Fractal image compression using visual-based particle swarm optimization

Chun-Chieh Tseng\textsuperscript{a,}\ast, Jer-Guang Hsieh\textsuperscript{a}, Jyh-Horng Jeng\textsuperscript{b}

\textsuperscript{a}Department of Electrical Engineering, National Sun Yat-Sen University, Kaohsiung 804, Taiwan
\textsuperscript{b}Department of Information Engineering, I-Shou University, Kaohsiung County 840, Taiwan

Abstract

Fractal image compression is promising both theoretically and practically. The encoding speed of the traditional full search method is a key factor rendering the fractal image compression unsuitable for real-time applications. In this paper, particle swarm optimization (PSO) method by utilizing the visual information of the edge property is proposed, which can speedup the encoder and preserve the image quality. Instead of the full search, a direction map is built according to the edge-type of image blocks, which directs the particles in the swarm to regions consisting of candidates of higher similarity. Therefore, the searching space is reduced and the speedup can be achieved. Also, since the strategy is performed according to the edge property, better visual effect can be preserved. Experimental results show that the visual-based particle swarm optimization speeds up the encoder 125 times faster with only 0.89 dB decay of image quality in comparison to the full search method.

Keywords: Fractal image compression; Particle swarm optimization; Edge-type classification

1. Introduction

The idea of the fractal image compression (FIC) is based on the assumption that the image redundancies can be efficiently exploited by means of block self-affine transformations [1,2]. The fractal transform for image compression was introduced in 1985 by Barnsley and Demko [3]. The first practical fractal image compression scheme was introduced in 1992 by Jacquin [4]. One of the main disadvantages using exhaustive search strategy is the low encoding speed. Therefore, improving the encoding speed is an interesting research topic for FIC.

Fisher’s classification method [5] constructed 72 classes for image blocks according to the variance and intensity. Since the search space is divided into 72 smaller classes and the search is performed only in the corresponding class, encoding speed can be improved. In Wang et al. [6], four types of range blocks were defined based on the edge property of the image. Such a method can provide speedup ratio from 1.6 to 5. Truong et al. [7] proposed Discrete Cosine Transform (DCT) inner product based algorithm which removes all of the redundant calculations for the eight Dihedral symmetries of the domain block to achieve a faster encoding process. Another approach to speedup the encoder is to use the stochastic optimization methods such as genetic algorithm (GA). The previous studies using these types of searching strategies do speedup the encoder, but the speedup ratio is still low and cannot preserve the image quality well [8,9]. Some recent GA-based methods are proposed to improve the efficiency [10,11]. These methods make use of the idea of special correlation of an image, which is of great interesting. While the chromosomes in GA consist of all range blocks and particular properties of natural images have never been used, image compression requires a great amount of computations and the retrieved image will lose the visual effect.

\ast Corresponding author. Tel.: +886 75254139.
E-mail addresses: d9131809@student.nsysu.edu.tw, nickt@anet.net.tw (C.C. Tseng).
In this paper, we adopt the particle swarm optimization (PSO) as an alternative searching method which provides another point of view about speedup. The proposed algorithm fuses the domain knowledge of image property, PSO model, and FIC coding scheme together to achieve the speedup purpose and preserve the retrieved image quality.

PSO is an optimization algorithm having origins from evolutionary computation together with the social psychology principle. Essentially, PSO is dependent on stochastic processes and also uses the concept of fitness, as do in GA. In addition, it provides a mechanism such that individuals in the swarm communicate and exchange information, which is similar to the social behavior of insects and human beings. Because of the mimicking of the social sharing of information, PSO directs the particles to search the solution more efficiently [12,13].

Another important feature of PSO is that the paradigm requires only primitive mathematical operations which can be implemented in a few lines of computer code. As PSO is computationally inexpensive and adopts models of social behaviors, it has attracted many real-world applications such as the analysis of human tremor, the reactive power and voltage control, the state-of-charge estimator for a battery pack, and the ingredient mix optimization. Many of them have been shown to perform successfully [14,15]. For fractal image compression, one will use PSO to find the best matched domain block for a given range block. Since the exploration of the search space is not exhaustive, encoding time can be improved.

