5.1 INTRODUCTION

In this chapter the proposed gray level difference method to speed up fractal image compression is described. The method is based on quadtree partitioning algorithm with classified search [9]. The maximum and minimum gray level intensities of the pixels in the domain block and range block are computed. The gray level difference of these pixels is then computed. During encoding, range blocks with gray level difference less than the gray level difference of domain blocks are only compared for a best match. This reduces the number of domain and range comparisons. Thus the encoding time is expected to reduce significantly. The detailed proof of proposed method, implementation, and the experimental results obtained on various standard 8 bit gray scale images of size 512x512 are presented. The results are compared with a related method, average variance function method proposed by [17].

5.2 PROPOSED METHOD

Let R and D be the range and domain blocks under comparison. The maximum and minimum gray level intensities of the pixels in the range block, and domain block are computed as gmax (R), gmin (R), gmax (D), and gmin (D), respectively. Then, gdiff(R) = gmax(R) – gmin(R), and gdiff(D) = gmax (D) – gmin (D) are computed. A comparison for the best matching domain-range pair is done only if, gdiff(R) <= gdiff(D). The method will be applied for fractal compression of gray scale images of size 512x512.
5.2.1 Gray Level Difference

Let the maximum and minimum gray level values of the pixels in a square image block, B, are respectively, \( \text{gmax}(B) \) and \( \text{gmin}(B) \). The gray level difference of image block B is defined by,

\[
g\text{diff}(B) = \text{gmax}(B) - \text{gmin}(B) \tag{5.1}
\]

5.2.2 Gray Level Difference Algorithm

Consider a single pixel in a domain block D. The affine transformation in fractal encoding maps its gray value \( g_i \) to the range block R, using the equation,

\[
g_i(R) = s \cdot g_i(D) + o \tag{5.2}
\]

The contrast scaling parameter, \( s \) must satisfy the condition \( 0 < s < 1 \).

Combining equations (5.1) and (5.2),

\[
g\text{diff}(R) = \text{gmax}(R) - \text{gmin}(R) \tag{5.3}
\]

\[
= \{ s \cdot \text{gmax}(D) + o \} - \{ s \cdot \text{gmin}(D) + o \}
\]

\[
= s \cdot \{ \text{gmax}(D) - \text{gmin}(D) \} \tag{5.4}
\]

Considering the contrast scaling requirement, \( 0 < s < 1 \), equation (5.4) is written as,

\[
g\text{diff}(R) < g\text{diff}(D) \tag{5.5}
\]

In actual implementation, equation (5.5) is modified as,

\[
g\text{diff}(R) < \beta \cdot g\text{diff}(D) \tag{5.6}
\]

Where, \( \beta \) is an adaptive scaling parameter (with an initial value between 1.0 and 2.0, for quadtree depth 0) for each quadtree partition. Equation (5.6) provides an effective decision rule to avoid an improper domain and range match. Only, domains satisfying the above condition will be subjected to further linear regression analysis.
Adaptive scale parameter for domain block gray level difference

An adaptive parameter $\beta_{\text{depth}}$ is used for each quadtree depth, $i$ to scale the gray level difference of the domain blocks. For quadtree depth 0, (corresponding to min_part), $\beta_0$ is assigned a small initial value (in the present work, $\beta_0=1.25$). For other quadtree depths, $\beta$ is computed using the formula, $\beta_{\text{depth}}=1.25*\beta_{\text{depth-1}}$. This equation is fit by conducting experiments on images of different sizes and textures, testing for optimal value of encoding time, fidelity and compression ratio.

5.2.3 Domain pool classification

The given image $I$ (of size $hsize \times vsize$) is read from the input file and stored in to an array ‘image’. The minimum partitions (min_part) and the maximum partitions (max_part), the rms error tolerance ($e_c$), are also read as input. Then, the number of quadtree partitions (depth=max_part-min_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.

The construction and classification of the domain pools is done using the procedure described in section 4.2.1. The gray level difference value of each domain block is computed and stored, using the equation (5.1).

