CHAPTER III
SEARCH SPACE REDUCTION BY REDUNDANT BLOCK REMOVAL STRATEGY (RBRS)

3.1 Introduction

The sheer volume of digital images, represented as a two dimensional function of pixel values can be alarming due to its memory requirement. In fact, the amount of data generated from digital images may be so huge that it may give rise to impractical storage, processing, and communication requirements. In such cases, representation of images will be referred to as image data. The basic feature of an image compression problem is the removal of redundant information present in the image. Three basic redundancies, i.e. coding redundancy, interpixel redundancy and psychovisual redundancy are exploited in image compression problems.

Whenever the term compression is used, it requires the quantitative estimation. The two commonly used measures are the compression ratio and data redundancy. If $N_o$ is the size of the original file, $N_c$ is the size of the compressed file then the compression ratio $C_R$ is $C_R = N_o / N_c$ and redundancy $R_D$ is $R_D = (1 - 1/ C_R ) \times 100\%$. When, clearly, $R_D$ is 100% for $N_c < N_o$, it indicates good compression. Besides the removal of redundant information (encoding), an image compression technique consists of retrieval of the original image information or an approximation of it (decoding). The RBRS mechanism proposed here comes under the category of coding redundancy.

The main aspect of fractal coding is to find a suitable technique for minimizing the search space for compression operation. A new approach is proposed to increase the performance of a Normal IFS algorithm which is used to reduce the search space through RBRS mechanism. Before employing this technique, a quadtree decomposition strategy is used for partitioning the image into blocks of various sizes for fractal coding. Then
implementing RBRS for eliminating redundant blocks in the original input image has been performed.

The basic idea of an IFS is to create a finite set of contraction mappings [56] written as affine transformations [54] based on the desired image. If these mappings are contractive, then applying the IFS to a seed image will eventually produce an attractor of that mapping which looks like the original image. The same transformation will be produced regardless of the seed image (domain blocks) for the mappings. This technique is known as local Iterated Function System.

Another important aspect of fractal image compression is to find subspaces (or sub-images) of the original image space that can be regenerated by using IFS. There is a possibility of using one IFS instead of several IFS's that reproduce similar sub-images. It is more efficient in terms of the storage space used for finding those IFS, and an image will require more than one IFS to reproduce a compressed image which resembles the original.

Consider an image space I in which a typical constant block based method is used where ‘D’ domain blocks are compared with ‘R’ range blocks in the original image I. It means that ‘D’ x ‘R’ block matching operations are to be performed. The total search space will be higher than that of the size of the original image since the domain block is a reduced version of the original image. So the total search space can be represented as:

\[ S = \{R_1, R_2, R_3...R_m\} + \{D_1, D_2, D_3...D_n\}. \]

Since the original image space I = \{R_1, R_2, R_3...R_m\}

Here, ‘m’ represents the total number of range blocks and ‘n’ represents the total number of domain blocks. If the transformations and intensity scaling are taken into account then the whole process is to be repeated ‘t’ times. In this way, \( m \times n \times t \) block matching operations are performed. This will take a lot of time and increases the search space and time rapidly depending on the number of transformations ‘t’ used.
3.1.1 MATLAB used

The implementation of the idea proposed in this work is performed using MATLAB 6.5 [93][114] since it has several features and toolkits for image processing operations. It is an Interpreter software containing several built-in functions and libraries to perform different kinds of operations related to image processing applications.

3.2 The Proposed Search Space Reduction Method

Iterated Function Systems are the basic idea of fractal image compression and Yuval Fisher’s Normal IFS algorithm for fractal coding is used. In Normal IFS algorithm, the constant block decomposition strategy was used to partition the image. But, a quadtree decomposition strategy is used to partition the image of different block sizes based on its features in the proposed algorithm. The blocks that are smaller than a predefined threshold value are considered as domain blocks. The rest of the blocks are considered as range blocks. Here, the total search space is within the size of the original image since the domain blocks and range blocks are taken inside the same image. This will reduce the search space significantly.

