CHAPTER 4

FAST FRACTAL IMAGE COMPRESSION
BASED ON DOMAIN-RANGE ENTROPY
4.1 INTRODUCTION

In this chapter, the proposed fast fractal image compression method, based on domain and range entropy is described. The proposed method is expected to reduce the encoding time significantly, while retaining the fidelity and compression ratio. The method is a two-step process. First, domains that are similar i.e. domains having nearly equal variances are eliminated from the domain pool. Second, during the encoding phase, only domains and ranges having equal entropies (with an adaptive error threshold for each quadtree depth) are compared for a match within the root mean square (rms) error tolerance. As a result, many unqualified domains are avoided from comparison and a significant reduction in encoding time is expected.

The results of experiments conducted on standard 8 bit gray scale images [60] of size 512x512 are presented. The results are compared with three other related methods: quadtree partition method with classified search (baseline method) [9, 23], variance method [11], and domain entropy method [18].

4.2 PROPOSED METHOD

4.2.1 Entropy

Assume a set of discrete set of symbols \{v_0, v_1, ..., v_k\} with associated probabilities \(p_i\). The entropy [39] of the discrete distribution denoted by \(H\), gives a measure of the randomness of a sequence of symbols drawn from it.
Entropy, $H$, is given in terms of $p_i$ by,

$$H = H(P) = -\sum p_i \log_2 p_i \quad \text{(4.1)}$$

In the context of coding a message, entropy represents the lower bound on the average number of bits per input value. The function $H$ has the following lower and upper limits.

$$0 = H(1,0,0,...,0) \leq H(p_0, p_1, ..., p_k) \leq H(\frac{1}{k}, \frac{1}{k}, ..., \frac{1}{k}) = \log_2 k \quad \text{(4.2)}$$

In image processing, the individual outcomes $p_0, p_1, ..., p_k$, represent the normalized frequency of occurrence of each gray level unit in the image. It gives the spatial information i.e. the distribution of gray levels in the image (image histogram). For an image block, the entropy is defined by,

$$H(p_0, p_1, ..., p_k) = H(P) = -\sum_{i=0}^{k} p_i \log_2 p_i \quad \text{(4.3)}$$

Where, $k$ denotes the maximum number of gray levels in the given image.

### 4.2.2 Domain-Range Entropy Algorithm

An affine transformation applied on the domain will not change the entropy [40]. Two image blocks (domain and range) having an affined relation will have similar entropies. Based on this property, the present fast fractal compression method is developed. The given image $I$ of size $n \times n$ (8 bit gray scale) is partitioned using quadtree partition algorithm into overlapped domain blocks $D_i$ of size $2n \times 2n$. During the encoding phase, the image is partitioned into non-overlapped range blocks $R_i$ of size $n \times n$. The absolute difference of entropy of the range block $R_i$ and domain block $D_i$ is compared using the relation.

$$| \text{Entropy} (R_i) - \text{Entropy} (D_i) | \leq \lambda_{\text{depth}} \quad \text{(4.4)}$$

where $\lambda_{\text{depth}}$ is the error tolerance for the domain-range entropy difference.
Domain blocks $D_i$ and range blocks $R_i$ satisfying the above condition, are subjected to linear regression analysis, i.e., comparing with the rms error tolerance $e_2$ for a best match of the domain and range blocks.

**Adaptive error threshold for entropy matching**

An adaptive error threshold $\lambda_{\text{depth}}$ is formulated in this proposed work, for each quadtree partition to match the entropy of the domain and range blocks. For quadtree depth corresponding to $\text{min}_\text{part}$, $\lambda_0$ is chosen as a small initial value (between 0 and 1). For other quadtree partitions, $\lambda$ is given by,

$$\lambda_{\text{depth}} = \lambda_{\text{depth-1}} + (\text{depth}-1)$$

(4.5)

The above formula is obtained by conducting repeated experiments on images of varying sizes and textures, testing for the optimal encoding time, quality and compression ratio. It is observed that for ranges corresponding to the maximum quadtree depth (smallest size ranges), more domains must be compared, while at minimum depth less number of domains are sufficient for a good match.

**4.2.3 Domain Pool Classification**

The given image $I$ (of size $h_{\text{size}} \times v_{\text{size}}$) is read from the input file and stored into an array ‘image’. The minimum partitions ($\text{min}_\text{part}$) and the maximum partitions ($\text{max}_\text{part}$), the rms error tolerance ($e_2$), are also read as input. Then, the number of quadtree partitions ($\text{depth} = \text{max}_\text{part} - \text{min}_\text{part}$), the block size and total number of domains, and ranges for each quadtree partition are computed. In this work, three quadtree partitions are used. The range and domain block sizes corresponding to these three partitions are $16 \times 16$, $8 \times 8$, and $4 \times 4$, and $32 \times 32$, $16 \times 16$, and $8 \times 8$, respectively. Initially, the domain blocks of size $2r \times 2r$ ($32 \times 32$), corresponding to the minimum quadtree partition depth ($\text{depth} = 0$), are considered. The domain pool $D$, corresponding to
this size, is constructed by sliding a window of size $2\times 2r$ across the given image. Next, the domains are classified. Each domain block is split into four quadrants. The average contrast $A_i$, and variance $V_i$ of pixels in the four quadrants is computed by,

$$A_i = \frac{1}{n} \sum_{j=1}^{n} d_{ij}$$

and

$$V_i = \frac{1}{n} \sum_{j=1}^{n} (d_{ij} - A_i)^2$$  \hspace{1cm} (4.6)

where, $d_{ij}$ refers to the pixel $j$ ($i = 1, ..., n$) in the quadrant $i$ ($i = 0, 1, 2, 3$).

Figure 4.1 Classification (major class) of domain/range block into 3 classes based on the canonical ordering of the average contrast of pixels in the four quadrants.

Figure 4.2 Classification (sub class) of domain/range block into 24 classes (1-4) based on the variance of the pixels in the four quadrants.
Each domain is classified into one of the four major classes (Figure 4.1) based on the canonical ordering of the average contrast of the pixels in the quadrants. The domain is further classified into one of the 24 (14) sub classes (Figure 4.2) based on the variance of the pixels in the four quadrants. The domain is contracted to the size of the range block (by averaging 2x2 pixels) and stored in separate arrays. This will avoid repeated averaging of the domain blocks during the encoding phase. The entropy value of each domain block is also computed using equation (2.3). All the domains of size 32x32 are classified using this procedure.

The above process is repeated for all quadtree partitions, i.e. the domain pools D of other sizes (16x16, 8x8) are extracted and classified. The domain pools are sorted in ascending order of the variance and placed in separate list structures (a total of depth x 4 x 24 sorted lists). Next, the domains having similar variance are removed from the domain pool, (within an error threshold of 1.0) only one of those is placed on the list. This reduces the size of the initial domain pool.