5.2.4 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, vsize
Step 2: Initialize \( \text{depth}=0, \text{best}_\text{rms}=\text{infinity} \). The given image is quadtree partitioned recursively until depth is equal to minimum partitions (\( \text{min}_\text{part} \)). The value of depth is incremented by one for each quadtree partition.

Step 3: Select a range block \( R_i \) (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. Compute the gray level difference \( \text{gdiff}(R) \) using equation (5.1).

Step 5: Select a domain from the classified domain lists corresponding to the size \( 2rx2r \), and same classification of the range.

Step 6: The domain and range gray level difference is compared using equation (5.6). If the condition in equation is satisfied, go to step 7, otherwise, go to step 5.

Step 7: Compute the \text{rms}\_\text{error} using equation (4.6) for the current domain, range pair. If the \text{rms}\_\text{error} \( \leq \text{best}_\text{rms} \), set this as the best\_rms and repeat steps 3 to 5 for all the domains in the current list.

Step 8: If the best\_rms > \( e \) and depth < \( \text{max}_\text{part} \), partition the range block and repeat steps 1 to 7, Else, mark the range \( R_i \) as covered.

Step 9: Write in to output file, the transformation \( w_i \) (fractal code), comprising of the domain position, (\( x,y \) coordinate values), the symmetry operation (\( \text{sym}\_\text{op} \)), the contrast scaling (\( s \)) and the luminance offset (\( o \)). This constitutes the fractal code for the given range.

Repeat steps 1 to 8 for all the quadtree partitions.

\textit{Computing s and o}: The scaling coefficient and the luminance offset are computed using the procedure described in Section 4.2.2.
5.2.5 Encoding Algorithm

Step 1: Construct domain pools $D_k$, corresponding to each quadtree partition level starting from minimum partitions to maximum partitions ($k=0$ to $\max\_part-\min\_part$).

Step 2: Calculate the block gray level difference of the entire domain blocks in each pool $D_k$.

Step 3: Classify and sort the domains in each pool $D_k$ in ascending order of the gray level difference, and place on a list structure.

Step 4: Search for the best match between a range and domain belonging to the same class.

write_header_info; (min_part, max_part, domain_step, hsize, vsize)

depth=0; $e_c=rms\_tol$

function Quadtree(image, depth) {

best_rms=infinity; $\beta_0=$initial value; $\beta_{\text{depth}} = 1.25 \times \beta_{\text{depth}}$

while (depth<min_part) Quadtree (image,depth+1);

Set $R_1 = I^2$ and mark it uncovered.

while there are uncovered ranges $R_i$ do {

//Select the domain pool list $D_k$ corresponding to the current range block $R_i$.

for (j=1; j<num_domains; ++j) {

If $(R_{\text{diff}} < \beta\times D_{\text{diff}})$ {

compute s, o, sym_op;

compute E $(R_i, D_i)$;

if $E (R_i, D_i) \leq$ best_rms {

best_rms= E $(R_i, D_i)$;

}
best_domain=(domain_x,domain_y);//domain address
}
}// end for num_domains

if (best_rms>e) and (depth<max_part) Quadtree (image, depth+1);
else write_transformations (best_domain, s, o, sym_op);
}//end while uncovered ranges
}//end function Quadtree()

Input parameters for the algorithm:

- The given image and its size (hsize, vsize)
- The rms tolerance threshold e.
- The initial value of adaptive scaling parameter β₀.
- The maximum depth of the quadtree partition (max_part) (depends on the size of input image).
- The minimum depth of the quadtree partition (min_part) (depends on the size of input image).
- The domain skip distance δ₀ and δₜ.
- The number of bits to be 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 final quadtree partition of the image.
- The domain address (domain_x, domain_y).
- The scaling and offset values for each range (sᵢ and oᵢ).
- The orientation and flip information (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 in this case.

**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 (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.

### 5.2.6 Improved Gray Level Difference Algorithm

The gray level difference method proposed in section 5.2.1, can be improved to further speed up the encoding process. The method is outlined below.