Another main enhancement is to remove the redundant blocks before the block matching operation begins. The redundant blocks are nothing but the exact image blocks. A suitable mathematical measure like RMSE is used to check the similarities of the image blocks. This RBRS mechanism will reduce the overall search time significantly [126][127] while block-matching operation is performed. In the Yuval Fisher’s Normal IFS algorithm, the redundant blocks are kept as they are during the block matching operation. Because of this reason the existing Normal IFS algorithm took more time for the compression operation.

In this section, the search space reduction mechanism is explained with suitable diagram. The figure 3.1 illustrates the block-matching scenario in a constant block size.
method. Generally, the constant block size method is used in an IFS coding of Normal IFS algorithm. Normally in the IFS coding process, a lot of computations are performed during the block matching operation. Hence, it leads to more computation time during fractal IFS coding process.

In a constant block size based algorithm, the blocks in the original image are mapped with the blocks in another reduced version of the same image. The search space is increased since N x N blocks are compared and this will lead to more computation time resulting in poor performance. To overcome this disadvantage, variable block size method is used in the proposed Normal IFS algorithm. Even though many techniques are available, a quadtree decomposition method is considered in this work for the image decomposition.

In the proposed algorithm to reduce the search space, exact redundant blocks are removed by leaving one copy intact. For finding the similar blocks, a simple block matching operation such as RMSE [31] measure is used. Since the RBRS mechanism consumes very low time compared to the overall time for finding IFS coding, this method is used in the implementation of all proposed algorithms of this research. Figure 3.2 illustrates the scenario of the original Face image of the size 128 x 128. This image is

FIGURE 3.1 Face Image for Constant Block Matching operation.
considered as a standard one next to Lena image for the image processing operations. The working procedure of the Normal IFS algorithm is as follows.

![Figure 3.2 Original Face Image (128 X 128)](image)

The redundant blocks with exact copy of contents are removed after leaving one block. Later, the blocks are classified into domain blocks and range blocks which will be suitable for performing the block matching operation. Figure 3.3 illustrates the block matching operation performed between range and domain blocks [73].

The basic task is to find an appropriate range block for each domain block. The information gained for the suitable range block is used to generate fractal code during compression. The information for each domain block and the entire image is enough to reproduce an approximation of the given image. The whole problem of finding appropriate range blocks can be viewed as a search problem. This idea is implemented and tested with sample images of varying details successfully.

The original Face image in figure 3.2 is partitioned into various image blocks based on its contents using quadtree decomposition strategy. This method divides the Face image into ‘T’ blocks of different sizes. After removing X redundant blocks, only T-X blocks are available in total. Then the available T-X blocks are called ‘Rm’ range
blocks and X redundant blocks are called as 'Dn' domain blocks. So the search space is reduced significantly for the block matching operation. Now it can be represented with the search space as $S = I - \{X_i\}$. Where I is the original image space and $X_i$ is the removed image space. Since $X_i$ is less than the original image space I, the proposed method will converge to a solution in lesser time when compared to the other methods theoretically. The compression ratio calculated thus is better and the improvement in reduction of compression time is the interesting factor of this method.

![Figure 3.3 Block Matching scenario of Face Image.](image)

**Remark 3.1:** In both the figures, the blocks are in different sizes but not marked with any lines of reference. Refer the original image marked with lines in Figure 3.1.

In Figure 3.3 the areas which are used as seed blocks and the areas reduced from the seed block search space are indicated. The distance measure Root Mean Square Error (RMSE) is used to check the similarity for the block matching operation.
3.2.1 The Procedure for the Quadtree decomposition and RBRS strategy:

Step 1: Represent the image $A$ mathematically as:

$$ A = \sum_{i}^{n} \sum_{j}^{n} A_{ij} $$

The summation of all the pixels of the image can be written as:

$$ A = (A_i) $$

The numerical representation of the pixels of the same image be represented as:

$$ A = (a_{ij}) $$

Step 2: Convert the image into a vector. That is $A_{vec} = X_n^2$.

Step 3: Arrange $X_n^2$ into non-decreasing order and denote it as $X_o^2$.