4.2.4 Encoding Procedure

The encoding procedure consists of the following steps:

Step 1: Initialize the output file by writing the header information:

min_part, max_part, domain_step, hsize, vszie

Step 2: Initialize depth = 0, best_rms = infinity. The given image is quadtree-partitioned recursively until depth is equal to minimum partitions (min_part). The value of depth is incremented by one for each quadtree partition.

Step 3: Select a range block Ri (of size rxr) in the current image partition.

Step 4: Classify the range block based on mean and variance (sec.4.2.1) and compute its entropy.
Step 5: Select a domain from the classified domain lists corresponding to the size (2rx2r) and same class as that of the range.

Step 6: Compute the absolute difference of the domain and range entropies. If the difference is less than or equal to the adaptive threshold, \( \lambda_{\text{depth}} \), go to step 7, otherwise, go to step 5.

Step 7: Compute the \( \text{rms\_error} \) (section 3.3.4) for the current domain, range pair. If the \( \text{rms\_error} \leq \text{best\_rms} \), set this as the best\_rms and repeat steps 5 to 7 for all the domains in the current list.

Step 8: If the \( \text{best\_rms} > e_c \) and \( \text{depth} < \text{max\_part} \), partition the range block and repeat steps 5 to 8. Otherwise, mark the range \( R_i \) as covered.

Step 9: Write into output file, the transformation \( w_i \) (fractal code), comprising of the domain position, (x,y coordinate values), the symmetry operation (sym_op), the contrast scaling (s), and the luminance offset (o). This constitutes the fractal code for the given range.

*Repeat steps 3 to 9 for all the quadtree partitions.*

**Computing s and o**

In practice, a domain and range are compared using the \( \text{rms\_error} \) metric (sec.). Using this metric also allows easy computation of optimal values for \( s \) and \( o \) in Equation (3.1).

Given two square images containing \( n \) pixel intensities, \( d_1...d_n \) (from \( D_i \)) and \( r_1...r_n \) (from \( R_i \)), one can seek \( s \) and \( o \) to minimize the quantity \( E \), given by,

\[
E(R_i, D_i) = \sum_{i=1}^{n} (s \cdot d_i + o - r_i)^2
\]  \hspace{1cm} (4.7)

This gives the settings for contrast scaling \( s \) and brightness \( o \) that make the affinely transformed \( d_i \) values to have the least squared distance from the \( r_i \) values. The
minimum value of E occurs when the partial derivatives with respect to s and o are zero, which occurs when,

\[
\begin{align*}
    s &= \frac{n}{n} \left( \sum_{i=1}^{n} d_i r_i - \frac{1}{n} \sum_{i=1}^{n} \sum_{i=1}^{n} r_i \right) \\
    o &= \frac{1}{n} \left( \sum_{i=1}^{n} r_i - s \sum_{i=1}^{n} d_i \right)
\end{align*}
\]  (4.8)

Thus, \( E(R_i, D_j) \) is computed by,

\[
E(R_i, D_j) = \frac{1}{n} \left[ \sum_{i=1}^{n} r_i^2 + s \left( \sum_{i=1}^{n} d_i^2 - 2 \sum_{i=1}^{n} d_i r_i + 2 o \sum_{i=1}^{n} d_i \right) + o \left( n o - 2 \sum_{i=1}^{n} r_i \right) \right]
\]  (4.10)

If \( n \sum_{i=1}^{n} d_i^2 - \left( \sum_{i=1}^{n} d_i \right)^2 = 0 \), then, \( s = 0 \) and \( o = \frac{1}{n} \sum_{i=1}^{n} r_i \)

The root mean square error, \( \text{erm}_s = \sqrt{E(R_i, D_j)} \)  (4.11)

4.2.5 Encoding Algorithm

Step 1: Construct the domain pools \( D_{\text{depth}} \) corresponding to each quadtree partition depth starting from minimum partitions to maximum partitions (\( \text{depth}=0 \) to \( \text{max}_\text{part-min}_\text{part} \)).

Step 2: Calculate the entropy of the domain blocks in each pool \( D_{\text{depth}} \).

Step 3: Classify and sort the domains in each pool \( D_{\text{depth}} \) in ascending order of entropy and place on a list structure.
Step 4: Remove similar domains (having nearly equal variance) from the list.

Step 5: Search for a best match between a range and domain block of same class using the following routine.

Write_header_info; (min_part, max_part, domain_step, hsize, vszie)

depth=0; c = rms_tol;

Function Quadtree (image, depth) {
  best_rms=infinity; λ₀=initial value; λ_depth= λ_{depth-1}+(depth-1);
  While (depth<min_part) Quadtree (image, depth+1);
  set R₁ = I² and mark it uncovered.
  While there are uncovered ranges Rᵢ do {
    //Select the domain pool list D_{depth} corresponding to the current range block Rᵢ.
    for (j=1;j<num_domains; ++j) {
      If | Entropy (Rᵢ) – Entropy (Dⱼ) | ≤ λ_depth, {
        Compute s, o, and sym_op;
        Compute E (Rᵢ, Dⱼ);
        If E (Rᵢ, Dⱼ) ≤ best_rms {
          best_rms= E(Rᵢ, Dⱼ);
          best_domain=(domain_x, domain_y);
        }
      } // end for
      If (best_rms<ε_c) && (depth<max_part) Quadtree (image, depth+1);
    } // end while
    Else write_transformations (best_domain, s, o, sym_op);
  } //end while
} // end function Quadtree( )
Input parameters for the algorithm:

- The given image I and size of image (hsize, vsize)
- The rms tolerance threshold $e_c$
- The initial value of adaptive entropy threshold $\lambda_0$
- The maximum depth of the quadtree partition ($\text{max\_part}$) and minimum depth of the quadtree partition ($\text{min\_part}$) (depends on the size of input image)
- The domain skip distance (pixel distance between adjacent domains) $\delta_h$ and $\delta_v$
- The number of bits allocated for quantizing, $s$, and $o$
- The type of search performed for a domain vs. range match
  (Full search=72 classes, classified search= single class).

An encoding of an image consists of the following data:

- The domain address ($\text{domain\_x}$, $\text{domain\_y}$)
- The scaling and offset values for each range ($s_i$ and $o_i$).
- The orientation and flip information ($\text{sym\_op}$)

**Domain address:** The domains $D_i$ must be referenced by position and size. The domains are indexed and referenced by this index. However, when the scaling value is zero, the domain is irrelevant, and so no domain or orientation information is stored.

**Orientation:** There are eight ways to map the four corners of $D_i$ to the corners of $R_i$. Three bits are used to determine this rotation and flip ($\text{sym\_op}$) information.