**Step 1:** Construct the domain pools $D_k$, corresponding to each quadtree partition level starting from minimum partitions to maximum partitions ($k=0$ to $\text{max_part-min_part}$).

**Step 2:** Compute the variance of individual domain blocks in each domain pool $D_k$.

**Step 3:** Eliminate duplicate domains (i.e. domains having similar variance, within an error threshold of 1.0) from each of the domain pools.

The above procedure reduces the number of domains in the domain pools significantly. Thus, faster encoding times are expected with a negligible loss in quality of the decoded image. The loss in quality is proportional to the error threshold used for computing variance similarity.

**Decoding of the compressed image:** The fractal compressed image is decoded using the procedure and algorithm described in Section 4.5.
5.3 EXPERIMENTATION

In this section results of the experimental investigations conducted on different images (size 512x512, 8 bit gray scale) are presented. The results are compared with a similar method, the average variance function method [17].

The following values are used for various parameters:

- 5 bits are used to quantize the scaling coefficient s, and 7 bits for the offset, o.
- For all images, the maximum range size is 16x16 (minimum quadtree depth 5), and the minimum range size is 4x4 (maximum quadtree depth 7). Three quadtree levels are used.
- The domain pool is constructed with domain skip distance, δ_h=4 and δ_v=4.
- The rms error tolerance, \( c \), is assigned values 1, 2, 4, 6, 8, 10, 15, and 20, which results in low to high compression. PSNR is computed after post processing.
- 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 with clock frequency @1.73 GHz, with 1.0 GB of RAM.

5.3.1 Encoding Parameters

Images size = 512x512 (8 bit gray scale)

Number of quadtree partitions = 3

Total Number of Domains:

(i) Domain Skip Distance, \( \delta_h=\delta_v=4 \).

Size 32x32 = \( \frac{(512-32)}{4+1} \)*\( (512-32)/4+1 \) = 14,641

Size 16x16 = \( \frac{(512-16)}{4+1} \)*\( (512-16)/4+1 \) = 15,625

Size 8x8 = \( \frac{(512-8)}{4+1} \)* \( (512-8)/4+1 \) = 16,129

129
(ii) Domain Skip Distance, $\delta_v=\delta_r=2$.

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

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

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

(iii) Domain Skip Distance, $\delta_v=\delta_r=1$.

Size 32x32 = \((512-32)/1+1\)\*\((512-32)/1+1\) = 2,31,361

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

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

In the proposed method, the adaptive parameter $\beta_{\text{depth}}$ (to scale the gray level difference of domain blocks) is assigned an initial value, $\beta_0=1.25$ for quadtree depth 0.

For other quadtree depths, $\beta_{\text{depth}}=1.25^*\beta_{\text{depth}-1}$
(ii) Domain Skip Distance, $\delta_k=\delta_r=2$.

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

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

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

(iii) Domain Skip Distance, $\delta_k=\delta_r=1$.

Size 32x32 = \((512-32)/1+1\) * \((512-32)/1+1\) = 2,31,361

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

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

In the proposed method, the adaptive parameter $\beta_{\text{depth}}$ (to scale the gray level difference of domain blocks) is assigned an initial value, $\beta_0=1.25$ for quadtree depth 0.

For other quadtree depths, $\beta_{\text{depth}}=1.25*\beta_{\text{depth}-1}$
5.4 RESULTS AND DISCUSSION

Figure 5.1 (a) shows the image of Lenna decoded by the average-variance method (PSNR=28.90 dB, CR=41.55). Figure 5.1 (b) shows the image of Lenna decoded by adaptive gray level difference (proposed) method (PSNR= 28.96 dB, CR=41.71).

The compression ratio, time and PSNR values obtained for image Lenna by the average-variance method and adaptive gray level difference (proposed) method (domain skip distance set at $\delta_h=\delta_v=4$) are given in Table-5.1.

For the average variance method, the compression ratios varied from 4.36 to 41.55 and PSNR varied from 36.05 dB to 28.90 dB. The encoding time varied from 8.07 seconds to 2.18 seconds.