At each search entry, the additional movement of particles can be directed according to the image edge properties. Instead of finding the best match in a huge searching space, the proposed schemes that work with the edge-type classifier can avoid a large amount of Mean Square Error (MSE) calculations. Edge-type classifier divides image blocks into five classes, i.e., smooth class, horizontal/vertical classes, and diagonal/sub-diagonal classes. Intuitively, image blocks of the same edge-type are promising to have high self-similarity. For a given range block, strategies according to the edge-type of the domain block at current search entry are applied to speedup the encoder. These strategies make use of the properties of Dihedral transformations subtly to control the behavior of particles in the swarm.

The structure of the edge-type classifier is designed in favor of the eight Dihedral transformations of fractal coder. For instance, for blocks of “vertical” type, some of its transformations are also of “vertical” type. Such properties are utilized to reduce the calculations of Dihedral transformations. Also, edge information is managed as a direction map controlling the movement of the particles in the swarm. That is, particles search in the landscape by following the directions of blocks with similar edge property and jump away from the regions of blocks with different edge-types. This method mimics the human visual behavior, and therefore may improve the performance of the fractal coder.

The edge-type classifier utilizes only two DCT coefficients, i.e., the lowest horizontal and vertical coefficients, to represent the strength of the horizontal and vertical energy variation in the block, respectively. Since only two DCT coefficients are involved, little overhead, in comparison to the complexity of the encoder, is imposed. Moreover, since the classifier is performed according to the edge property, the image quality can be preserved and better visual effect can be achieved.

2. Full search fractal image compression

The fundamental idea of fractal image compression is the Iteration Function System (IFS) in which the governing theorems are the Contractive Mapping Fixed-Point Theorem and the Collage Theorem [16]. For a given gray level image of size \( N \times N \), let the range pool \( R \) be the set of the \((N/L)^2\) non-overlapping blocks of size \( L \times L \) which is the size of encoding unit. Let the contractivity of the fractal coding be a fixed quantity of 2. Thus, the domain pool makes up the set of \((N-2L+1)^2\) overlapping blocks of size \((2L \times 2L)\). For the case of \(256 \times 256\) image with \(8 \times 8\) coding size, the range pool contains 1024 blocks of size \(8 \times 8\) and the domain pool contains 58081 blocks of size \(16 \times 16\). For each range block \(v\) in \(R\), one searches in the domain pool \(D\) to find the best match, i.e., the most similar domain block. The parameters describing this fractal affine transformation form the fractal compression code of \(v\).

At each search entry, the domain block is first down-sampled to \(8 \times 8\) and denoted by \(u\). Let the set of down-sampled domain blocks be denoted by \(D\). The down-sampled block is transformed subject to the eight transformations in the Dihedral on the pixel positions. If the origin of \(u\) is assumed to locate at the center of the block, the eight transformations \(T_k; k = 0, \ldots , 7\) can be represented by the following matrices:

\[
T_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, T_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, T_2 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, T_3 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix},
\]

\[
T_4 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, T_5 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, T_6 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, T_7 = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}.
\]

(1)

The eight transformed blocks are denoted by \(u_k, k = 0,1, \ldots , 7\), where \(u_0 = u\). Fig. 1 is the diagram of eight Dihedral transformations. The transformations \(T_1\) and \(T_2\) correspond to the flips of \(u\) along the horizontal and vertical lines, respectively. \(T_3\) is the flip along both the horizontal and vertical lines. \(T_4, T_5, T_6,\) and \(T_7\) are the transformations of \(T_0, T_1, T_2,\) and \(T_3\) performed by an additional flip along the main diagonal line, respectively.