That is, $X_o(1) \leq X_o(2) \leq X_o(3) \leq \ldots \leq X_o(n^2)$

Step 4: Eliminate the redundant blocks in $X_o^2$ and denote it by $X_o^{vec,K1}$.

where $K_1 \leq n^2$

That is $X_o(1) < X_o(2) < X_o(3) < \ldots < X_o(K_1)$

Step 5: Assume for every $X_o(i) = A$ and proceed the steps 1 to 4 for all distinct $X_o(i)$.

Step 6: Repeat steps 1 to 5 until the threshold value is reached.

Remarks 3.2: The above model is general in nature, i.e. every time the image has been read by $n \times n$ pixels.

If $n = 2$, then it becomes a quadtree approach.

3.3 Block Matching scenario

In this implementation, the range blocks are compared with domain blocks for finding a match and the indexes of identical blocks are stored. To find the best-matching domain block, one has to search all possible domain blocks with the help of distance.
measure formula. This can also be achieved by using any one of the following eight transformations on the domain blocks.

1. Rotation about $0^0$
2. Rotation about $90^0$
3. Rotation about $180^0$
4. Rotation about $270^0$
5. Reflection about mid-horizontal line
6. Reflection about mid-vertical line
7. Reflection about principal diagonal line
8. Reflection about principal secondary diagonal line.

These eight transformations are all isometric transformations since they preserve distances. This is an estimation of pixel values of a range block from the pixel values of a transformed domain block [2]. This process is repeated for all the range blocks and the storage is minimized in IFS coding [3]. Here, the working procedure of the algorithm is also presented to understand the whole IFS coding generation process.

The set of affine transformations along with the selected domain block location provides a complete representation of the original image. A great amount of reduction in the memory is achieved by storing the parameters of the affine transformations [78] and the locations of the domain blocks, instead of an image. The affine contractive transformation $f_i$ is constructed based on the fact that the gray values of the domain block are a scaled, translated, and rotated version of the gray values of range block. Also $f_i$ can be separated into two parts: one for spatial information and the other for information of gray values. In the process of reconstruction of the original image, some loss in the image information may occur. Using appropriate decompression technique, the image can be reproduced.
3.4 Implementation of Normal IFS Coding Algorithm

The encoding process is computationally intensive. Millions or billions of iterations are required to find the fractal patterns in an image. Depending upon the resolution, contents of the input bitmap data, output quality, compression time, and file size parameters selected, compressing a single image could be between few seconds and few hours (or more) even on a very fast computer. Decoding a fractal image is a much simpler process. The hard work was performed in finding all the fractals during the encoding process. All the decoding processes should interpret the fractal codes and translate them into a bitmap image.

The working procedure of the algorithm is explained in various steps with a sample image. The Yuval Fisher’s Normal IFS algorithm is experimented using the principles of fractal image compression [90]. The results are significant because of the introduction of RBRS strategy.

[1] Take an input image ‘I’ of particular size and crop it as a square image which is suitable for quadtree decomposition as shown in figure 3.4.

![FIGURE 3.4 Input Face Image](image.png)

[2] Decompose the given input image into a number of non-overlapping blocks of variable size based on its similarity in contents using quadtree decomposition method as shown in figure 3.5.
FIGURE 3.5 Face Image after decomposition.

[3] Remove redundant blocks from the original image by leaving one seed block (domain block) intact as a seed block of size smaller than a particular threshold value is to be used for coding the image as shown in figure 3.6.

FIGURE 3.6 Face Image after applying RBRS strategy

[4] Classify the image as ‘D’ domain blocks (seeds) and ‘R’ range blocks of various sizes.

[5] Collect the local contractive affine transformations mapping of domain block $D_i$ to the range block $R_i$.

[6] Find, a corresponding match in range blocks using RMSE as the distance measure for the computation of IFS coding for each domain block.


Figure 3.7 is the decompressed image of the corresponding IFS code [65]. It is of good quality and suitable for further image processing operation. The novel RBRS mechanism is used in all proposed algorithms for better performance in this research. The test results indicate that the performance of a proposed algorithm is good for all sample.
images chosen for the compression operation. The compression ratio presented in Table 3.3 indicates almost 50% reduction in size and significant increase in the compression speed.

