pool experiment is done without performing contrast

scaling estimation, and so on. As we can see, the encoding time is propotional to the number of MSE, which in turn, is propotional to number of domains times number of ranges.

From these experimental results, we can see that both domain pool reduction and modified adaptive search are effective. For $256 \times 256$ Lenna image, using only modified adaptive search seems sufficient. However, when the image size increases from $256 \times 256$ to $512 \times 512$, encoding time increases from 3 to 33 seconds, if only the adaptive search is performed. That is because the number of blocks (both domain and range) increases as fast as the image size does, while the proposed domain pool reduction method can decrease the number of domain blocks effectively, in particular when image size is large.

Table 4: Experimental results to $256 \times 256$ Lenna image

| | RS | DS | Dsh | $T_0$ | $T_1$ | $T_2$ | *Avg.* $\#MSE$ | Time (sec.) | PSNR |
|---|---|---|---|---|---|---|---|---|---|
| Baseline | 4 | 8 | 8 | N/A | N/A | N/A | 8061 | 36 | 29.330 |
| Reduced domain pool | 4 | 8 | 8 | 1 | N/A | 3 | 1832 | 9 | 29.190 |
| Modified Adaptive Search | 4 | 8 | 8 | 1 | $Std(R)/8$ | 3 | 757 | 3 | 28.997 |
| Combined | 4 | 8 | 8 | 1 | $Std(R)/8$ | 3 | 432 | 2 | 28.860 |

Table 5: Experimental Results to the $512 \times 512$ Lenna image

| | RS | DS | Dsh | $T_0$ | $T_1$ | $T_2$ | *Avg.* $\#MSE$ | Time (sec.) | PSNR |
|---|---|---|---|---|---|---|---|---|---|
| Baseline | 4 | 8 | 8 | N/A | N/A | N/A | 31876 | 604 | 35.757 |
| Reduced domain pool | 4 | 8 | 8 | 1 | N/A | 3 | 2309 | 43 | 35.159 |
| Modified Adaptive Search | 4 | 8 | 8 | 1 | $Std(R)/8$ | 3 | 1877 | 33 | 35.296 |
| Combined | 4 | 8 | 8 | 1 | $Std(R)/8$ | 3 | 558 | 9 | 34.914 |

Table 6 shows encoding results of using different parameters. If encoding speed is mainly concerned, parameters could be set to that in the fourth row and the $512 \times 512$ Lenna image can be encoded in 3 seconds with PSNR value over 33. On the other hand, if we focus on image fidelity, parameters could be set to that in the first row, then taking 31 seconds can achieve

PSNR value over 35. When we set parameters to taht in the last row, we can achieve PSNR value over 41 and take only 7 seconds. However, in this case, more spaces are required to store encoding codes.

Table 6: Encoding results of using different parameters to $512 \times 512$ Lenna image.

| RS | DS | DSh | $T_0$ | $T_1$ | $T_2$ | $\#D$ | Time (sec.) | PSNR |
|---|---|---|---|---|---|---|---|---|
| 4 | 8 | 4 | **1** | $\sigma_r/8$ | 3 | 2030 | 31 | 35.521 |
| 4 | 8 | 4 | **3** | $\sigma_r/8$ | 3 | 374 | 6 | 34.257 |
| 4 | 8 | **8** | 1 | $\sigma_r/8$ | 3 | 655 | 9 | 34.914 |
| 4 | 8 | 8 | **3** | $\sigma_r/8$ | 3 | 176 | 3 | 33.870 |
| 2 | 4 | 4 | 3 | $\sigma_r/8$ | 3 | 541 | 7 | 41.611 |

Table 7 focuses on how much time domain pool reduction technique can reduce. The first row corresponds to encoding the image without performing domain pool reduction. As shown in the second row, if domain pool reduction is applied, then encoding time can be decreased to less than half of the original one while PSNR loss is very small ($< 0.3dB$). This is the reason why we set $T_0$ to 1.

Table 7: Without and with domain pool reduction to $512 \times 512$ Lenna image.

| RS | DS | DSh | $T_0$ | $T_1$ | $T_2$ | $\#D$ | Time (sec.) | PSNR |
|---|---|---|---|---|---|---|---|---|
| 4 | 8 | 8 | N/A | $\sigma_r/8$ | 3 | 4096 | 33 | 35.296 |
| 4 | 8 | 8 | 1 | $\sigma_r/8$ | 3 | 655 | 9 | 34.914 |
| 4 | 8 | 8 | 2 | $\sigma_r/8$ | 3 | 334 | 5 | 34.385 |
| 4 | 8 | 8 | 3 | $\sigma_r/8$ | 3 | 176 | 3 | 33.877 |
| 4 | 8 | 8 | 4 | $\sigma_r/8$ | 3 | 105 | 2 | 33.361 |

In addition to getting fast encoding speed, we also emphasize that our method is insensitive to image size. To show this point, we encode Lenna image of different sizes. The experimental results are shown in Figure 6. The parameters are the same as those in Table 4.

Comparison of our method to others, including *lean domain pool reduction* [11] and *adaptive search* [1], are also presented in Figure 6. The experimental results show that our method is most effective.