**Quadtree:** One bit is used at each quadtree level to denote a further recursion or ensuing transformation information. At the maximum depth, however, no such bit is used, since the decoder knows that no further division is possible.
4.3 DECODING THE ENCODED IMAGE

Decoding an encoded image consists of iterating \( W \) from any initial image. The quadtree partition is used to determine all the ranges in the image. The domain \( D_i \) mapping on to each range \( R_i \) is spatially contracted by averaging 2x2 pixels. The contracted domain pixel values are then multiplied by \( s_i \), added to \( o_j \), and placed in the location of the range determined by the orientation information. This constitutes a decoding iteration. The decoding step is iterated, until the fixed point is approximated i.e., until further iteration doesn't change the image or until the change is below a small threshold value.

4.3.1 Decoding procedure

Step1: Start by reading the compressed image file into input_file.

Step2: The header information (min_part, max_part, domain_step, hsize, vsize), is read from the input_file. The number of iterations (num_iterations) for decoding is read as input.

Step 3: Initialize arrays image, and temp_image of size hsize, vsize of type unsigned char.

Step 4: If any other initial image (hypothetical image) is given, we read it into the array image.

Step 5: Read the transformations from the input file.

Step 6: Initialize depth=0. Recursively partition the array image up to minimum partition level (min_part); depth is incremented for each quadtree partition.

Step 7: Read a transformation \( w_i \) from the input_file. A flag bit '1' in the transformation \( w_i \) indicates further partition. If depth is less than max_part, then partition the image.
Step 8: Read the transformation \( w_i \), \((s, o, domain_x, domain_y, sym_op)\) from input file. Compute the range size and position information, corresponding to the domain address. Store \( w_i \) and range information in the structure \( node \). The node is added to the list \( trans \). Repeat steps 7 and 8 for all partitions of the image. Finally, the list \( trans \) contains the transformations for decoding the compressed image.

Step 9: Apply transformations. The first transformation is read from the list \( trans \). The domain is contracted to the size of the range, by averaging 2x2 pixels from the array \( image \). These pixels are stored in the array \( temp_image \). Next, the rotation and flip information is applied to the pixels in \( temp_image \) with \( sym_op \). Finally, the scaling and offset transformations are applied on the pixels in \( temp_image \).

10. Repeat the step 9 by applying all the transformations in the list \( trans \) on the range and domain blocks. Transfer \( temp_image \) to \( image \). This constitutes a decoding iteration.

Step 11: Repeat step 9 and 10 until \( num_iterations \). Write the decoded image into the output_file.

4.3.2 Decoding Algorithm

Read the compressed image into \( input_file \);

Read the number of iterations for decoding, \( num_iterations \);

Read the header_info, \((min_part, max_part, domain_step, hsize, vszie)\);

Initialize arrays \( image, temp_image \) \((hsize, vszie)\), type unsigned char;

Create list, \( trans_node \) for saving transformation parameters;

Initialize the list \( 'trans' \) of type \( trans_node \); Set, \( depth=0 \);

Function Quadtree \((image, depth)\) {
    While \((depth<min_part)\) Quadtree \((image, depth+1)\);
    Read \( w_i \) from the input_file;
If (flag bit ==1 & depth<max_part) Quadtree (image, depth+1);
  Read wi; Compute range size and position;
  Enter wi and range_info into trans_node;
  Add trans_node to the list 'trans';
  While (there are more ranges) do {
    Read wi; Compute range size and position;
    Enter wi and range_info into trans_node;
    Add trans_node to the list 'trans'; }

  //Perform transformations on domains to decode the image
  for(i=0;i<num_iterations;++i){
    while (j=0;j<num_trans;++j) {
      temp_image(Domainj)=image(contracted(Domainj));
      \(image\)(Domainj)=apply_transform(temp_image(Domainj));
    } // end while
    \(image\)=temp_image;
  } // end for
  Write ('image', output_file);
} // End function Quadtree()

4.3.3 Decoding at Larger Scale

The advantage of fractal image compression is its multi-resolution property, i.e. an image can be decoded at higher or lower resolutions than the original, and it is possible to zoom sections of the image. These properties made it a very attractive method for applications in image processing. The compressed image can be decoded at higher scales, by reading an optional input parameter \textit{scale}. 
4.3.4 Post Processing

Since the ranges are encoded independently the pixel values may have discontinuities at block boundaries. These artifacts are minimized by post processing the image. In this implementation, pixels at the boundary of the range blocks are modified, using a weighted average of their values. If the pixel values on either size of a boundary are \( p \) and \( q \), then these are replaced by \( \omega_1 p + \omega_2 q \) and \( \omega_2 p + \omega_1 q \) with \( \omega_1 + \omega_2 = 1 \). Ranges that occur at the maximum depth of the quadtree are averaged with weights of \( \omega_1 = 5/6 \) and \( \omega_2 = 1/6 \), and those above this depth are averaged with weights of \( \omega_1 = 2/3 \) and \( \omega_2 = 1/3 \). These values are largely heuristic, but the results are satisfactory.

4.3.5 Fidelity Measure

In practice, the Peak signal-to-noise ratio (PSNR) [45] is used to measure the quality of the decoded image. It is a measure of the difference between two images, given by,

\[
PSNR = 20 \log_{10} \left( \frac{I_{\text{max}}}{\text{rms}} \right)
\]

(4.12)

Where, \( I_{\text{max}} \) is the maximum gray level intensity of the pixel (255 for an 8 bit image), and \( \text{rms} \) is the root mean square difference between two images. The PSNR is given in decibel units (dB), which is a measure of the ratio of the peak signal and the difference between two images.

4.3.6 Compression Ratio and Bit Rate

The compression ratio [1], \( C_R \) for an image is defined by,

\[
C_R = \frac{\text{size of the original image}}{\text{size of the encoded image}}
\]

The bit rate is defined [45] as the number of bits per pixel (bpp). Low bit rates indicate high compression, and vice versa. Bit rate for 8 bit gray scale images is given by,

\[
B_r = 8 \cdot \frac{1}{C_R}
\]

(4.13)
4.4 EXPERIMENTATION

Experiments are conducted on standard 8 bit gray scale images (raw format) of size 512x512 [60], using the proposed algorithm. The results are tabulated and compared with other related methods discussed in the literature - classified search method [9, 23], variance method [11], and domain entropy method [18]. The decoded images (at high compression ratio) in each of the above methods are presented for visual inspection. The encoded images with the above methods (at high compression ratio) are also compared with the images encoded in JPEG standard. The difference of original image and image decoded (at high compression ratio) by the proposed method and JPEG encoding is presented.

4.4.1 Encoding Parameters

The following values are assigned for various parameters in the encoding and decoding algorithms (common to all images).