![Decompressed Face Image after applying Normal IFS Algorithm](image)

**FIGURE 3.7** Decompressed Face Image after applying Normal IFS Algorithm

### 3.5 Experimental Results

The implementation of all proposed algorithms in this work has been tested on images of different size with varying details using MATLAB in a Laptop, Intel Celeron M 340 Processor 1.5 GHz, with 256 MB RAM, 400 MHz FSB, 512 KB L2 cache. To experiment the Normal IFS algorithm, the images Face1, Face2, Face3, Lena and Nature have been dealt in this chapter. Moreover, the same set of low-resolution images of the sizes 128 x 128 and 256 x 256 have been taken for the implementation of all proposed algorithms. The study has been conducted on high-resolution images of size 1024 x 1024 also and the details of result are presented in chapters three, four and five of this thesis. Simultaneously, the computation result in the form of error image is also displayed.

![Various Input Images](image)

**FIGURE 3.8** various input images taken for the compression operation
The summary of results for various images of the size 128 x 128 using Normal IFS algorithm with RBRS mechanism is given in Table 3.1. The main aspect of this IFS coding is the time taken by various images in different stages and is presented in Table 3.1.

### TABLE 3.1
EXPERIMENTAL RESULTS IN TERMS OF TIME AFTER APPLYING RBRS MECHANISM FOR 128 x 128 SIZE IMAGES FOR NORMAL IFS

<table>
<thead>
<tr>
<th>Image</th>
<th>Range Blocks</th>
<th>Domain Blocks (4x4)</th>
<th>Time (in Sec.)</th>
<th>Comp. ratio</th>
<th>PSNR In db</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Quad tree</td>
<td>Redundant Block Removal</td>
<td>IFS's coding</td>
</tr>
<tr>
<td>Face1</td>
<td>733</td>
<td>184</td>
<td>0.01 0.20</td>
<td>66.49</td>
<td>66.70</td>
</tr>
<tr>
<td>Face2</td>
<td>811</td>
<td>283</td>
<td>0.02 0.24</td>
<td>100.89</td>
<td>101.15</td>
</tr>
<tr>
<td>Face3</td>
<td>745</td>
<td>174</td>
<td>0.01 0.21</td>
<td>66.84</td>
<td>67.06</td>
</tr>
<tr>
<td>Lena</td>
<td>943</td>
<td>310</td>
<td>0.02 0.23</td>
<td>131.35</td>
<td>131.60</td>
</tr>
<tr>
<td>Nature</td>
<td>631</td>
<td>133</td>
<td>0.18 0.201</td>
<td>43.66</td>
<td>44.04</td>
</tr>
</tbody>
</table>

### TABLE 3.2
EXPERIMENTAL RESULTS IN TERMS OF TIME WITHOUT APPLYING RBRS MECHANISM FOR 128 x 128 SIZE IMAGES FOR NORMAL IFS

<table>
<thead>
<tr>
<th>Image</th>
<th>Total Pixels</th>
<th>Pixels used for Coding</th>
<th>Range Blocks</th>
<th>Domain Blocks (4x4)</th>
<th>Time (in Sec.)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Quad tree</td>
<td>IFS's coding</td>
</tr>
<tr>
<td>Face1</td>
<td>16384</td>
<td>2944</td>
<td>733</td>
<td>184</td>
<td>0.01</td>
<td>110.01</td>
</tr>
<tr>
<td>Face2</td>
<td>16384</td>
<td>4528</td>
<td>811</td>
<td>283</td>
<td>0.02</td>
<td>129.51</td>
</tr>
<tr>
<td>Face3</td>
<td>16384</td>
<td>2784</td>
<td>745</td>
<td>174</td>
<td>0.01</td>
<td>112.24</td>
</tr>
<tr>
<td>Lena</td>
<td>16384</td>
<td>4960</td>
<td>943</td>
<td>310</td>
<td>0.02</td>
<td>115.47</td>
</tr>
<tr>
<td>Nature</td>
<td>16384</td>
<td>2128</td>
<td>631</td>
<td>133</td>
<td>0.18</td>
<td>60.23</td>
</tr>
</tbody>
</table>
The time is calculated separately for the image decomposition using quadtree method, the RBRS mechanism adopted, and for the generation of the new IFS code. To understand the significance of RBRS mechanism, another experiment is done on Normal IFS without RBRS. The time taken during this implementation for the various images is computed and the results are presented in Table 3.2.