Tables 5.2 and 5.3 give the encoding results obtained for the image Lenna for domain skip distances equal to 2 ad 1. The results indicate that the encoding times are high as the skip distance is reduced to 2 and 1. This is because of the more number of domains computed in all the methods, which result in larger number of domain-range comparisons.

For the adaptive gray level difference (proposed) method, the compression ratios varied from 4.36 to 41.07 and PSNR varied from 35.97 dB to 28.92 dB. The encoding time varied from 4.29 seconds to 1.48 seconds.
Figure 5.3: Decoded Images of Lena (512x512, 8 bit gray scale)

(a) Average Variance Method
   (PSNR=28.90, CR=41.55)

(b) Adaptive Gray Level Difference (Proposed) Method
   (PSNR=28.96, CR=41.71)
Table 5.1 Compression Ratio, Time and PSNR for average variance method, adaptive gray level difference method, and improved adaptive gray level difference method (domain skip distance=4)

<table>
<thead>
<tr>
<th>Tolerance</th>
<th>Average Variance Method</th>
<th>Adaptive Gray Level Difference (Proposed Method-1)</th>
<th>Improved Adaptive Gray Level Difference (Proposed Method-2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_2$</td>
<td>CR</td>
<td>Time (sec)</td>
<td>PSNR (dB)</td>
</tr>
<tr>
<td>1.0</td>
<td>4.36</td>
<td>8.07</td>
<td>36.05</td>
</tr>
<tr>
<td>2.0</td>
<td>4.84</td>
<td>7.56</td>
<td>35.99</td>
</tr>
<tr>
<td>4.0</td>
<td>8.63</td>
<td>5.07</td>
<td>35.29</td>
</tr>
<tr>
<td>6.0</td>
<td>11.88</td>
<td>4.14</td>
<td>34.38</td>
</tr>
<tr>
<td>8.0</td>
<td>15.29</td>
<td>3.56</td>
<td>33.46</td>
</tr>
<tr>
<td>10.0</td>
<td>18.97</td>
<td>3.09</td>
<td>32.39</td>
</tr>
<tr>
<td>15.0</td>
<td>29.02</td>
<td>2.52</td>
<td>30.36</td>
</tr>
<tr>
<td>20.0</td>
<td>41.55</td>
<td>2.18</td>
<td>28.90</td>
</tr>
</tbody>
</table>
Fig. 5.2 Graph showing the encoding time vs. PSNR for Image Lenia by different methods
Fig. 5.3 Graph showing compression ratio vs. PSNR for Lenna by different methods
Table 5.2 Compression Ratio, Time and PSNR for average variance method, adaptive gray level difference method, and improved adaptive gray level difference method (Domain skip distance = 2)

<table>
<thead>
<tr>
<th>Tolerance $e_0$</th>
<th>Average Variance Method</th>
<th>Adaptive Gray level Difference (Proposed Method-1)</th>
<th>Improved Adaptive Gray level Difference (Proposed Method-2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CR</td>
<td>Time (sec)</td>
<td>PSNR (dB)</td>
</tr>
<tr>
<td>1.0</td>
<td>4.09</td>
<td>34.23</td>
<td>37.01</td>
</tr>
<tr>
<td>2.0</td>
<td>4.62</td>
<td>31.95</td>
<td>36.93</td>
</tr>
<tr>
<td>4.0</td>
<td>8.40</td>
<td>21.04</td>
<td>35.99</td>
</tr>
<tr>
<td>6.0</td>
<td>11.80</td>
<td>16.87</td>
<td>34.83</td>
</tr>
<tr>
<td>8.0</td>
<td>15.57</td>
<td>14.46</td>
<td>33.78</td>
</tr>
<tr>
<td>10.0</td>
<td>19.22</td>
<td>12.75</td>
<td>32.63</td>
</tr>
<tr>
<td>15.0</td>
<td>29.88</td>
<td>10.37</td>
<td>30.52</td>
</tr>
<tr>
<td>20.0</td>
<td>42.37</td>
<td>9.12</td>
<td>28.96</td>
</tr>
</tbody>
</table>
Table 5.3 Compression vs. PSNR for average variance method, and adaptive Gray level Difference method (Domain skip distance = 1)