In fractal coding, it is also allowed a contrast scaling \(p\) and a brightness offset \(q\) on the transformed blocks. Thus, the fractal affine transformation \(\Phi\) of \(u(x, y)\) in \(D\) can be expressed as...
In each search entry, there are eight separate MSE computations required to find the index \(d\) such that

\[
d = \arg \min \{ \text{MSE}(p_d, q), \ t = 0, 1, \ldots, 7\},
\]

where

\[
\text{MSE}(u, v) = \frac{1}{L} \sum_{i,j=0}^{L-1} (u(i,j) - v(i,j))^2.
\]

Here, \(p_k\) and \(q_k\) can be computed directly as

\[
p_k = \frac{L^2 \langle u_k, v \rangle - \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} u_k(i,j) \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} v(i,j)}}{L^2 \langle u_k, u_k \rangle - \left( \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} u_k(i,j) \right)^2},
\]

\[
q_k = \frac{1}{L^2} \left[ \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} v(i,j) - p_k \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} u_k(i,j) \right].
\]

As \(u\) runs over all of the 58081 domain blocks in \(D\) to find the best match, the terms \(t_x\) and \(t_y\) in (2) can be obtained. Together with \(d\) and the specific \(p\) and \(q\) corresponding to this \(d\), the affine transformation (2) is found for the given range block \(v\). In practice, \(t_x, t_y, d, p,\) and \(q\) can be encoded using 8, 8, 3, 5, and 7 bits, respectively, which are regarded as the compression code of \(v\). Finally, as \(v\) runs over all of the 1024 range blocks in \(R\), the encoding process is completed.

To decode, one first makes up the 1024 affine transformations from the compression codes and chooses any image as the initial one. Then, one performs the 1024 affine transforms on the image to obtain a new image, and proceeds recursively. According to Partitioned Iteration Function Theorem (PIFS), the sequence of images will converge. The stopping criterion of the recursion is designed according to user’s application. The final image is the retrieved image of fractal coding.

3. Particle swarm optimization

PSO is a population-based algorithm for searching global optimum. The original idea of PSO is to simulate a simplified social behavior. It ties to artificial life, like bird flocking or fish schooling, and has some common features of evolutionary computation such as fitness evaluation. For example, PSO is like a GA in that the population is initialized with random solutions. The adjustment toward the best individual experience (PBEST) and the best social experience (GBEST) is conceptually similar to the crossover operation of the GA. However, it is unlike a GA in that each potential solution, called particle, is “flying” through hyperspace with a velocity. Moreover, the particles and the swarm have memory, which does not exist in the population of the GA [12,13].

Let \(x_{i,d}(t)\) and \(v_{i,d}(t)\) denote the \(d\)th dimensional value of the vector of position and velocity of \(i\)th particle in the swarm, respectively, at time \(t\). The PSO model can be expressed as

\[
v_{i,d}(t) = v_{i,d}(t-1) + c_1 \cdot \phi_1 \cdot (x_{i,d} - x_{i,d}(t-1)) + c_2 \cdot \phi_2 \cdot (x_{GBEST}^d - x_{i,d}(t-1)),
\]

\[
x_{i,d}(t) = x_{i,d}(t-1) + v_{i,d}(t),
\]

where \(x_{i}^d\) (PBEST) denotes the best position of \(i\)th particle up to time \(t-1\) and \(x_{GBEST}^d\) (GBEST) denotes the best position of the whole swarm up to time \(t-1\), \(\phi_1\) and \(\phi_2\) are random numbers, and \(c_1\) and \(c_2\) represent the individuality and sociability coefficients, respectively.

The population size is first determined, and the position and velocity of each particle are initialized. Each particle moves according to (7) and (8), and the fitness is then calculated. Meanwhile, the best positions of each particle and the swarm are recorded. Finally, as the stopping criterion is satisfied, the best position of the swarm is the final solution. The block diagram of PSO is displayed in Fig. 2 and the main steps are given as follows:

1. Set the swarm size. Initialize the position and the velocity of each particle randomly.
2. For each \(i\), evaluate the fitness value of \(x_i\) and update the individual best position \(x_i^*\) if better fitness is found.
3. Find the new best position of the whole swarm. Update the swarm best position \(x_{GBEST}^d\) if the fitness of the new best position is better than that of the previous swarm.
4. If the stopping criterion is satisfied, then stop.
5. For each particle, update the velocity and the position according (7) and (8). Go to step 2.