The encoding time taken by the Normal IFS algorithm without RBRS mechanism is doubled. Further, the compression ratio and quality of a decoded image in PSNR are calculated and the results are presented in Table 3.1. The comparative analysis of Normal IFS algorithm with RBRS and Normal IFS algorithm without RBRS are shown in Graph 3.1. As specified in Graph 3.1, the Normal IFS with RBRS mechanism provides better result and it can be treated as a novel approach in the field of fractal image compression. In Tables 3.1 and 3.2 the time consumed by Normal IFS algorithm for quadtree decomposition, redundant block removal, and IFS coding are reported separately. The sum of all these values is calculated as total time.
A significant contribution to the fractal image compression is carried out in this work by suggesting computationally efficient algorithms. By adapting the RBRS mechanism for reducing the search space, the problem of expensive computational cost is met. To judge the quality of the decoded image, Peak Signal-to-Noise Ratio (PSNR) has been used. The consequences of the efficiency of compression can be seen quantitatively by observing the decoded image quality measured in PSNR. In addition, the implementation of Normal IFS algorithm with RBRS and without RBRS mechanism is presented here.

To analyze more and test the suitability of the Normal IFS algorithm, the 256 x 256 size images of different types have been chosen and experimented. Calculating the size of the total Domain pool can focus the importance of categorization of blocks. Suppose the image is partitioned into 4 x 4 domain blocks, a 256 x 256 size image will have 4096 domain blocks, which makes for a maximum of 167,77,216 possible pairings to test in a Normal IFS. Increasing the search speed is the main challenge faced by fractal image compression. The proposed RBRS mechanism, which removes redundant blocks to minimize the search space requires a number of comparisons to 1,34,872 approximately. The obtained results are presented in Table 3.3 for the 256 x 256 size image.

**TABLE 3.3**

EXPERIMENTAL RESULTS IN TERMS OF TIME AFTER APPLYING RBRS MECHANISM FOR 256 x 256 SIZE IMAGES

<table>
<thead>
<tr>
<th>Image</th>
<th>Total Pixels</th>
<th>Pixels used for Coding</th>
<th>Range Blocks</th>
<th>Domain Blocks (4x4)</th>
<th>Time (in Sec.)</th>
<th>Quad tree</th>
<th>Duplicate Block Removal</th>
<th>IFS's coding</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face1</td>
<td>65536</td>
<td>10048</td>
<td>733</td>
<td>184</td>
<td>0.04</td>
<td>0.412</td>
<td>566.49</td>
<td>566.94</td>
<td></td>
</tr>
<tr>
<td>Face2</td>
<td>65536</td>
<td>15488</td>
<td>811</td>
<td>283</td>
<td>0.08</td>
<td>0.492</td>
<td>700.89</td>
<td>701.46</td>
<td></td>
</tr>
<tr>
<td>Face3</td>
<td>65536</td>
<td>10800</td>
<td>745</td>
<td>174</td>
<td>0.04</td>
<td>0.441</td>
<td>586.47</td>
<td>586.95</td>
<td></td>
</tr>
<tr>
<td>Lena</td>
<td>65536</td>
<td>15632</td>
<td>943</td>
<td>310</td>
<td>0.09</td>
<td>0.474</td>
<td>731.35</td>
<td>731.91</td>
<td></td>
</tr>
<tr>
<td>Nature</td>
<td>65536</td>
<td>11600</td>
<td>631</td>
<td>133</td>
<td>0.12</td>
<td>0.610</td>
<td>643.60</td>
<td>644.94</td>
<td></td>
</tr>
</tbody>
</table>
The figure 3.9, illustrates the experimentation done by using Normal IFS algorithm with RBRS mechanism for different types of images. The proposed idea is implemented in four steps to understand the behavior of the algorithm better. In Figure 3.9, in each line, the first picture is the original input image taken for the compression operation, the second picture is the image after applying quadtree decomposition strategy, the third picture is the image after applying RBRS mechanism and the fourth one is the reconstructed image after applying the decompression operation. There are four distinct stages followed in the implementation of the Normal IFS algorithm.