<table>
<thead>
<tr>
<th>Tolerance</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>3.84</td>
<td>163.67</td>
<td>37.73</td>
<td>3.84</td>
<td>140.89</td>
<td>37.76</td>
<td>3.84</td>
<td>49.59</td>
<td>37.43</td>
</tr>
<tr>
<td>2.0</td>
<td>4.40</td>
<td>152.57</td>
<td>37.62</td>
<td>4.40</td>
<td>127.39</td>
<td>37.65</td>
<td>4.30</td>
<td>47.65</td>
<td>37.36</td>
</tr>
<tr>
<td>4.0</td>
<td>8.10</td>
<td>102.75</td>
<td>36.48</td>
<td>8.12</td>
<td>86.00</td>
<td>36.51</td>
<td>7.94</td>
<td>39.06</td>
<td>36.41</td>
</tr>
<tr>
<td>6.0</td>
<td>11.48</td>
<td>83.65</td>
<td>35.22</td>
<td>11.50</td>
<td>72.05</td>
<td>35.28</td>
<td>11.20</td>
<td>35.76</td>
<td>35.27</td>
</tr>
<tr>
<td>8.0</td>
<td>15.00</td>
<td>72.96</td>
<td>34.06</td>
<td>15.10</td>
<td>65.59</td>
<td>34.10</td>
<td>14.85</td>
<td>33.84</td>
<td>34.08</td>
</tr>
<tr>
<td>10.0</td>
<td>18.96</td>
<td>64.90</td>
<td>32.85</td>
<td>19.05</td>
<td>61.14</td>
<td>32.90</td>
<td>18.67</td>
<td>32.32</td>
<td>32.91</td>
</tr>
<tr>
<td>15.0</td>
<td>29.62</td>
<td>54.68</td>
<td>30.69</td>
<td>30.00</td>
<td>55.43</td>
<td>30.72</td>
<td>29.62</td>
<td>30.41</td>
<td>30.65</td>
</tr>
<tr>
<td>20.0</td>
<td>41.28</td>
<td>49.40</td>
<td>29.10</td>
<td>41.04</td>
<td>52.84</td>
<td>29.22</td>
<td>40.65</td>
<td>29.46</td>
<td>29.20</td>
</tr>
</tbody>
</table>
Figures 5.4 and 5.5 show the effect of varying the skip distance on the encoding times in average variance method and the proposed adaptive gray level difference method. It is observed that the encoding time increases by decreasing the skip distance. This is due to the increased number of domain and range comparisons.

Figures 5.6 to 5.11 show the number of domains and ranges computed in each method and the number of domain-range mapping comparisons taking place during encoding. The proposed methods result in reduced encoding times due to the less number of domain-range mapping comparisons made during encoding when compared to the average-variance method.
Figure 5.4 Encoding time vs. PSNR for image Lena with domain skip distances of 4, 2, and 1
Figure 5.5: Encoding time vs. PSNR for image Lena with domain skip distances of 4, 2, and 1.
Figure 5.6 Histogram showing the number of domains and ranges in different methods.
NO. OF DOMAIN AND RANGE MAPPING COMPARISONS
FOR IMAGE LENNA (Domain Skip Distance = 4)

![Bar chart showing comparisons between methods.](chart.png)

Figure 5.7 Histogram showing the number of domains and range comparisons in different methods
Figure 5.8 Histogram showing the number of domains and ranges in different methods.
Figure 5.9 Histogram showing the number of domains and ranges in different methods
Figure 5.10 Histogram showing the number of domains and ranges in different methods
Figure 5.11 Histogram showing the number of domains and range comparisons in different methods.
Figure 5.12 (a) shows the image of Baboon decoded by the average-variance method (PSNR=21.26 dB, CR=19.58). Figure 5.12 (b) shows the image of Baboon decoded by adaptive gray level difference (proposed) method (PSNR= 21.44 dB, CR=18.50).