- 5 bits are used to quantize the scaling coefficient $s$, and 7 bits for the offset, $o$.
- In all the images, three quadtree partitions are used. The biggest range size is 16x16 (minimum quadtree depth 5) and smallest range size is 4x4 (maximum quadtree depth 7)
- Domain skip distance of 4.2 and 1 is used ($\delta_h=\delta_v=4$)
- The rms tolerance value $c_r$ is given values of 1.2.4.6.8.10.15, and 20, leading to results ranging from low to high compression.
- The algorithm is implemented in C language, using VC++6.0 compiler. Execution is carried out on a Personal Computer with Intel Centrino Duo T2250 processor. @1.73 GHz, with 1.0 GB of RAM, 100 GB Hard disk.
Image Size = 512x512 (8 bit gray scale, .raw format)

Number of quadtree partitions = 3

Total Number of Domains:

(i) Domain Skip Distance, $\delta_1=\delta_3=4$

Domain Size 32x32 = $(512-32)/4+1) \times (512-32)/4+1) = 14,641$

Domain Size 16x16 = $(512-16)/4+1) \times (512-16)/4+1) = 15,625$

Domain Size 8x8 = $(512-8)/4+1) \times (512-8)/4+1) = 16,129$

(ii) Domain Skip Distance, $\delta_2=\delta_4=2$

Domain Size 32x32 = $(512-32)/2+1) \times (512-32)/2+1) = 58,081$

Domain Size 16x16 = $(512-16)/2+1) \times (512-16)/2+1) = 62,001$

Domain Size 8x8 = $(512-8)/2+1) \times (512-8)/2+1) = 64,009$

(iii) Domain Skip Distance, $\delta_5=\delta_7=1$

Domain Size 32x32 = $(512-32)/1+1) \times (512-32)/1+1) = 2,313,61$

Domain Size 16x16 = $(512-16)/1+1) \times (512-16)/1+1) = 2,47,009$

Domain Size 8x8 = $(512-8)/1+1) \times (512-8)/1+1) = 2,55,025$

For the proposed method, the adaptive tolerance parameter (for domain and range entropy matching) is assigned initial value for quadtree depth 0 as $\lambda_0 = 0.7$. For other quadtree depths, $\lambda_{\text{depth}} = \lambda_{\text{depth}-1} + (\text{depth}-1)$.
4.5 RESULTS AND DISCUSSION

Figure 4.3 (a) and (b) show the original image of Lenna (512x512, 8 bit, raw format) and corresponding quadtree partition of image Lenna (by classified search method, CR=19.13, PSNR=32.44 dB, domain skip distance=4).

Figure 4.4 (a) shows the image of Lenna decoded by classified search algorithm (PSNR=28.95 dB, CR=41.91). Figure 4.4 (b) shows the image of Lenna decoded by variance method (PSNR=28.91 dB, CR=40.03). Figure 4.5 (a) shows the image of Lenna decoded by domain entropy method (PSNR=28.29 dB, CR=22.78). Figure 4.5 (b) shows the image of Lenna decoded by the proposed domain-range entropy method (PSNR=28.81 dB, CR=41.02).

Figure 4.6 (a) & (b) show the comparison of the decoded images of Lenna by proposed method (PSNR=28.81 dB, CR=41.02) and image decoded by standard JPEG (PSNR=30.41 dB, CR=40.0).

Figure 4.7 (a) shows the difference of original image Lenna and decoded image Lenna by the proposed domain-range entropy method. Figure 4.7 (b) shows the difference image between the original image Lenna and the decoded JPEG image.

The compression ratio, time and PSNR values obtained for image Lenna by the classified search method, variance method, domain entropy method, and domain-range entropy (proposed) method, with domain skip distance set at $\delta_h=\delta_v=4$ are given in Table-4.1. The compression ratios varied from 4.36 (classified search method) to 41.91 (classified search method). PSNR varied from 36.09 dB (classified search method) to 28.29 dB (domain entropy method). The encoding time varied from 8.76 (classified search method) seconds to 1.53 (variance method) seconds.
Table 4.2 gives the compression ratio, time and PSNR values obtained for image Lenna by the domain-range entropy (proposed) method and other methods, with domain skip distance $\delta_h=\delta_v=2$. The compression ratios varied from 4.09 (classified search method) to 42.70 (classified search method). PSNR varied from 37.04 dB (classified search method) to 28.32 dB (domain entropy method). The encoding time varied from 35.79 (classified search method) seconds to 5.72 (variance method) seconds.

Table 4.3 gives the compression ratio, time and PSNR values obtained for image Lenna by the domain-range entropy (proposed) method and other methods, with domain skip distance $\delta_h=\delta_v=1$. The compression ratios varied from 3.84 (classified search method) to 41.37 (classified search method). PSNR varied from 37.76 dB (classified search method) to 28.43 dB (domain entropy method). The encoding time varied from 152.75 seconds (classified search method) to 22.23 seconds (variance method).
Figure 4.3 (a) Original Image of Lenna (512x512, 8 bit gray scale, raw format) (b) Quadtree Partition of Image Lenna
Figure 4.4 Decoded Images of Lena

(a) Classified Search Method
(PSNR=28.95, CR=41.91)

(b) Variance Method
(PSNR=28.91, CR=40.03)
Figure 4.5 Decoded Images of Lenna

(a) Domain Entropy Method
     (PSNR=28.29, CR=22.78)

(b) Domain-Range Entropy (Proposed) Method
     (PSNR=28.81, CR=41.02)
Figure 4.6 Comparison of Proposed method and JPEG

(a) Decoded image encoded by Proposed Method
(PSNR=28.81, CR=41.02)

(b) Decoded Image encoded in standard JPEG
(PSNR=30.41, CR=40.00)
Figure 4.7 Difference Images
Table 4.1 Compression Ratio (CR), time and PSNR for Image Lenna by different methods (domain skip distance=4)

<table>
<thead>
<tr>
<th>Tolerance e_c</th>
<th>Classified Search Method</th>
<th>Variance Method</th>
<th>Domain Entropy Method</th>
<th>Domain-Range Entropy (Proposed) Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CR (Sec)</td>
<td>Time (Sec)</td>
<td>PSNR (dB)</td>
<td>CR (Sec)</td>
</tr>
<tr>
<td>1.0</td>
<td>4.36</td>
<td>8.76</td>
<td>36.09</td>
<td>4.36</td>
</tr>
<tr>
<td>2.0</td>
<td>4.85</td>
<td>8.18</td>
<td>36.04</td>
<td>4.84</td>
</tr>
<tr>
<td>4.0</td>
<td>8.67</td>
<td>5.53</td>
<td>35.34</td>
<td>8.64</td>
</tr>
<tr>
<td>6.0</td>
<td>11.96</td>
<td>4.46</td>
<td>34.43</td>
<td>11.94</td>
</tr>
<tr>
<td>8.0</td>
<td>15.46</td>
<td>3.84</td>
<td>33.49</td>
<td>15.25</td>
</tr>
<tr>
<td>10.0</td>
<td>19.13</td>
<td>3.35</td>
<td>32.44</td>
<td>18.69</td>
</tr>
<tr>
<td>15.0</td>
<td>29.23</td>
<td>2.70</td>
<td>30.43</td>
<td>27.93</td>
</tr>
<tr>
<td>20.0</td>
<td>41.91</td>
<td>2.31</td>
<td>28.95</td>
<td>40.03</td>
</tr>
</tbody>
</table>
Table 4.2 Compression Ratio (CR), time and PSNR for Image Lena by different methods (Domain Skip Distance=2)

