Chapter 5: Difference based Non-linear and randomized fractal image compression

5.1: Introduction
The major application of compressing images lies in efficient storage and transmission. Good results have been achieved with fractal image compression in the recent past. Barnsley [72] promoted Iterated Function System (IFS) which forms a key technique on which many fractal compression techniques rely on. An iterated function system is a collection of contractive affine transformations. A fundamental theorem in fractal geometry is that each IFS, that is each set of contractive affine transformations, defines a unique image or what is called fractal. The Fractal image compression technique is the inverse problem. Instead of generating an image from a given formula, the aim in fractal image compression is to find a set of transformations that can represent a given image.

The basic implementation of the IFS compression method has split the image into B x B blocks in the image as the set of range blocks, and all (overlapping) 2B x 2B blocks in the image as the set of domain blocks [73]. The set of transformations applied consists of a spatial contraction, followed by one of the eight square symmetry operations (4 rotations and 4 reflections) followed by a contractive affine transformation on the grayscale value. The domain block is first brought down to the size of the range blocks. Each domain pixel is then multiplied by the scaling factor S and an offset O is added to it to get the corresponding range pixel. The following equation represents the transformation of domain pixels to range pixels.

\[ R_i = S \times D_i + O \]  
(Equation 5.1)

Where \(-1 < S < 1\) guarantees contraction. In the above equation, \(R_i\) represents the range pixels and \(D_i\) represents the domain pixels. Here \(S\) and \(O\) are so chosen that the RMS error between the domain and the range is a minimum. Search is made on the whole domain pool to identify the closest domain for each range. The union of mappings defines the image.

Efforts have been made to reduce the search time for the closest domain by reducing the domain space based on polynomial approximation [65]. The decompression algorithm is
improved by taking an initial seed value [69] other than arbitrary values or zero. If the closeness of the range and domain is not within a limit, quad-tree splitting can be made to reduce the size of range and domains [71] and increase accuracy at the cost of size of compressed image.

In this thesis, we discuss about construction of transforms based on the difference between the range and domain pixels. Linear fractal compression has proved been significantly able to reduce the size of images. We intend to present our findings on taking a non-linear approach in hopes to optimize upon compression. This thesis further discusses the results comprising of a combination of difference based approach and a non-linear approach combined as one to create a compressed image. Attempts have been made to implement non-linear models of image compression using complicated transforms, Dan C. Popescu[67], we have used simple function, power for non-linearization.

5.2: PIFS METHOD OF FRACTAL IMAGE COMPRESSION

In this section, we will first briefly discuss the method of PIFS [67] compression and decompression. We then investigate the methods of compression using the difference method.

5.2.1: Encoding procedure

The following steps describe the encoding procedure:

1. Partitioning of the original image into N non-overlapping range blocks.
2. Splitting of the image into M (possibly overlapping) domain blocks.

The following procedure is repeated for all range blocks.

1. Choose a range block Ri.
2. From all combinations of transformations for domain blocks, based on (equation 5.1), choose the best transformation which minimizes the RMS distance between the range block and domain.

When the best pair has been found, only the transform detail for each range is stored. This transformation contains information about the positional description of the domain block Dj associated with a given range, the number of rotation operation, scaling (S) and offset (O) parameters.
5.2.2: Decoding procedure

The decoding steps are as follows:

- An initial image $X$ is chosen at random (usually a uniform gray image). A transformed image, is created from the transformation as follows:

---

**Psuedocode 5.1: Flowchart for PIFS based fractal image compression**

Partition the image into $N$ non-overlapping range blocks

Partition image into $M$ (possibly overlapping) domain blocks

Choose next range block $R_i$
Among all domains, choose the best transformation which minimizes the RMS distance between the range block and domain. Use equation 5.1.

Store the transform detail for each range. Description of the domain block $D_j$ associated with a given range, the number of rotation operation, scaling($S$) and offset ($O$) parameters. The loop is run for all ranges one after the other.
To get a range block, apply the transformation on its corresponding domain. The domain number is stored in the transformation.

When all range blocks are exhausted, the resulting image will contain the transformed version of the starting image.

In the next step we will transform the resulting image again starting from step 2.

Due to the contractive nature of the mappings, the resulting image will converge towards a final image after a few iterations (Typically 9 iterations are sufficient).