FIGURE 3.9 Sample Images of various stages in Normal IFS Algorithm
3.6 Results of High Resolution Images

The experimentation has been done so far on low-resolution images to measure the performance of the Normal IFS algorithm. The RBRS mechanism used here makes necessary search space reduction in the encoding process of the algorithm. The problem with fractal coding is the high computational complexity in its encoding process. Most of the encoding time is spent on finding the best-matched domain block from a large domain pool to represent an input range block. By minimizing the difference between the range block and the transformed domain block, the corresponding iterated contractive transformation can be found for the range block. Hence, it is decided to conduct the test on the high-resolution images to study the performance.

The Normal IFS algorithm discussed in this chapter is implemented on 1024 x 1024, 8 bits/pixel 'Lena', 'Beach', 'Pepper' and 'Concord aerial' images. In order to handle the encoding complexity, the RBRS mechanism is used and this will certainly reduce the search space considerably. The next important mechanism used for reducing this complexity is by using variable block size decomposition of the image rather than using the constant block size strategy. The other way to reduce the complexity is by using the minimum number of transformation for block matching operation. The essence of the compression process is the pairing of each domain block to a range block such that the difference between the two under an affine transformation is minimal.

In fact, there is nothing that says the blocks have to be squares or even rectangles. That is, just an imposition is made to keep the problem tractable. Generally, the method of finding good image blocks for any given input image involves five main issues:

1. Partitioning the image into range blocks.
2. Forming the set of domain blocks.
3. Choosing the type of transformations.
4. Selecting a distance metric between blocks.
5. Specifying a method for pairing range blocks to domain blocks.
The greatest irony of the coding community is measuring and quantifying the error present in a decompressed image and great effort is taken toward minimizing an error measure that most often gives a dubious value. These measures include root mean square error and signal-to-noise ratio. With respect to those dubious error measures, the results of tests reveal that for low compression ratios JPEG like method is better and for high compression ratios fractal encoding is better. The softwares that create JPEG images such as paint programs allows the user to set a "quality factor" to specify the percentage level of degradation. This quality factor is a more or less arbitrary scale and may not give exactly the same results in different software packages. But as a rule a 75% quality factor gives a perfectly acceptable image while quality factors above 95% give very little improvement in the image and quality factors below 10% produce a completely unacceptable image.

When the level of information loss can be expressed as a function of the original or input image, the compressed and subsequently decompressed output image, it is based on an objective fidelity criterion. A good example is the root-mean-square (rms) error between an input and output image. Let \( f(x,y) \) represent an input image and \( f^*(x,y) \) denote an estimate or approximation of \( f(x,y) \) that results from compressing subsequently decompressing the input. For any value of \( x \) and \( y \), the error \( e(x,y) \) between \( f(x,y) \) and \( f^*(x,y) \) can be defined as

\[
e(x,y) = f^*(x,y) - f(x,y)
\]

So that the total error between the two images is

\[
\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f^*(x,y) - f(x,y)]
\]

where the images are of size \( M \times N \). The root-mean-square error, \( e_{rms} \), between \( f(x,y) \) and \( f^*(x,y) \) then is the square root of the squared error averaged over the \( M \times N \) array, or

\[
e_{rms} = \sqrt{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f^*(x,y) - f(x,y)]^2}
\]
Although objective fidelity criteria offer a simple and convenient mechanism for evaluating information loss, most decompressed images are viewed by humans. Consequently, measuring image quality by the subjective evaluations of a human observer often is more appropriate. This can be accomplished by showing a typical decompressed image to an appropriate cross section of viewers and averaging their evaluations. As the \( \text{rms} \) error increases, the amount of difference between the original and compressed images also increases.