For FIC, PSO is used to search the near-best matches so as to speedup the encoder. Since the evaluated value of the sub-optimum is close to that of the best match, the quality of the retrieved image can be preserved.
As discussed in Section 2, the parameters \( t_x, t_y, d, p, \) and \( q \) constitute the fractal code. In the proposed method, we encode the particle as \((t_x, t_y)\), which is the position of the domain block. The quantities \( p \) and \( q \) can be calculated from (5) and (6), and \( k \) is searched separately. At each search entry, all of the eight Dihedral transformations in (1) are performed and the best index \( d \) in (3) can be obtained. The fitness value of a particle is defined as the minus of the minimal MSE produced from eight Dihedral transformations, i.e., \(-\text{MSE}(p_d u_d + q_d, v)\). When the stopping criterion is satisfied, the final \( x = (t_x^*, t_y^*) \) together with the corresponding \( d, p, \) and \( q \) is the fractal code of the given range block \( v \). The steps of encoding a range block using PSO are summarized as follows:

1. Initialize the parameters of PSO.
2. For each particle \((t_x, t_y)\), fetch the domain block at \((t_x, t_y)\) in the image. Sub-sample the block and denote it by \( u \).
3. For each Dihedral transformation, compute \( u_k, k = 0, 1, \ldots, 7 \). Calculate the contrast scaling \( p_k \) and brightness offset \( q_k \). Find the fitness of the particle corresponding to the best parameter \( d \) as given in (3).
4. Update the PBEST and the GBEST if required. The corresponding fractal codes are also updated accordingly.
5. If stopping criterion is satisfied, then stop.
6. For each particle, update the velocity and the position according (7) and (8). Go to step 2.

FIC using full search strategy can exactly find the best domain block corresponding to each range block, but it is time-consuming. PSO can provide a faster way to encode the range blocks. Since the swarm can only find the sub-optimal domain block, there will be quality decay.

4. Particle swarm optimization with \( q \)-restriction

One of the main reasons for FIC being time-consuming is the MSE computations of the eight orientations induced from the Dihedral transformations in (1). To save the computations, one will directly choose the suitable ones out of the eight orientations according to edge property.

In the minimization of (3), not all of the eight transformations are required to test due to the edge property of blocks. For example, if both the domain block \( u \) and the range block \( v \) have horizontal edges, the transformation \( T_d \) will change the domain block to a block with vertical edges. In this case, \( T_d u \) and \( v \) tend to have large MSE, and therefore \( T_k, k = 4, 5, 6, 7 \) are not good candidates for the minimization. Therefore, one will confine the search only on \( T_d u \) for \( k = 0, 1, 2, 3 \). Such a mechanism is referred to as PSO with \( q \)-restriction.

The edge property is specified through an edge-type classifier. The classifier partitions image blocks into five classes which are smooth class, horizontal/vertical edge classes, and diagonal/sub-diagonal edge classes. In terms of fractal similarity, the underlying idea of such a classifier is that blocks tend to be similar to blocks of the same type.

The classifier is implemented through two DCT coefficients which represent the strength of the horizontal and vertical energy in the block. Let \( f \) be a given image block of size \( L \times L \). The DCT of \( f \), denoted by \( F \), is computed from the formula

\[
F(m,n) = \frac{2}{L} C_m C_n \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} f(i,j) \cos \left( \frac{(2i+1)m\pi}{2L} \right) \cos \left( \frac{(2j+1)n\pi}{2L} \right),
\]

where \( m, n = 0, 1, \ldots, L - 1 \), and

\[
C_k = \begin{cases} 
1/\sqrt{2}, & \text{if } k = 0, \\
1, & \text{else}.
\end{cases}
\]

Assume \( L = 8 \), one can compute \( F(1,0) \) from (9) as

\[
F(1,0) = \frac{\sqrt{2}}{8} \sum_{j=0}^{7} \sum_{i=0}^{7} f(i,j) \cos \left( \frac{(2i+1)\pi}{16} \right).
\]

The term \(|F(1,0)|\) measures energy variation between the left half and the right half of \( f \). Therefore, if \( f \) has strong vertical edges, \(|F(1,0)|\) will be significant. Similarly, \(|F(0,1)|\) reflects the strength of horizontal edges.