Psuedocode 5.2: Flowchart for PIFS based fractal image decompression
5.3: DIFFERENCE METHOD

This method relies on applying transforms on the difference between the range and the domain pixel values.

5.3.1: Transforms
The set of transforms are of the form

\[ R_i = D_i + S \times (R_i - D_i) + O \]  \hspace{1cm} (Equation 5.2)

Where \( R_i \) is a range pixel and \( D_i \) is a domain pixel. Here \( S \) is a scaling factor between 0 and 1 and \( O \) the offset.

The values stored for each transform are
- Scaling factor which scales the difference between the domain pixel and range pixel.
- Offset which minimizes the RMS error between the range block and domain block.
- The domain number.

5.3.2: Encoding Procedure

The encoding procedure is as mentioned in 5.2.1.

The image is decoded starting with a seed value for a certain number of iterations. The optimal seed value, number of iterations and the set of transformations which has least RMS difference with the original image is stored in the encoded image. The image does not contract to a single image but keeps changing based on different seed values and the number of iterations. The size of the compressed image is two bytes more than the one for PIFS method. One byte has the seed value and the other byte is used for the number of iterations.

5.3.3: Decoding Procedure

1. Initial image S has seed value.
2. Apply the stored transformation to each range block.
3. Repeat the above step 2 as many times as the number of iterations.

Though the Compressed image (Figure 5.2 and figure 5.4) appears slightly blurrier than the original image (Figure 5.1 and figure 5.3), it still gives the overall details present in the image.

*Figure 5.1: Original Image*

*Figure 5.2: Compressed Image (difference method)*

*Figure 5.3: Original image*
5.4: NON-LINEAR METHOD

Generally, fractal compressions give a good result to linear compression. However mapping of sound has shown that non-linear approaches may be better suited to some methods than linear methods. Based on the type of the image, certain images may be best suited for non-linear compression.

5.4.1: Transforms
The set of transforms are of the form

\[ R_i = D_i^p + O \]  \hspace{1cm} (Equation 5.3)

Where \( D_i \) is a domain pixel and \( R_i \) is a range pixel. Here, \( O \) is the offset and \( p \) is a real number between 0 and 1.

The values stored for each transform are

- The domain number \( D \) from the pool of domains which reduces the RMS error from the range after applying transformation in (equation 4.3).
- Power \( p \) in the range between 0 and 1. This helps in the convergence of the image.
- Offset \( O \) which minimizes the RMS error between the range block and domain block.

5.4.2: Encoding Procedure

The encoding procedure is as mentioned in 5.2.1.
5.4.3: Decoding Procedure

The decoding procedure is as follows.
1. Initial image $S$ has seed value of 0.
2. Apply stored transformation to each range block.
3. Repeat step 2 till the image converges. The image converges because of $p$ being a value between 0 and 1. $p$ is a constant 0.75 in all our algorithms.

Figure 5.5 and figure 5.6 below show the compressed and decompressed images using this technique.

![Figure 5.5: Compressed Image (Non-linear method)](image)

![Figure 5.6: Compressed Image (Non linear method)](image)

5.5: NON-LINEAR DIFFERENCE BASED METHOD

A combination of difference based and non-linear approach may lead to a better image quality in some types of images. The transform applied is based on the power of the difference between the domain and the range pixels. Care has to be taken while raising negative values to power less than 1.
5.5.1: Transforms

The set of transforms are of the form

\[ R_i = D_i + (R_i - D_i)^p + O \]  

(Equation 5.4)

Where \( R_i \) is a range pixel and \( D_i \) is a domain pixel. Here \( p \) is the power factor which is a fraction between 0 and 1 and \( O \) the offset.

The values stored for each transform are

- The domain number \( D \) from the pool of domains which reduces the RMS error from the range after applying transformation in (equation 5.4).
- Power \( p \) in the range between 0 and 1 (fixed to 0.75 in our algorithm). This helps in the convergence of the image.
- Offset \( O \) which minimizes the RMS error between the range block and domain block.

The compressed image also stores

- Initial seed value.
- Number of iterations.

5.5.2: Encoding Procedure

The encoding procedure is as mentioned in 5.2.1.

The image is decoded starting with a seed value for a certain number of iterations. The optimal seed value, number of iterations and the set of transformations which has least RMS difference with the original image is stored in the encoded image. The image does not contract to a single image but keeps changing based on different seed values and the number of iterations. The size of the compressed image is two bytes more than the PIFS method.