<table>
<thead>
<tr>
<th>Classified Search Method</th>
<th>Variance Method</th>
<th>Domain Entropy Method</th>
<th>Domain-Range Entropy (Proposed) Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tolerance ( e_c )</td>
<td>CR (Sec) PSNR (dB)</td>
<td>CR (Sec) PSNR (dB)</td>
<td>CR (Sec) PSNR (dB)</td>
</tr>
<tr>
<td>1.0</td>
<td>4.09 35.79 37.04</td>
<td>4.09 22.64 36.62</td>
<td>4.09 24.23 37.04</td>
</tr>
<tr>
<td>2.0</td>
<td>4.62 32.98 36.96</td>
<td>4.62 20.29 36.55</td>
<td>4.49 22.37 36.97</td>
</tr>
<tr>
<td>4.0</td>
<td>8.42 21.78 36.02</td>
<td>8.39 12.14 35.14</td>
<td>7.04 16.04 36.10</td>
</tr>
<tr>
<td>6.0</td>
<td>11.87 17.32 34.87</td>
<td>11.82 9.50 34.68</td>
<td>8.87 13.53 35.01</td>
</tr>
<tr>
<td>8.0</td>
<td>15.50 14.70 33.82</td>
<td>15.31 8.21 33.72</td>
<td>10.60 11.97 33.87</td>
</tr>
<tr>
<td>10.0</td>
<td>19.42 12.82 32.67</td>
<td>18.96 7.34 32.67</td>
<td>12.31 10.75 32.80</td>
</tr>
<tr>
<td>15.0</td>
<td>30.25 10.10 30.54</td>
<td>29.40 6.28 30.67</td>
<td>16.25 8.92 30.55</td>
</tr>
<tr>
<td>20.0</td>
<td>42.70 8.70 29.01</td>
<td>40.96 5.72 29.20</td>
<td>21.55 7.45 28.32</td>
</tr>
</tbody>
</table>
Table 4.3 Compression Ratio (CR), time and PSNR for Image Lenna by different methods (Domain Skip Distance=1)

<table>
<thead>
<tr>
<th>Tolerance $e_c$</th>
<th>Classified Search Method</th>
<th>Variance Method</th>
<th>Domain Entropy Method</th>
<th>Domain-Range Entropy (Proposed) Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CR</td>
<td>Time (Sec)</td>
<td>PSNR (dB)</td>
<td>CR</td>
</tr>
<tr>
<td>1.0</td>
<td>3.84</td>
<td>152.75</td>
<td>37.76</td>
<td>3.84</td>
</tr>
<tr>
<td>2.0</td>
<td>4.40</td>
<td>139.43</td>
<td>37.65</td>
<td>4.39</td>
</tr>
<tr>
<td>4.0</td>
<td>8.12</td>
<td>90.31</td>
<td>36.51</td>
<td>8.11</td>
</tr>
<tr>
<td>6.0</td>
<td>11.53</td>
<td>72.75</td>
<td>35.26</td>
<td>11.45</td>
</tr>
<tr>
<td>8.0</td>
<td>15.10</td>
<td>60.04</td>
<td>34.09</td>
<td>15.01</td>
</tr>
<tr>
<td>10.0</td>
<td>19.10</td>
<td>51.75</td>
<td>32.88</td>
<td>18.72</td>
</tr>
<tr>
<td>15.0</td>
<td>30.08</td>
<td>40.70</td>
<td>30.71</td>
<td>29.32</td>
</tr>
<tr>
<td>20.0</td>
<td>41.37</td>
<td>35.25</td>
<td>29.20</td>
<td>40.10</td>
</tr>
</tbody>
</table>
Figures 4.8 and 4.9 show the rate distortion curves for image Lenna by the various methods. As seen from the graphs, it is observed that the encoding time for Image Lenna by the proposed method is less than the classified search method and Domain entropy method. The variance method resulted in much faster encoding times. From Figure 4.9, it is observed that the Compression ratio and PSNR for the proposed method are very near to those obtained by the classified search method.

Thus the proposed method resulted in fast encoding than the baseline method and domain entropy method while retaining the compression ratio and PSNR factor.

Figures 4.10 and 4.11 show the rate distortion curves by varying domain skip distances $\delta_h$ and $\delta_v$ (4, 2, 1) on Lenna Image, by classified search method and domain-range entropy (proposed) method. The encoding time increases with decreased skip distance in all the methods. This is due to the increase in number of domain-range mapping comparisons. The graphs indicate that the time taken for encoding by the proposed method is less than the classified search method in all the three cases.

Figures 4.12, 4.14, and 4.16 show the number of domains and ranges used during encoding in each of the methods (for domain skip distances 4, 2, 1). Figures 4.13, 4.15, and 4.17 show the number of domain-range mapping comparisons during the encoding phase in each of the methods. It is seen that the proposed domain-range method results in least number of comparisons. The entropy function used as a decision rule for mapping comparisons reduces the encoding time by eliminating many of the unqualified domain-range comparisons.
Figure 4.8 Graph showing encoding time vs. PSNR for Image Lenna by different methods
Figure 4.9 Graph showing compression Ratio vs. PSNR for Image Lena by different methods
Figure 4.10 Encoding time vs. PSNR for image Lena with domain skip distances of 4, 2, and 1.
Figure 4.11 Encoding time vs. PSNR for image Lenna with domain skip distances of 4, 2, and 1
NO. OF DOMAINS AND RANGES FOR IMAGE LENNA
(Domain Skip Distance=4)

Figure 4.12 Histogram showing the number of domains and ranges in different methods
Figure 4.13 Histogram showing the number of domains and range comparisons in different methods
NO. OF DOMAINS AND RANGES FOR IMAGE LENNA
(Domain Skip Distance = 2)

Figure 4.14 Histogram showing the number of domains and ranges in different methods
Figure 4.15 Histogram showing the number of domains and range comparisons in different methods
NO. OF DOMAINS AND RANGES FOR IMAGE LENNA
(Domain Skip Distance = 1)

Figure 4.16 Histogram showing the number of domains and ranges in different methods.
NO. OF DOMAIN AND RANGE MAPPING COMPARISONS
FOR IMAGE LENNA (Domain Skip Distance = 1)

Figure 4.17 Histogram showing the number of domains and range comparisons in different methods
Figures 4.18 (a) and (b) show the original image of Baboon (512x512, 8 bit) and the quadtree partitioning of image Baboon (classified search method, CR=5.96, PSNR=25.22 dB, domain skip distance=4). Figure 4.19 (a) shows the image of Baboon decoded by classified search algorithm (PSNR=21.28 dB, CR=19.58). Figure 4.19 (b) shows the image of Baboon decoded by variance method (PSNR=20.89 dB, CR=16.37). Figure 4.20 (a) shows the image of Baboon decoded by domain entropy method (PSNR=21.32 dB, CR=13.14). Figure 4.20 (b) shows the image of Baboon decoded by the proposed domain-range entropy method (PSNR=21.28 dB, CR=19.09). Figures 4.21 (a) and (b) show the comparison between Baboon decoded by the proposed method (PSNR=21.28 dB, CR=19.09) and JPEG decoding (PSNR=23.42 dB, CR=20.48). Figures 4.22 (a) and (b) show the difference of the original and decoded images of Baboon by the proposed domain-range entropy method and JPEG decoding.