The classification is performed according to the magnitudes of \( F(1,0) \) and \( F(0,1) \). Let \( T_S \) be the threshold to determine the smooth class and \( T_D \) be the threshold to determine the diagonal/sub-diagonal class. The block diagram of the DCT-based classifier is given in Fig. 3. The input is an image block \( f \) and the output is the class type \( S \) (smooth), \( H_h \) (horizontal), \( H_v \) (vertical), \( D_d \) (diagonal),
or $D_s$ (sub-diagonal). The detailed steps of the classification scheme of $f$ are given as follows:

1. Compute $F(1,0)$ and $F(0,1)$ from (9).
2. IF $|F(0,1)| < T_S$ and $|F(1,0)| < T_S$, THEN
   Type = $S$.
ELSE IF $|F(0,1)| - |F(1,0)| > T_D$, THEN
   Type = $H_h$.
ELSE IF $|F(1,0)| - |F(0,1)| > T_D$, THEN
   Type = $H_v$.
ELSE IF $F(1,0) \times F(0,1) < 0$, THEN
   Type = $D_d$.
ELSE
   Type = $D_s$.

Since the image blocks can be assigned to one of the five classes, there are 25 combinations between domain and range blocks. Among these 25 combinations, nine of them will produce smaller MSE values subject to a subset of the Dihedral transformations. The set of nine combinations with potential candidates is called the superior clan. The possible Dihedral transformations for the nine superior clan pairs of domain and range blocks with the specific edge property are listed in Table 1. At the fourth row, the edge-type of the domain and range blocks are $H_h$ and $H_v$, respectively. In this case, only $T_k$ with $k = 4, 5, 6, 7$ are good candidates. Only these Dihedral transformations are performed.

The other 16 combinations tend to produce larger MSE values subject to the eight Dihedral transformations, since the edge-types do not match. This set of combinations is referred to as the inferior clan. For a search entry, if the combination is of this clan, no Dihedral transformation is required. In other words, this search entry is abandoned. The practical implementation of $k$-restriction method is summarized as follows. For a given range block, when the domain block fetched by a particle is of the types in Table 1, the listed transformations are performed. If the combination of the pair is not in Table 1, no transformation is needed and the process jumps to the next PSO epoch.

It is obvious that the mechanism of the $k$-restriction scheme is simple and there are only few computations required to calculate the lower DCT coefficients. The overhead of this classifier is very small compared to the fractal coding process. Moreover, since the classifier is designed according to the edge property, the image quality can be preserved.

5. Particle swarm optimization with $k$-restriction and intuitive move

The edge-type classification scheme using DCT coefficients mentioned in the previous section can provide an
intuitive guidance for particles in PSO. Originally, particles fly to the next position according to the sociality and individuality that are derived from the best experience of the swarm and individual, i.e., GBEST and PBEST, respectively. To make use of the edge property of natural images, one introduces a follow/jump strategy which guides particles to regions having promising candidates. This strategy, which handles the pairs in the inferior clan, is shown in Table 2. If a particle locates a domain block such that the pair is in the inferior clan, i.e., not in Table 1, a recursive re-positioning mechanism will be performed according to its edge property until the pair is in the superior clan. The recursive re-positioning mechanism is referred to as intuitive move.

The intuitive move strategy consists of three types of movements, which are summarized in Table 2. Type 1 is the “follow” case. As shown in Table 2, if both domain and range blocks are edged, the intuitive move of the particle will follow the edge direction of the domain block. The intrinsic idea is that the particle following the edge direction of corresponding domain block will have better chance to find another edged block which might be transferred to pose suitable edge properties, i.e., fall into the superior clan. Type 2, called “jump” case, occurs when encoding range block is of smooth type and certain particle finds an edged one. In this case, the particle tends to leave the edged region and jumps into the smooth region. If encoding range block is edged and the particle finds a smooth domain block, Type 3 of jump offers the particle a random move to find an edged one around the current position.