The compression ratio, time and PSNR values obtained for image Baboon by the average-variance method and adaptive gray level difference (proposed) method (domain skip distance $\delta_h=\delta_v=4$) are given in Table-5.4.

For the average variance method, the compression ratios varied from 4.36 to 19.58 and PSNR varied from 25.26 dB to 21.26 dB. The encoding time varied from 7.26 seconds to 2.73 seconds.

For the adaptive gray level difference (proposed) method, the compression ratios varied from 4.36 to 18.29 and PSNR varied from 25.39 dB to 21.41 dB. The encoding time varied from 4.35 seconds to 1.58 seconds.

Figure 5.13 shows the variation of PSNR with encoding time and figure 5.14 shows the variation of PSNR with compression ratio. The proposed method results in reduced encoding times when compared to the average variance method. The drop in PSNR and compression ratio is very small.
Figure 5.12 Decoded images of Baboon (512x512, 8 bit gray scale)

(a) Average Variance Method
(PSNR=21.26, CR=19.58)

(b) Adaptive Gray Level Difference (Proposed) Method
(PSNR=21.44, CR=18.50)
Table 5.4 Compression Ratio, Time and PSNR for average variance method, adaptive gray level Difference method, and improved adaptive gray level difference method (Domain skip distance=4)

<table>
<thead>
<tr>
<th>Tolerance $c_x$</th>
<th>Average Variance Method</th>
<th>Adaptive Gray Level Difference (Proposed Method-1)</th>
<th>Improved Adaptive Gray Level Difference (Proposed Method-2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CR</td>
<td>Time (sec)</td>
<td>PSNR (dB)</td>
</tr>
<tr>
<td>1.0</td>
<td>4.36</td>
<td>7.26</td>
<td>25.26</td>
</tr>
<tr>
<td>2.0</td>
<td>4.36</td>
<td>7.29</td>
<td>25.26</td>
</tr>
<tr>
<td>4.0</td>
<td>4.44</td>
<td>7.17</td>
<td>25.26</td>
</tr>
<tr>
<td>6.0</td>
<td>4.92</td>
<td>6.64</td>
<td>25.22</td>
</tr>
<tr>
<td>8.0</td>
<td>5.43</td>
<td>6.23</td>
<td>25.14</td>
</tr>
<tr>
<td>10.0</td>
<td>5.95</td>
<td>5.73</td>
<td>24.99</td>
</tr>
<tr>
<td>15.0</td>
<td>7.30</td>
<td>4.96</td>
<td>24.44</td>
</tr>
<tr>
<td>20.0</td>
<td>9.14</td>
<td>4.21</td>
<td>23.49</td>
</tr>
<tr>
<td>30.0</td>
<td>19.58</td>
<td>2.73</td>
<td>21.26</td>
</tr>
</tbody>
</table>
Figure 5.13 Graph showing the encoding time vs. PSNR for Baboon by different methods
Figure 5.14 Graph showing compression ratio vs. PSNR for Baboon by different methods
Figure 5.15 (a) shows the image of Goldhill decoded by the average-variance method (PSNR=26.85 dB, CR=43.47). Figure 5.15 (b) shows the image of Goldhill decoded by adaptive gray level difference (proposed) method (PSNR=26.92 dB, CR=42.52).

The compression ratio, time and PSNR values obtained for image Goldhill by the average-variance method and adaptive gray level difference (proposed) method (domain 6h=6v=4) are given in Table-5.5.

For the average variance method, the compression ratios varied from 4.37 to 43.47 and PSNR varied from 33.85 dB to 26.85 dB. The encoding time varied from 8.82 seconds to 2.39 seconds.

For the adaptive gray level difference (proposed) method, the compression ratios varied from 4.37 to 42.30 and PSNR varied from 33.70 dB to 26.88 dB. The encoding time varied from 4.76 seconds to 1.48 seconds.