Table 4.4 shows the compression ratio, time and PSNR values obtained for image Baboon by the classified search method, variance method, domain entropy method, and domain-range entropy (proposed) method, with domain skip distance set at $\delta_h=\delta_v=4$. The above values are obtained by varying the value of tolerance parameter $e$, between 1.0 and 20.0. The compression ratios varied from 4.36 (classified search method) to 19.58 (classified search method). PSNR varied from 25.49 dB (classified search method) to 20.89 dB (variance method). The encoding time varied from 7.45 (classified search method) seconds to 1.18 (variance method) seconds. Figures 4.23 and 4.24 show the rate distortion curves for image Baboon by the above methods. It can be seen that the proposed method takes reduced time for encoding when compared to the classified search and domain entropy method. The variance method results in slightly less encoding time.
than the proposed method. But the compression ratio and PSNR factor are better with the proposed method when compared to the variance method. Thus, the proposed method results in both reduced encoding time and also good fidelity of the decoded images at high compression ratios, nearer to the classified search algorithm (baseline method). The reduction in time is due to the reduced number of domain-range comparisons in the encoding phase.
Figure 4.18 (a) Original Image of Baboon (512x512, 8 bit grey scale, raw format) (b) Quadtree Partition of Baboon
Figure 4.18 (a) Original Image of Baboon (512x512, 8 bit grayscale, raw format) (b) Quadtree Partition of Baboon
Figure 4.19 Decoded Images of Baboon
Figure 4.20 Decoded Images of Baboon

(a) Domain Entropy Method
(PSNR=21.33, CR=15.14)

(b) Domain-Range Entropy (Proposed) Method
(PSNR=21.28, CR=19.09)
Figure 4.21 Comparison of Proposed method and JPEG

(a) Decoded image encoded by Proposed Method
(PSNR=21.28, CR=19.09)

(b) Decoded image encoded in standard JPEG
(PSNR=22.42, CR=20.48)
Figure 4.22 Difference Images

(a) Difference of original image Baboon and image encoded by proposed method

(b) Difference of original image Baboon and image encoded in standard JPEG
Table 4.4 Compression Ratio (CR), time and PSNR for Image Baboon by different methods (domain skip distance=4)

<table>
<thead>
<tr>
<th>Tolerance $e_c$</th>
<th>CR</th>
<th>Time (Sec)</th>
<th>PSNR (dB)</th>
<th>CR</th>
<th>Time (Sec)</th>
<th>PSNR (dB)</th>
<th>CR</th>
<th>Time (Sec)</th>
<th>PSNR (dB)</th>
<th>CR</th>
<th>Time (Sec)</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.0</td>
<td>4.36</td>
<td>7.45</td>
<td>25.49</td>
<td>4.37</td>
<td>3.98</td>
<td>22.71</td>
<td>4.36</td>
<td>4.81</td>
<td>25.49</td>
<td>4.36</td>
<td>5.96</td>
</tr>
<tr>
<td></td>
<td>2.0</td>
<td>4.36</td>
<td>7.36</td>
<td>25.49</td>
<td>4.37</td>
<td>3.96</td>
<td>22.71</td>
<td>4.36</td>
<td>4.82</td>
<td>25.49</td>
<td>4.36</td>
<td>5.90</td>
</tr>
<tr>
<td></td>
<td>4.0</td>
<td>4.45</td>
<td>7.26</td>
<td>25.49</td>
<td>4.45</td>
<td>3.89</td>
<td>22.71</td>
<td>4.40</td>
<td>4.78</td>
<td>25.49</td>
<td>4.42</td>
<td>5.85</td>
</tr>
<tr>
<td></td>
<td>8.0</td>
<td>5.44</td>
<td>6.26</td>
<td>25.37</td>
<td>5.43</td>
<td>2.98</td>
<td>22.64</td>
<td>5.08</td>
<td>4.23</td>
<td>25.38</td>
<td>5.40</td>
<td>5.20</td>
</tr>
<tr>
<td></td>
<td>10.0</td>
<td>5.96</td>
<td>5.82</td>
<td>25.22</td>
<td>5.94</td>
<td>2.62</td>
<td>22.56</td>
<td>5.46</td>
<td>4.00</td>
<td>25.25</td>
<td>5.91</td>
<td>4.90</td>
</tr>
<tr>
<td></td>
<td>15.0</td>
<td>7.30</td>
<td>5.00</td>
<td>24.64</td>
<td>7.21</td>
<td>2.09</td>
<td>22.33</td>
<td>6.42</td>
<td>3.57</td>
<td>24.76</td>
<td>7.25</td>
<td>4.31</td>
</tr>
<tr>
<td>30.0</td>
<td>19.58</td>
<td>2.70</td>
<td>21.28</td>
<td>16.37</td>
<td>1.18</td>
<td>20.89</td>
<td>13.14</td>
<td>2.18</td>
<td>21.32</td>
<td>19.09</td>
<td>2.64</td>
<td>21.28</td>
</tr>
</tbody>
</table>
Figure 4.23 Graph showing encoding time vs. PSNR for Image Baboon by different methods
Figure 4.24 Graph showing compression Ratio vs. PSNR for Image Baboon by different methods
Figures 4.25 (a) and (b) show the original image of Peppers (512x512, 8 bit) and quadtree partitioning Peppers (classified search method, CR=21.14, PSNR=31.90 dB, domain skip distance=4). Figure 4.26 (a) shows the image of Peppers decoded by classified search algorithm (PSNR=28.72 dB, CR=45.64). Figure 4.26 (b) shows the image of Peppers decoded by variance method (PSNR=28.41 dB, CR=41.78). Figure 4.27 (a) shows the image of Peppers decoded by domain entropy method (PSNR=28.41 dB, CR=24.71). Figure 4.27 (b) shows the image of Peppers decoded by the proposed domain-range entropy method (PSNR=28.65 dB, CR=43.85). Figures 4.28 (a) and (b) gives a comparison of the image Peppers decoded by the proposed method (PSNR=28.65 dB, CR=43.85) and JPEG decoding (PSNR=29.70 dB, CR=42.80). It is noticed, that the JPEG decoded image shows blocky artifacts at a compression ratio which is less than the proposed method. The proposed method shows a smooth decoding.

Figures 4.29 (a) and (b) show the difference of the original and decoded images of Peppers by the proposed domain-range entropy method and JPEG decoding. The difference image by the proposed method has no difference in smooth regions of the image. The difference image in JPEG shows differences over the entire image.