## 5 Conclusions

In this study, an encoding strategy that combines different speed up techniques are described. Our method focuses on reducing encoding time while

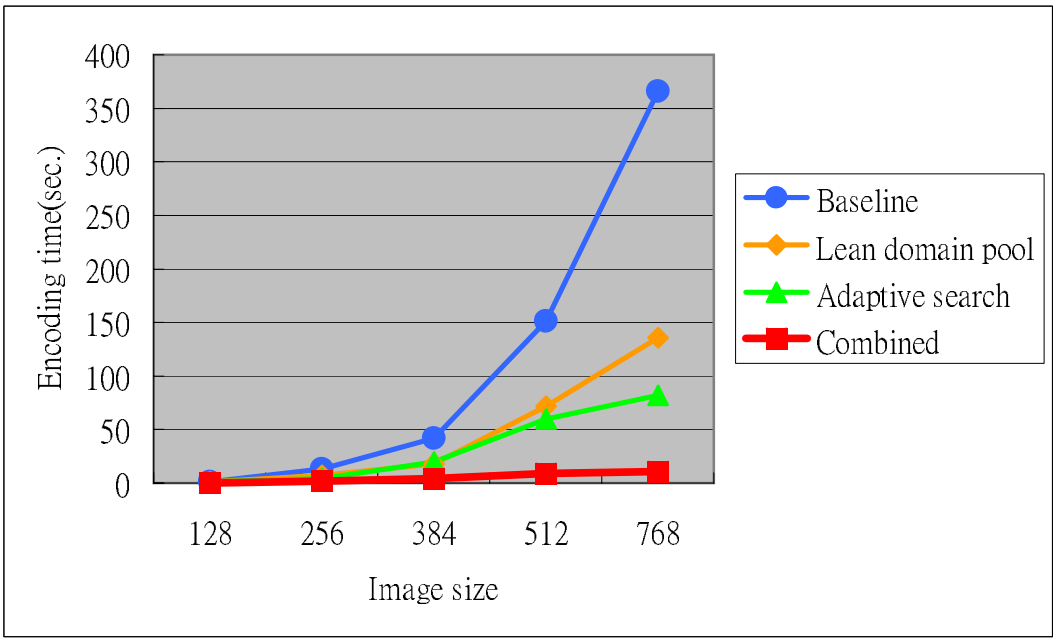


Figure 6: Encoding time of different methods versus different image sizes.

maintaining acceptable PSNR values. The experimental results show that our method is superior to current methods such as Ref. [1].

We also emphasize that our method is insensitive to image size. For example, when the size of image increases from $256 \times 256$ to $512 \times 512$ , the encoding time of our method increases from 2 to 9 seconds, while the encoding time increases from 7 to 125 seconds in Ref [1].

Future works can be directed to the following topics:

(1) Speed up encoding time using the pairwise relationship of range and domain and the spatial relationship of neighboring ranges.

(2) Improve PSNR.

(3) Improve compression ratio.

(4) Apply fractal image encoding to frequency domain.

(5) Extend the proposed method to sequence images.

(6) Since the encoding results, namely fractal codes, provide useful information, they cen be used for many applications. Such as, image retrieval [12] and contour detection [13].

## References


[1] C. S. Tong and M. Pi, "Fast fractal image encoding based on adaptive search," *IEEE Trans. Image Processing*, vol. 10, pp. 1269–1277, Sep. 2001.

[2] M. F. Barnsley and S. Demko, "Iterated function system and the global construction of fractals," *Proc. Roy. Soc.*, vol. A399, pp. 243–275, 1985.

[3] M. F. Barnsley, V. Ervin, D. Hardin, *et al.*, "Solution of an inverse problem for fractals and other sets," *Proc. Nat. Acad. Sci.*, vol. 83, pp. 1975–1977, 1986.

[4] M. F. Barnsley, *Fractals Everywhere*. New York: Academic Press, 1988.

[5] A. E. Jacquin, "Image coding based on fractal theory of iterated contractive image transformations," *IEEE Trans. Image Processing*, vol. 1, pp. 18–30, Jan. 1992.

[6] Y. Fisher, ed., *Fractal Image Compression Theory and Application*. New York: Springer-Verlag, 1995.

[7] T. K. Trung, J. H. Jeng, I. S. Reed, *et al.*, "A fast encoding algorithm for fractal image compression using the dct inner product," *IEEE Trans. Image Processing*, vol. 9, pp. 529–535, Apr. 2000.

[8] C. K. Lee and W. K. Lee, "Fast fractal image block coding on local variances," *IEEE Trans. Image Processing*, vol. 7, pp. 888–891, Jun. 1998.

[9] M. Polvere and M. Nappi, "Speed-up in fractal image coding: Comparison of methods," *IEEE Trans. Image Processing*, vol. 9, pp. 1002–1009, Jun. 2000.

[10] B. Mandelbrot, *The Fractal Geometry of Nature*. Freeman, 1982.

[11] D. Saupe, "Lean domain pools for fractal image compression," *Conf. Proc. SPIE Electronic Imageing '96, Science and Technology, Still Image Compression II*, vol. 2669, 1996.

[12] G. T. M. Nappi, G. Polese, "First: Fractal indexing and retrieval system for image database," *Image and Vision Computing*, vol. 16, pp. 1019–1031, 1998.

[13] Y. S. Takashi Ida, "Self-affine mapping system and its application to object contour extraction," *IEEE Trans. Image Processing*, vol. 9, pp. 1926–1936, 2000.