Figure 5.16 shows the variation of PSNR with encoding time and figure 5.17 shows the variation of PSNR with compression ratio. The proposed method results in reduced encoding time when compared to the average variance method. The drop in PSNR and compression ratio is very small.
Figure 5.15 Decoded images of Goldhill (512x512, 8 bit gray scale)
Table 5.5 Compression Ratio, Time and PSNR for average variance method, adaptive gray level Difference method and improved adaptive gray level difference method (domain skip distance=4)

<table>
<thead>
<tr>
<th>Tolerance ( \epsilon_c )</th>
<th>Method</th>
<th>Average Variance</th>
<th>Adaptive Gray level Difference (Proposed Method-1)</th>
<th>Improved Adaptive Gray level Difference (Proposed Method-2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CR</td>
<td>Time (sec)</td>
<td>PSNR (dB)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>Average Variance</td>
<td>4.37</td>
<td>8.82</td>
<td>33.85</td>
</tr>
<tr>
<td></td>
<td>Adaptive Gray level Difference (Proposed Method-1)</td>
<td>4.61</td>
<td>8.60</td>
<td>33.84</td>
</tr>
<tr>
<td></td>
<td>Improved Adaptive Gray level Difference (Proposed Method-2)</td>
<td>5.10</td>
<td>8.01</td>
<td>33.71</td>
</tr>
<tr>
<td>6.0</td>
<td>Average Variance</td>
<td>6.73</td>
<td>6.68</td>
<td>32.89</td>
</tr>
<tr>
<td></td>
<td>Adaptive Gray level Difference (Proposed Method-1)</td>
<td>9.12</td>
<td>5.45</td>
<td>31.72</td>
</tr>
<tr>
<td></td>
<td>Improved Adaptive Gray level Difference (Proposed Method-2)</td>
<td>12.46</td>
<td>4.48</td>
<td>30.62</td>
</tr>
<tr>
<td>15.0</td>
<td>Average Variance</td>
<td>25.47</td>
<td>3.03</td>
<td>28.29</td>
</tr>
<tr>
<td></td>
<td>Adaptive Gray level Difference (Proposed Method-1)</td>
<td>43.47</td>
<td>2.39</td>
<td>26.85</td>
</tr>
<tr>
<td></td>
<td>Improved Adaptive Gray level Difference (Proposed Method-2)</td>
<td>43.47</td>
<td>2.39</td>
<td>26.85</td>
</tr>
</tbody>
</table>
Figure 5.16 Graph showing encoding time vs. PSNR for Goldhill by different methods
Figure 5.17 Graph showing compression Ratio vs. PSNR for Goldhill by different methods
Figure 5.18 (a) shows the image of Peppers decoded by the average-variance method (PSNR=28.64 dB, CR=45.04). Figure 5.15 (b) shows the image of Peppers decoded by adaptive gray level difference (proposed) method (PSNR= 28.76 dB, CR=45.04).

The compression ratio, time and PSNR values obtained for image Peppers by the average-variance method and adaptive gray level difference (proposed) method (domain skip distance $\delta_h=\delta_v=4$) are given in Table-5.6.

For the average variance method, the compression ratios varied from 4.36 to 45.04 and PSNR varied from 34.07 dB to 28.64 dB. The encoding time varied from 8.32 seconds to 2.14 seconds.

For the adaptive gray level difference (proposed) method, the compression ratios varied from 4.36 to 44.69 and PSNR varied from 34.69 dB to 28.74 dB. The encoding time varied from 4.54 seconds to 1.59 seconds.

Figure 5.19 shows the variation of PSNR with encoding time and figure 5.20 shows the variation of PSNR with compression ratio. The proposed method results in reduced encoding time when compared to the average variance method. The drop in PSNR and compression ratio is very small.
Figure 5.18 Decoded images of Peppers (512x512, 8 bit gray scale)

(a) Average Variance Method (PSNR=28.64, CR=55.04)

(b) Adaptive Gray Level Difference (Proposed) Method (PSNR=28.76, CR=54.04)
### Table 5.6 Compression Ratio, Time and PSNR for average variance method, adaptive gray level Difference method, and improved adaptive gray level difference method (Domain skip distance = 4)