Table 4.5 shows the compression ratio, time and PSNR values obtained for image Peppers by the classified search method, variance method, domain entropy method, and domain-range entropy (proposed) method, with domain skip distance, δh=δv=4. The above values are obtained by varying the value of tolerance parameter $e_o$ between 1.0 and 20.0. The compression ratios varied from 4.36 (classified search method) to 24.71 (domain entropy method). PSNR varied from 34.83 dB (classified search method) to 28.41 dB (variance method). The encoding time varied from 9.64 (classified search method) seconds to 1.62 (variance method) seconds. Figures 4.30 and 4.31 show the rate
distortion curves for image Peppers by the above methods. It can be seen that the proposed method takes reduced time for encoding when compared to the classified search and domain entropy method. The variance method results in slightly less encoding time than the proposed method. But the compression ratio and PSNR factor are better with the proposed method when compared to the variance method. The domain entropy method results in good fidelity of the decoded image but the compression ratios obtained are very low, than all other methods.

The proposed method results in reduced encoding time and good fidelity of the decoded images at high compression ratios, nearly equal to the classified search method (baseline method). The reduction in time is due to the reduced number of domain-range comparisons in the encoding phase.
Figure 4.25 (a) Original Image of Peppers (512x512, 8 bit gray scale, raw format) (b) Quadtree Partition of Peppers
(a) Classified Search Method  
(PSNR=28.72, CR=45.64)

(b) Variance Method  
(PSNR=28.41, CR=41.78)

Figure 4.26 Decoded Images of Peppers
Figure 4.27 Decoded Images of Peppers
Figure 4.28: Comparison of Proposed method and JPEG.
Figure 4.29 Difference Images

(a) Difference of original image Peppers and image encoded by proposed fractal method

(b) Difference of original image Peppers and image encoded in standard JPEG
Table 4.5 Compression Ratio (CR), Time and PSNR for Image Peppers by different methods (domain skip distance=4)

<table>
<thead>
<tr>
<th>Tolerance $c_s$</th>
<th>CR</th>
<th>Time (Sec)</th>
<th>PSNR (dB)</th>
<th>CR</th>
<th>Time (sec)</th>
<th>PSNR (dB)</th>
<th>CR</th>
<th>Time (Sec)</th>
<th>PSNR (dB)</th>
<th>CR</th>
<th>Time (sec)</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>4.36</td>
<td>9.64</td>
<td>34.83</td>
<td>4.36</td>
<td>5.90</td>
<td>31.94</td>
<td>4.36</td>
<td>6.39</td>
<td>34.83</td>
<td>4.36</td>
<td>4.92</td>
<td>34.73</td>
</tr>
<tr>
<td>2.0</td>
<td>4.44</td>
<td>9.48</td>
<td>34.83</td>
<td>4.44</td>
<td>5.87</td>
<td>31.94</td>
<td>4.42</td>
<td>6.32</td>
<td>34.83</td>
<td>4.40</td>
<td>4.79</td>
<td>34.73</td>
</tr>
<tr>
<td>4.0</td>
<td>6.93</td>
<td>7.18</td>
<td>34.43</td>
<td>6.92</td>
<td>4.10</td>
<td>31.73</td>
<td>6.32</td>
<td>4.85</td>
<td>34.43</td>
<td>6.71</td>
<td>4.00</td>
<td>34.32</td>
</tr>
<tr>
<td>8.0</td>
<td>16.53</td>
<td>4.25</td>
<td>32.75</td>
<td>16.21</td>
<td>2.42</td>
<td>30.77</td>
<td>11.58</td>
<td>3.12</td>
<td>32.70</td>
<td>16.18</td>
<td>2.70</td>
<td>32.61</td>
</tr>
<tr>
<td>10.0</td>
<td>21.14</td>
<td>3.70</td>
<td>31.90</td>
<td>20.44</td>
<td>2.14</td>
<td>30.40</td>
<td>13.73</td>
<td>2.78</td>
<td>31.90</td>
<td>20.71</td>
<td>2.45</td>
<td>31.78</td>
</tr>
<tr>
<td>15.0</td>
<td>32.92</td>
<td>2.84</td>
<td>30.19</td>
<td>31.15</td>
<td>1.79</td>
<td>29.34</td>
<td>18.81</td>
<td>2.26</td>
<td>30.19</td>
<td>31.98</td>
<td>2.09</td>
<td>30.08</td>
</tr>
<tr>
<td>20.0</td>
<td>45.64</td>
<td>2.40</td>
<td>28.72</td>
<td>41.78</td>
<td>1.62</td>
<td>28.41</td>
<td>24.71</td>
<td>1.89</td>
<td>28.41</td>
<td>43.85</td>
<td>1.90</td>
<td>28.65</td>
</tr>
</tbody>
</table>
Figure 4.30 Graph showing encoding time vs. PSNR for Image peppers by different methods
Figure 4.31 Graph showing compression Ratio vs. PSNR for Image Peppers by different methods
Figures 4.32 (a) and (b) show the original image of Goldhill (512x512, 8 bit) and quadtree partitioning of image Goldhill (classified search method, CR=12.50, PSNR=30.64 dB, domain skip distance=4). Figure 4.33 (a) shows the image of Goldhill decoded by classified search algorithm (PSNR=26.87 dB, CR=43.55). Figure 4.33 (b) shows the image of Goldhill decoded by variance method (PSNR=26.96 dB, CR=39.31). Figure 4.34 (a) shows the image of Goldhill decoded by domain entropy method (PSNR=26.96 dB, CR=15.78). Figure 4.34 (b) shows the image of Goldhill decoded by the proposed domain-range entropy method (PSNR=26.81 dB, CR=40.11). Figures 4.35 (a) and (b) show the comparison between Goldhill decoded by the proposed method (PSNR=26.81 dB, CR=40.11) and JPEG decoding (PSNR=28.18 dB, CR=41.09). Figures 4.36 (a) and (b) show the difference of the original and decoded images of Goldhill by the proposed domain-range entropy method and JPEG decoding.

Table 4.6 shows the compression ratio, time and PSNR values obtained for image Goldhill by the classified search method, variance method, domain entropy method, and domain-range entropy (proposed) method, with domain skip distance, $\delta_h=\delta_v=4$. The above values are obtained by varying the value of tolerance parameter $c_0$ between 1.0 and 20.0. The compression ratios varied from 4.37 (classified search method) to 23.24 (domain entropy method). PSNR varied from 33.87 dB (classified search method) to 26.13 dB (domain entropy method). The encoding time varied from 9.01 (classified search method) seconds to 1.48 (variance method) seconds. Figures 4.37 and 4.38 show the rate distortion curves for image Goldhill by the above methods. It can be seen that the proposed method takes reduced time for encoding when compared to the classified search and domain entropy method. The variance method results in slightly less encoding time than the proposed method. But the compression ratio and PSNR factor are better with the
proposed method when compared to the variance method. The domain entropy method results in good fidelity of the decoded image but the compression ratios obtained are very less than all other methods.