In the original PSO algorithm, the new positions of particles are updated according to (7) and (8). If the combination of domain and range blocks is in the inferior clan, the position-updating rule will be confined to some specific directions. The confined direction of the move \( \hat{v}_I \) is given in column 3 of Table 2, and the final move \( \hat{v}_I \) is determined by \( v_I = \gamma \cdot \hat{v}_I \), where \( \gamma \) is the sign of the quantity \( \{v_I, \hat{v}_I\} \). In “random” case shown in Table 2, \( \rho_1, \rho_2 \in \{-1,0,1\} \) are randomly selected. Although most particles can find another suitable domain block, sometimes, intuitive move will fail on cyclic path and cannot find a proper one. To solve this problem, we predefine a maximum number of rounds to terminate the intuitive move.

The cost of the proposed PSO with intuitive move is minor since the mechanism utilizes the same edge information constructed by edge-type classifier. Furthermore, both the methods of PSO with \( k \)-restriction and PSO with intuitive move can be merged to speedup fractal image compression. This merged method is called PSO-kI. The block diagram of Fig. 4 demonstrates the process of \( k \)-restriction and intuitive move schemes for a given range block \( v \) and a searching entry \( u \) which is represented by a particle. When \( v \) and \( u \) are of the superior clan, we check Table 1 to find the corresponding Dihedral transformation. Otherwise, we perform the intuitive move operations. The detailed steps of PSO-kI for encoding a given range block are described as follows:

1. Initialize the parameters of PSO.
2. When particles move to new regions, evaluate the edge-types of domain and range blocks.
3. If the relation of the edge-type pair is in the superior clan and has candidates of \( k \) in Table 1, go to step 6.
4. Else use intuitive move mechanism to guide the particle according to Table 2.
5. Repeat steps 3 and 4 until the edge-type pair falls in the superior clan or a predefined maximum number of intuitive move rounds has been reached.
6. Perform the corresponding transformations and evaluate the particle by finding the optimal fitness in all listed transformations.
7. Update the PBEST and the GBEST if required. The corresponding fractal codes are also updated accordingly.
8. If stopping criterion is satisfied, then stop.
9. For each particle, update the velocity and the position according (7) and (8). Go to step 2.

Since the features of \( k \)-restriction and intuitive move both utilize the edge-type information provided by edge-type classifier, the proposed PSO-kI method is merged perfectly. Moreover, the mechanism is embedded into the fractal coding scheme and induces little overhead, therefore the purpose of speedup can be achieved. Also, since visual information is highly adopted, the image quality can be preserved.

### 6. Experimental results

The proposed visual-based PSO for fractal coding using edge property is simulated and verified. Three types of PSO
are implemented, which are traditional PSO, PSO with 
$k$-restriction (PSO-k) and PSO with both 
$k$-restriction and intuitive move (PSO-kI). The tested images are Lena, Pepper, F16 and Baboon, each of which is of size $256 \times 256$. The coding size for fractal coder is $8 \times 8$. Simulation programs are implemented using Borland C++ Builder 5.0 running on Microsoft Windows XP, Pentium M 1-GHz platform.

The speedup ratio is defined as the ratio of the encoding time of the full search method over that of the method under consideration. The distortion between the original image $f(i,j)$ and the retrieved image $\hat{f}(i,j)$ caused by lossy compression is measured in Peak Signal to Noise Ratio (PSNR) defined by

$$\text{PSNR} = 10 \cdot \log_{10} \left( \frac{255^2}{\text{MSE}} \right),$$

where MSE is given in (4). In the edge-type classifier, the threshold values are set as $T_S = 29.2$ and $T_D = 52.8$. These thresholds and all PSO parameters used below are chosen by experiments.