The algorithm has been tested on the above-mentioned four images of size 1024 x 1024. The experimentation on high-resolution images have been performed on Intel Pentium IV Processor, 2.66 GHz, 512 MB RAM, 533 MHz FSB, 1 MB L2 cache, 80 GB HDD machine for Normal IFS method and algorithms proposed in chapter four and five. Figures 3.10, 3.11, 3.12 and 3.13 show the rescaled original, decoded, and error images for all four-sample images. To judge the quality of the decoded image Peak-to Signal-to-Noise-ratio (PSNR) has been measured and presented in Table 3.4 along with compression time and ratio. It is very clear from the test results that the compression ratio achieved and the PSNR values computed are good. The compression time mentioned in [24] for Lena image of size 512 X 512 is reduced considerably as shown in Table 3.4. The obtained results are significant in the sense the RBRS strategy considerably reduced storage space.

To manage the encoding complexity in handling high-resolution images when the RBRS mechanism is followed, threshold value is increased so that more blocks can be removed. The execution results of the images are presented in three forms such as original image, decoded image, and error image. The snapshots of the software execution screens are given in the annexure separately for all the four images. In the error image,
the black regions are the good reconstruction area, and white and gray areas are the poor reconstruction area shown for all the four images in Figure 3.10, 3.11, 3.12 and 3.13.

TABLE 3.4
EXPERIMENTAL RESULTS OF HIGHER RESOLUTION IMAGES OF SIZE 1024 X 1024 IN NORMAL IFS ALGORITHM

<table>
<thead>
<tr>
<th>Image</th>
<th>Range Blocks</th>
<th>Domain Blocks</th>
<th>Compression ratio</th>
<th>IFS coding Time in Sec.</th>
<th>PSNR in db</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>24456</td>
<td>7248</td>
<td>9.2</td>
<td>1875.81</td>
<td>30.59</td>
</tr>
<tr>
<td>Beach</td>
<td>36542</td>
<td>8496</td>
<td>9.3</td>
<td>2010.34</td>
<td>31.30</td>
</tr>
<tr>
<td>Pepper</td>
<td>39604</td>
<td>7878</td>
<td>13.9</td>
<td>2165.80</td>
<td>33.60</td>
</tr>
<tr>
<td>Concord</td>
<td>43893</td>
<td>9862</td>
<td>6.7</td>
<td>2307.62</td>
<td>30.72</td>
</tr>
</tbody>
</table>

In the experiments, the performance of a Normal IFS algorithm with RBRS strategy has been explored. In a paper, reported by Cheung-Ming Lai et al. [24] Normal IFS algorithm combined with DCT for Lena image of size 512 X 512 consumes 1342 seconds for encoding and PSNR value 31.14 db for decoded image quality. In the proposed Normal IFS algorithm for the same Lena image of size 1024 X 1024 the encoding time is 1875 seconds and the decoded image quality is 30.59 db. Similarly, this algorithm is compared with the performance reported in adaptive search algorithm [29] and the tree search algorithm in [11] and the results of this algorithm found good. Experimental results show that the Normal IFS algorithm provides slightly better performance in terms of compression time than the existing algorithms mentioned in [11][24][29] due to the efficiency of RBRS strategy. The decoded image quality measured in PSNR (db) as 30.59 db is a good performance in fractal image compression.
Figure 3.10 Lena Image of various stages
Figure 3.11 Beach image of various stages
Figure 3.12 Pepper image of various stages
Figure 3.13 Concord image of various stages
3.7 Summary

A novel block selection, block removal and classification strategy to reduce the overall search time for IFS coding in fractal image compression have been experimented. The central contribution is a new optimized IFS algorithm that extends ideas from statistical and numerical pattern search [163]. The novel RBRS mechanism is the achievement in this work that reduces the search space obviously. This is considered as one of the main reasons for achieving improved performance of the same Normal IFS algorithm. Other areas that can be explored in future are improving both image quality and processing speed which includes trying different transformations of the blocks other than the affine transformation.