Thus, the proposed method results in both reduced encoding time and also good fidelity of the reconstructed images at high compression ratios, nearly equal to the classified search algorithm (baseline method). The reduction in time is due to the reduced number of domain-range comparisons in the encoding phase.
Figure 4.32 (a) Original Image of Goldhill (512x512, 8 bit gray scale, raw format) (b) Quadtree Partition of Goldhill
Figure 4.33 Decoded Images of Goldhill

(a) Classified Search Method
   (PSNR=26.87, CR=43.55)

(b) Variance Method
   (PSNR=26.96, CR=39.31)
<table>
<thead>
<tr>
<th>Domain Entropy Method</th>
<th>Domain-Range Entropy (Proposed) Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISNR=26.96, CR=15.78</td>
<td>PSNR=26.81, CR=40.11</td>
</tr>
</tbody>
</table>

(a) (b) Figure 4.34 Decoded Images of Goldhill
Figure 4.35 Comparison of Proposed method and JPEG

(a) Decoded image encoded by Proposed Method
(PSNR=26.81, CR=40.11)

(b) Decoded Image encoded in standard JPEG
(PSNR=28.18, CR=41.09)
(a) Difference of original image Goldhill and image encoded by proposed method

(b) Difference of original image Goldhill and image encoded in standard JPEG

Figure 4.36 Difference Images
Table 4.6 Compression Ratio (CR), Time and PSNR for Image Goldhill by different methods (domain skip distance=4)

<table>
<thead>
<tr>
<th>Classified Search Method</th>
<th>Variance Method</th>
<th>Domain Entropy Method</th>
<th>Domain-Range Entropy (Proposed) Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tolerance ( \epsilon_c )</td>
<td>CR</td>
<td>Time (Sec)</td>
<td>PSNR (dB)</td>
</tr>
<tr>
<td>1.0</td>
<td>4.37</td>
<td>9.01</td>
<td>33.87</td>
</tr>
<tr>
<td>2.0</td>
<td>4.61</td>
<td>8.72</td>
<td>33.86</td>
</tr>
<tr>
<td>4.0</td>
<td>5.11</td>
<td>8.09</td>
<td>33.72</td>
</tr>
<tr>
<td>6.0</td>
<td>6.76</td>
<td>6.75</td>
<td>32.90</td>
</tr>
<tr>
<td>8.0</td>
<td>9.15</td>
<td>5.62</td>
<td>31.75</td>
</tr>
<tr>
<td>10.0</td>
<td>12.50</td>
<td>4.52</td>
<td>30.64</td>
</tr>
<tr>
<td>15.0</td>
<td>25.63</td>
<td>3.01</td>
<td>28.31</td>
</tr>
<tr>
<td>20.0</td>
<td>43.55</td>
<td>2.29</td>
<td>26.87</td>
</tr>
</tbody>
</table>
Figure 4.3: Graph showing encoding time vs. PSNR for image Goldhill by different methods
Figure 4.38 Graph showing Compression Ratio vs. PSNR for Image Goldhill by different methods
Figure 4.36 Graph showing Compression Ratio vs. PSNR for Image Goldhill by different methods.
4.6 INFERENCE

Based on the experimental results for the images Lenna, Baboon, Peppers and Goldhill, the following inferences are drawn.

1) The subjective quality of the decoded image by domain-range entropy method is comparable to the image decoded by classified search method. The drop in PSNR is between 0.10 to 0.14 dB. The reduction in compression ratio varied from 0.01 to 0.89, which is very small.

2) The visual quality of image decoded by the proposed method is observed to be better than the JPEG decoded image at same compression ratio. In the JPEG decoded image, at compression ratio 40.00, the PSNR obtained is 30.41dB. With the proposed method, at compression ratio 41.02, the PSNR obtained is 28.81dB. From figure 4.7 (a) & (b) it is noticed that the JPEG encoded image resulted in blocky artifacts. The image decoded by the proposed method resulted in a smooth decoding.

3) The difference of original Lenna image and image decoded by proposed method (figure 4.8 (a)) shows no difference in smooth regions. The difference image obtained with JPEG decoder shows a difference over the entire image.

4) Figure 4.9 & 4.10 show the rate distortion curves for image Lenna by different methods. The graphs show that the encoding time by the domain range entropy method is less than the classified search method, variance method and domain entropy method. At low PSNR values, the variance method resulted in lesser encoding times. The compression ratios obtained by variance method are less than that obtained by the proposed method.

5) Changing the domain skip distances, \( \delta_h \), and \( \delta_v \) from 4 to 2 and 1, resulted in more time for encoding in all the methods. This is due to the more number of domains
and ranges involved in the comparison during encoding phase. However, the proposed domain-range entropy method resulted in fast encoding time when compared to all the other methods, at low and high compression ratios. The quality of the decoded image and compression ratio are nearly equal to that obtained by classified search method.

6) The compression ratios and PSNR values obtained are nearly equal to those obtained by classified search method (baseline algorithm). The fast encoding time of the proposed method is due to the reduced number of domain-range mapping comparisons during the encoding phase, when compared to the other methods. This is due to the entropy function used as the decision parameter for domain-range mapping comparison. Only the domains and ranges, which possess nearly equal entropies, are compared for a match within the rms error tolerance (which is dependent on the parameter $e_c$).

7) The results also indicate that for images (Lenna, Peppers and Goldhill) which possess a smooth texture, the compression ratios obtained are high in all the methods. The compression ratio and PSNR for image Baboon are less due to the continuously varying texture of the image in all regions.

8) The time for decoding the image varied from 0.280 seconds (for low compression) to 0.171 seconds (for high compression). The initial image is chosen with all pixels set to a mid gray value. This choice resulted in a faster decoding.

9) For decoding the original image, 10 iterations are used. However, 8 iterations are sufficient to obtain good quality of the reconstructed image. An increase of 0.1 dB is obtained from 8 to 10 iterations. After 10 iterations, the PSNR of decoded image has fairly remained constant.

10) Applying post processing resulted in a gain of PSNR by about 0.2 dB for high fidelity images, and about 0.3 dB to 0.5 dB for other images.
4.7 SUMMARY

In this chapter, implementation of the proposed domain-range entropy method for fast fractal image compression has been described. The algorithms for compression and decompression are presented. Experiments are conducted on gray scale images of size 512x512 (8-bit) possessing different textures. The results obtained are discussed, and compared with other related methods. It is inferred that the proposed method results in reduced encoding time when compared to other related methods. The fidelity and compression ratio are nearly equal to the classified search method.

In the next chapter, the proposed gray level difference algorithm to speed up fractal image compression is described.
CHAPTER 5

GRAY LEVEL DIFFERENCE METHOD TO SPEED-UP FRACTAL IMAGE COMPRESSION