5.5.3: Decoding Procedure
1. Initial image $S$ has seed value stored in the compressed image.
2. Apply the stored transformation to each range block.
3. Repeat above step 2 as many times as the number of iterations.

Figure 5.7 relates to the compressed and decompressed image following the above technique.

![Figure 5.7: Compressed Image (Non-linear difference method)](image)

5.6: Randomized PIFS method

We will discuss the usage of variable scaling factors for transforming the domain pixels to range pixels. In PIFS, the scaling factor $S$ (equation 5.1) is a constant for a range-domain pair. These variable scaling factors are generated by the use of a pseudo random generator.

In randomized PIFS method, we store image using a pseudo random seed value and offset for every transformation. If the size of a range is 16, then, using the seed, 16 pseudo random numbers are generated in the range 0 to 1. These 16 values are considered to be the scale values for multiplying the domain pixels. A fixed number of different seed values are considered and an optimal seed value is stored along with the offset for every transformation. Different domains are not searched for optimal values of scaling and offset. Instead, only the first domain is considered for all ranges. Hence, it is not required to store the domain number in every transformation. This reduces the size of the compressed images to 7.81 KB from 11.7 KB in other methods considered above.

5.6.1: Transforms
The set of transforms are of the form
\[ R_i = R_i + s_i \times D_i + O \]

(Equation 5.5)

Where \( R_i \) is the Range value of \( i^{th} \) pixel, \( D_i \) is the Domain value of \( i^{th} \) pixel, \( O \) is the offset value, \( s_i \) is the scale value that is a pseudo random number between 0 and 1 (\( 0 < s_i < 1 \)) for a particular seed.

The values stored for each transform are

1. Seed for pseudo random generator.
2. Offset which minimizes the RMS error between the range block and domain block.

5.6.2: Encoding Procedure

1. Partitioning of the original image into \( N \) non-overlapping range blocks \( \{R_i\}_{k=1}^N \).

Repeat the following steps for all range blocks.

1. Choose a range block.
2. For all possible seed values, choose a seed value that minimizes the RMS error between the domain block and range block.
3. Store the seed value, offset and symmetry for each transform.

5.6.3: Decoding Procedure

1. Create a random image.
2. Apply the transformation for each range block.
3. Repeat step 2 till the image converges. This normally happens in about 9 iterations.

Figure 5.8: Compressed Randomized PIFS Image
Figure 5.9: Compressed Randomized PIFS Image

Figure 5.10: Compressed Randomized PIFS Image

Figure 5.11: Original Image

Figure 5.10 seems to have a quality better than figure 5.8 and figure 5.9.

5.7: Randomized non-linear PIFS

In this case, we compress the image using a power factor instead of scaling factor for every transform. This makes the transforms non-linear.

5.7.1: Transform

The set of transforms are of the form

\[ R_i = D_i^{P_i} + O \]  \hspace{1cm} \text{(Equation 5.6)}

Where \( R_i \) is the Range value of \( i^{th} \) pixel, \( D_i \) is the Domain value of \( i^{th} \) pixel, \( O \) is the offset value, \( P_i \) is the power value that is a pseudo random number between 0 and 1 (0 < \( P_i < 1 \)) for a particular seed.
The values stored for each transform are pseudo-random seed which minimizes the RMS error between the domain and the range and offset which minimizes the RMS error between the range block and domain block.

5.7.2: Encoding Procedure

1. Partitioning of the original image into N non-overlapping range blocks \( \{R_i\}_{N_{k=1}} \)
2. Selection of a pseudo random seed for the power factor that minimizes the distance between the domain and the range.

The encoding procedure is as described in section 5.6.2.

5.7.3: Decoding Procedure

1. Create a random image.
2. Apply the transformation (equation 5.6) for each range block.
3. Repeat step 2 till the image converges. This normally happens in about 9 iterations.

![Figure 5.12: Compressed Randomized non-linear PIFS image](image)

5.8: Difference based Randomized compression

Here difference based compression [64] is implemented using randomized scaling factors using pseudo random generators. Here, the difference between range and domain pixel is scaled. Pseudo random numbers are used for scaling rather than a fixed scaling factor.