Table 3 shows the performance of the PSO methods on Lena image compared to that of the full search method. The parameter settings, population size, and maximum number of epochs are selected such that the three PSO methods result in approximately the same image quality. Since the MSE computations in three versions of PSO are involved in different ways, this is a better view for the comparison. For the traditional PSO with population size 35 and a maximum number of 33 epochs for encoding each range block, the encoding time is 115.52 seconds, which is 42.8 times faster and the image quality PSNR is 28.08 dB. For PSO with $k$-restriction, the speedup ratio is 66.85 times faster with the same image quality. When intuitive move mechanism is also added, the speedup ratio of PSO-kI is increased up to 124.6 times faster, which is about three times faster than that of the traditional PSO. The last two rows of Table 3 show the number of MSE computations and the reduced ratios. As shown, the tendency of the reduced ratio is roughly coincident to the speedup ratio with some deviation. Since there are computation overheads introduced from the classifier and the follow/jump mechanisms, the speedup ratios do not catch the MSE reduced ratios.

For a fixed population size of 35, the image quality versus the encoding time is depicted in Fig. 5. The encoding time is controlled by the maximal number of epochs and some selected samples are dotted on the graph. As demonstrated, PSO-k and PSO-kI compared to PSO have faster speed to get acceptable image quality in shorter time. One feature of the PSO-kI method is, for PSNR being 28 dB, that this method outperforms PSO-k method in terms of the speedup ratio discussed previously. Another feature is that the image quality can be further improved.

Table 3

<table>
<thead>
<tr>
<th>Compression method</th>
<th>Full search</th>
<th>PSO</th>
<th>PSO-k</th>
<th>PSO-kI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Particle population size</td>
<td>–</td>
<td>35</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>No. of rounds</td>
<td>–</td>
<td>33</td>
<td>98</td>
<td>28</td>
</tr>
<tr>
<td>PSNR</td>
<td>28.91</td>
<td>28.08</td>
<td>28.00</td>
<td>28.02</td>
</tr>
<tr>
<td>Time (s)</td>
<td>4943.87</td>
<td>115.52</td>
<td>73.95</td>
<td>39.67</td>
</tr>
<tr>
<td>Speedup ratio</td>
<td>1.00</td>
<td>42.80</td>
<td>66.85</td>
<td>124.62</td>
</tr>
<tr>
<td>No. of MSE computations</td>
<td>475799552</td>
<td>9748760</td>
<td>5787110</td>
<td>2876440</td>
</tr>
<tr>
<td>Reduced ratio</td>
<td>1.00</td>
<td>48.81</td>
<td>82.22</td>
<td>165.41</td>
</tr>
</tbody>
</table>
with the expense of more encoding time. The $k$-restriction can speed up the search and the intuitive move overcomes the decay of retrieved image quality by providing the follow/jump guidance. Fig. 6 shows PSNR versus the number of MSE computations, which reveals the fact that encoding times coincide to the number of MSE computations. In comparison to the full search method, the speedup ratio of Fig. 6 is given in Fig. 7.

Fig. 8 shows the PSNR decay, which is compared with the full search method, versus encoding time of the four images using PSO-kI method. As the encoding time increases, the decays of PSNR decrease to an acceptable range. Such a phenomenon is independent of the type of images. The PSNR versus the speedup ratio is given in Fig. 9. As shown, when increasing the speedup ratio, qualities decrease slowly for each of the images.

Fig. 10 shows the decoded images under PSO-k and PSO-kI methods with the same speedup ratio of 125. When the encoding time is the same, intuitive move mechanism in PSO-kI method can prevent the decay of the image quality. PSO-kI method produces image quality of 28.02 dB, which is better than 27.77 dB produced by PSO-k method. Fig. 11 shows the portion of the magnified images. As observed,

7. Conclusion

In this paper, a visual-based PSO with $k$-restriction and intuitive move has been proposed for fractal image compression. The $k$-restriction is designed by using classification scheme based on edge-type classifier, which avoids
unnecessary MSE computations. The intuitive move utilizes the edge information constructed by edge-type classifier, which spends little effort to find promising domain blocks. Since the classifier is designed according to edge properties of image blocks, image quality can be preserved. Using this method, the encoding speed is improved up to 125 times faster with quality decay from 28.91 to 28.02 in dB.

References