<table>
<thead>
<tr>
<th>Tolerance ( \varepsilon_s )</th>
<th>Average Variance Method CR</th>
<th>Time (sec)</th>
<th>PSNR (dB)</th>
<th>Adaptive Gray level Difference (Proposed Method-1) CR</th>
<th>Time (sec)</th>
<th>PSNR (dB)</th>
<th>Improved Adaptive Gray level Difference (Proposed method-2) CR</th>
<th>Time (Sec)</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>4.36</td>
<td>8.32</td>
<td>34.07</td>
<td>4.36</td>
<td>7.48</td>
<td>34.83</td>
<td>4.36</td>
<td>4.54</td>
<td>34.69</td>
</tr>
<tr>
<td>2.0</td>
<td>4.44</td>
<td>8.34</td>
<td>34.07</td>
<td>4.44</td>
<td>7.46</td>
<td>34.83</td>
<td>4.43</td>
<td>4.53</td>
<td>34.68</td>
</tr>
<tr>
<td>4.0</td>
<td>6.91</td>
<td>6.25</td>
<td>33.75</td>
<td>6.93</td>
<td>5.42</td>
<td>34.40</td>
<td>6.88</td>
<td>3.54</td>
<td>34.30</td>
</tr>
<tr>
<td>6.0</td>
<td>11.49</td>
<td>4.51</td>
<td>32.96</td>
<td>11.59</td>
<td>4.09</td>
<td>33.53</td>
<td>11.53</td>
<td>2.70</td>
<td>33.45</td>
</tr>
<tr>
<td>8.0</td>
<td>16.26</td>
<td>3.64</td>
<td>32.25</td>
<td>16.49</td>
<td>3.43</td>
<td>32.72</td>
<td>16.43</td>
<td>2.28</td>
<td>32.65</td>
</tr>
<tr>
<td>10.0</td>
<td>20.83</td>
<td>3.15</td>
<td>31.50</td>
<td>21.09</td>
<td>3.12</td>
<td>31.88</td>
<td>20.96</td>
<td>2.06</td>
<td>31.83</td>
</tr>
<tr>
<td>15.0</td>
<td>32.47</td>
<td>2.46</td>
<td>30.07</td>
<td>32.78</td>
<td>2.75</td>
<td>30.19</td>
<td>32.46</td>
<td>1.76</td>
<td>30.16</td>
</tr>
<tr>
<td>20.0</td>
<td>45.04</td>
<td>2.14</td>
<td>28.64</td>
<td>45.04</td>
<td>2.50</td>
<td>28.76</td>
<td>44.69</td>
<td>1.59</td>
<td>28.74</td>
</tr>
</tbody>
</table>
Figure 5.19 Graph showing encoding time vs. PSNR for Peppers by different methods
Figure 5.20 Graph showing compression ratio vs. PSNR for Peppers by different methods.
5.5 INFERENCES

Based on the experiments conducted on standard gray scale images Lenna, baboon, Goldhill and Peppers (512x512, 8 bit) with varying textures [77], the following inferences are drawn.

1) The proposed method resulted in fast encoding times on all images, than the average variance method. The compression ratio and PSNR obtained are nearly equal to the average variance method. A speed up factor of 1.88 is obtained on compressed Lenna image, at a compression ratio of 4.36 and PSNR of 35.97. The drop in PSNR is 0.08 dB.

2) The subjective fidelity of the decoded image is fairly good without any artifacts or blocky effects at low compression rates.

3) 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.

4) 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 for 8 to 10 iterations. After 10 iterations, the PSNR of decoded image has fairly remained constant.

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

In this chapter, the proposed gray level difference method for speeding up fractal image compression is described. The algorithms for compression and decompression are presented. Results of the experiments conducted on various gray scale images of size 512x512 (8-bit) are discussed and compared with other related methods. It is inferred that the proposed method results in reduced encoding times when compared to other related methods. The fidelity and compression ratio are nearly equal to those obtained by the classified search method.

In the next chapter, application of the proposed fractal image compression techniques for satellite and medical scan images is described.