5.8.1: Transform

The set of transforms are of the form

\[
R_i = D_i + s_i \times (R_i - D_i) + O \quad \text{ (Equation 5.7)}
\]
Where \( R_i \) is the Range value of \( i^{th} \) pixel, \( D_i \) is the Domain value of \( i^{th} \) pixel, \( O \) is the offset value, \( s_i \) is the scaling factor that is a pseudo random number between 0 and 1 \((0 < s_i < 1)\) for a particular seed.

5.8.2: Encoding Procedure

The encoding procedure is as described in section 5.6.2.

Repeat the following procedure for all range blocks.

1. Choose a range block.
2. From all available seeds for generating pseudo random numbers, choose a seed which minimizes the distance between the only domain (first) and the range.
3. Store the seed value and offset \((O)\) that minimizes the distance between the domain and the range.

5.8.3: Decoding Procedure

1. Create an image initialized with seed value.
2. Apply the transformation (equation 5.7) for each range block.
3. Repeat step 2 as many times as the number of iterations.

Figure 5.13: Compressed Difference based-Randomized image

Figure 5.13 represents the decoded image using this technique.

5.9: Difference based Non-linear Randomized compression

5.9.1: Transforms

The set of transforms are of the form

\[
R_i = R_i + (D_i - R_i) s_i + O
\]

(Equation 5.8)
Where \( R_i \) is the Range value of \( i^{th} \) pixel, \( R_n \) is the Range value of \( nth \) pixel, \( D_i \) is the Domain value of \( i^{th} \) pixel, \( O \) is the offset value, \( P_i \) is the power value that is a pseudo random number between 0 and 1 \((0 < P_i < 1)\) for a particular seed.

5.9.2: Encoding Procedure

1. Partitioning of the original image into \( N \) non-overlapping range blocks \( \{R_i\}_{k=1}^N \)
2. Selection of a pseudo random seed for the power factor that minimizes the distance between the only domain and the range.
3. Initial seed value for image that minimizes the RMS error between the compressed and decompressed image and the initial image.
4. The number of iterations that minimizes the RMS error between the compressed image and the initial image

Repeat the following procedure for all range blocks.

1. Choose a range block.
2. From all available seeds for generating pseudo random numbers, choose a seed which minimizes the distance between the only domain (first) and the range.
3. Store the seed value, symmetry and offset \((O)\) that minimizes the distance between the domain and the range.

5.9.3: Decoding Procedure

1. Create an image initialized with seed value.
2. Apply the transformation (equation 5.8) for each range block.
3. Repeat step 2 as many times as the number of iterations.

*Figure 5.14: Compressed Difference based non-linear randomized Image*

Figure 5.14 represents the decoded image using this technique.
5.10: Discussion
Difference based image compression is sensitive to initial seed value and number of iterations. We need to figure out optimum values for these two quantities. If the seed value is changed, the resultant image also changes. The image does not converge to the same value as the number of iterations increase.

In all the above implementations, the scaling factor S has been kept as 0.75 and power p is also assigned 0.75. Varying these values give small changes in RMS error values. The pattern of image changes based on seed values and the number of iterations are of interest. The RMS error decreases and then increases with the change in seed values.

Non-linear methods can be used to compress images. Convergence can be achieved by keeping the exponent to a value between 0 and 1. Other simple functions like logarithm, sine and cosine can also be used to achieve convergence. The closest domain for each range is identified in each of the techniques mentioned in sections 5.2 to 5.5. In PIFS, the RMS error between the range and the closest domain is found to be the least across all ranges. In difference based techniques, the closest domain for each range have higher RMS error when compared to that of PIFS. Hence these images have degraded when compared to that of PIFS generated in section 5.2. The cumulative decoding error also increases the RMS error in the decoded images.

Table 5.1 shows the comparison between the different image compression schemes experimented and the different RMS errors associated with them when the image was compressed from 15.6 Kb to 11.7 Kb each.

<table>
<thead>
<tr>
<th>Method</th>
<th>Size</th>
<th>RMS Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIFS</td>
<td>11.7 Kb</td>
<td>222</td>
</tr>
<tr>
<td>Nonlinear PIFS</td>
<td>11.7 Kb</td>
<td>1022</td>
</tr>
<tr>
<td>Difference</td>
<td>11.7 Kb</td>
<td>2763</td>
</tr>
<tr>
<td>Difference-based Non-linear</td>
<td>11.7 Kb</td>
<td>48000</td>
</tr>
</tbody>
</table>

Table 5.1: Compression results of difference based compression
Instead of using constant scaling and power factors for compression using fractal techniques, we can use variable scaling and power factors. Since the pixels in a range are generally different, different scaling factors would help in getting better compression ratios. We have used randomized scaling factors being used for four different techniques and achieved reasonable compression ratios with good picture quality.

Table 5.2 shows the comparison between the different image compression schemes experimented and the different RMS errors associated with them when the image was compressed to 7.81 Kb each from an original image of size 15.6 Kb. Looking at the table, we see that difference based techniques perform better than the normal PIFS (randomized) techniques.

<table>
<thead>
<tr>
<th>Method</th>
<th>RMS Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIFS</td>
<td>222</td>
</tr>
<tr>
<td>PIFS – Random</td>
<td>7350</td>
</tr>
<tr>
<td>Difference based Random</td>
<td>2591</td>
</tr>
<tr>
<td>Nonlinear Random</td>
<td>3448</td>
</tr>
<tr>
<td>Nonlinear Difference-based Random</td>
<td>6964</td>
</tr>
</tbody>
</table>

Table 5.2: Compression results of difference based randomized compression.

**5.11: Conclusions**

We can use the difference method to achieve fractal image compression. Here, we have used linear transforms for compression of images based on difference between range and domain pixel values, performed non-linear transform on the image as well as performed a difference based non-linear transform. Non-linear transforms can be used to improve the compression factor. Study of the change in image due to seed change and change in iterations seem to be interesting.
Certain images are suitable for compression using PIFS where the transformation from the range to domain is taking place in a linear way. We have to identify some other images where difference based techniques and non-linear techniques give better results of compression. We should also identify suitable degrees in the corresponding images to get the appropriate terms of compression for non linear fractal compression. We see that compression is possible with non-linear and difference based techniques and fine tuning is required to identify appropriate images and appropriate parameters to demonstrate that this technique is practically applicable.

Increasing the number of seed values could improve the quality of the image but this could also increase the size of the compressed image. The image can be partitioned into sectors and each sector can have a different seed value. But these seed values have to be stored. Hence the size of the image could get increased by increasing the number of seed values and would contribute to a reduced benefit in terms of compression.

Compression can be achieved with randomized techniques. A fixed domain is sufficient to achieve reasonable levels of compression with good accuracy. The compressed images have smaller size when compared to other methods of fractal compression. The space required to address a large number of domains is replaced by a seed value which takes only 8 bits. Improvement in quality is not seen with increase in trying out more seed values.

PIFS performs compression most of the cases. Randomized algorithms where only one domain is searched gives better compression ratio. In the table 5.2 we see that PIFS performs better.

There is a trade off between quality and the size of the compressed image. We can compare table 5.1 and table 5.2 the RMS errors are less in table 5.1 when compared to table 5.1. However, the size of the compressed images based on randomized algorithm is 7.81 KB and size of PIFS compressed image is 11.7 KB. Size of original image is 15.6 KB. All other advances made in fractal compression can be applied to non-linear algorithms and randomized algorithms to arrive at current day compression ratios.

In decompression, We iterate over different seed values and figure out noise levels (RMS error). One with minimum noise is chosen. This increases the compression time.
We were hoping that difference based compression technique would give better compression ratio than PIFS. We have achieved close results when compared to PIFS but have not identified any image with better quality for same compression. Randomized PIFS has given better results with higher compression ratio. With randomized difference based PIFS, we have been able to achieve higher compression ratio at the cost of signal quality. The compression and decompression time increases as we add non-linear terms which add to the computation complexity.

The domain-ranges pair has values varying uniformly in terms of difference and the non-linear factor used. This should happen across all selected domain-range pairs. This may result in better compression with non-linear difference based techniques.

We should know the difference between domain-range values and the variation in terms of non-linearity of this difference to figure out which technique would be suitable for which image. It is very difficult to categorize this since there are a lot of domain-range pairs.

Now a day, JPEG and TIFF compression techniques give better compression ratio and quality when compared to fractal based compression. TIFF is normally a lossless compression technique. The JPEG compression and decompression is generally faster than fractal compression.