Fractal Image Compression on Hexagonal Structure

Huaqing Wang, Xiangjian He, Tom Hintz, and Qiang Wu
Computer Vision Research Group
University of Technology, Sydney
{huwang,sean,hintz,wuq}@it.uts.edu.au

Abstract

Image compression has many applications. For example, it is an important step for distributed and network based pattern recognition. For real time object recognition or reconstruction, image compression can greatly reduce the image size, and hence increase the processing speed and enhance performance. Fractal image compression is a relatively recent image compression method. Its basic idea is to represent images as a fixed point of a contractive Iterated Function System (IFS). Spiral Architecture (SA) is a novel hexagonal image structure on which images are displayed as a collection of hexagonal pixels. The efficiency and accuracy of image processing on SA have been demonstrated in many recently published papers. We have shown the existence of contractive IFS’s through the construction of a Complete Metric Space on SA. The construction of range and domain blocks for fractal image compression on SA is highly related to the uniform image separation specific to SA. In this paper, two presentations of SA on the traditional display device will be discussed. Then we will review the current research work on fractal image compression based on SA using both presentations. The performance on SA will be compared with it on the traditional square structure in terms of compression ratio and PSNR. In the experimental results, higher PSNR values can be achieved at various compression ratios for all test images.

Keywords—fractal coding, hexagonal image structure, image storage, image addressing, Spiral Architecture.

1. Introduction

Fractal image compression is a relatively recent image compression method which exploits similarities in different parts of the image. During more than two decades of development, the Iterated Function System (IFS) based compression algorithm stands out as the most promising direction for further research and improvement [1]. The basic idea is to represent an image as the fixed points of IFSs. Therefore, an input image can virtually be represented by a series of IFS codes. In this way, a compression ratio 10000:1 can be achieved [2].

The research work to be presented in this paper is based on a novel data structure, Spiral Architecture (SA) [3], which is inspired from anatomical considerations of the primate’s vision [4]. On Spiral Architecture, an image is a collection of hexagonal picture elements as shown in Figure 1. In the case of human eye, these elements (hexagons) would represent the relative position of the rods and cones on the retina.

In our earlier research work on image compression based on SA [5], we focused on the properties of the hexagonal pixel address labeling scheme. The property of interest was the physical proximity of the hexagonal pixels with neighboring addresses.

The research reported in [6] used the properties of uniform image partitioning based on Spiral Architecture. On Spiral Architecture, an image can be partitioned into a few sub-images [7] each of which is a scaled down near copy of the original image. Namely, each sub-image holds all the representative intensity information contained in the original. Using
such properties, in our work, the points in the original image first were re-allocated into a few groups, sub-images. The similar pixel intensity was found between the corresponding points in the different sub-images. Then, one sub-image was chosen as the reference image, and the intensity difference between the reference image and other sub-images was computed. After that, the information of the original image was coded by recording only the reference sub-image and the intensity difference information.

Based on the features of locality and uniformity on SA, in [8], we investigated various methods for lossless and lossy image compression and concluded that SA-based image compression would improve the performance of image compression.

In our work shown in [9, 10], we proposed fractal image compression (FIC) on SA. In those papers, we adopted the algorithms on traditional square image structure. The FIC algorithms relied on the uniform image separation to 7^n parts. The size of range blocks and domain blocks for FIC was assumed to be a multiple of 7 pixels.

In [11], a more flexible FIC method on SA was proposed. In order to have a clearer comparison with the Fisher’s FIC method [12] based on square structure, the size of domain blocks was set to be 4 times of the size of range blocks.

Up to this point, all fractal image compression research on SA is based on a virtual SA [13] that presents SA on the traditional device. However, the conversion between image structures requires heavy computational complexity, which results in a considerable period of time. Therefore, the research work was further improved in [14], where a novel virtual hexagonal structure was applied to increase the computation speed and to compress images with rectangular shapes. In addition, with the new presentation of SA, developed FIC algorithms on the traditional rectangular structure can be implemented on SA with obvious modification.

In this paper, we review FIC algorithms on SA. The organization of this paper is as follows. Section 2 reviews the Spiral Architecture. Section 3 introduces the virtual SA. The FIC algorithm on SA is then presented in Sections 4 with an improved version in Section 5. The FIC on a novel hexagonal structure is depicted in Section 6. We conclude in Section 7.

2. Spiral Architecture

On the Spiral Architecture, an image is a collection of hexagonal elements [4]. Each pixel on The Spiral Architecture is identified by a designated positive number of based 7, called Spiral Address as shown in Figure 1. The numbered hexagons form the cluster of size 7^n. The hexagons tile the plane in a recursive modular manner along the spiral direction [7].

Spiral Counting [4] can be considered as a spiral movement from a given commencing hexagon counting for a predetermined number and terminating at another certain hexagon. Any hexagon in an image can be reached by Spiral counting from any other given hexagon in the same image. When applying Spiral counting, it is strictly dependent on a predetermined key define by Sheridan in [4]. A key is the first hexagon to be reached in an instance of a Spiral counting, which determines two important parameters: the distance and the orientation. For instance, given a Spiral address 15, the key of 15 can determine two values. One is the distance between the given hexagon 15 and the hexagon 0; the other is the orientation of hexagon 15 from hexagon 0.

Spiral counting is used to define two operations in the SA, which are Spiral Addition and Spiral Multiplication [3]. Let a and b be Spiral addresses of two arbitrarily chosen hexagons in SA.

Definition 1. [15] Spiral addition of a and b, denoted by a \( \oplus \) b, is the Spiral address of the hexagon found by Spiral counting b hexagons in the key of Spiral address 1 from the hexagon with Spiral address a.

For example, 3 \( \oplus \) 2 = 26.

Definition 2. [15] Spiral multiplication of a and b, denoted by a \( \otimes \) b, is the Spiral address of the hexagon found by Spiral counting b hexagons in the key of Spiral address a from the hexagon with Spiral address 0.

For example, 15 \( \otimes \) 2 = 26.

A Spiral Multiplication corresponds to an image rotation with scaling and a Spiral Addition is equivalent to an image translation. These two operations are used for the definition of range blocks and domain blocks on Spiral Architecture.

3. Virtual SA

SA has many advantages against the traditional rectangular image structure. However due to the lack of capture and display devices, SA has not yet been widely used in image processing. In order to make SA applicable on the current available devices, we constructed a mimic scheme called Virtual Spiral Architecture (VSA), with which images on rectangular structure can be smoothly converted to SA.

VSA mimicking scheme is so called ‘virtual’ because it only exists on computer memory during the procedures of image processing. The processing result will still be displayed on the traditional rectangular structure (see Figure 2).
In order to keep the resolution, hexagonal and square pixels are defined as the same size, i.e. 1 unit area. Then if we map the SA on a traditional image structure and then let $N$ denote the number of square pixels covered a hexagonal pixel and let $s_i$ represent the size of overlapped area in a certain square pixel $i$ (See Figure 3), so the contribution of gray level given by this square pixel to the hexagonal pixel is measured by the percentage of the overlapped area, i.e. $p_i$, as follows

$$p_i = \frac{s_i}{100}.$$  \hfill (3.1)  

Therefore the gray value of this hexagonal pixel is

$$g_{\text{hex}} = \sum_{i=1}^{N} (g_{\text{Squ}_i} \cdot p_i),$$  \hfill (3.2)  

where $g_{\text{Squ}_i}$ is the gray level of the $i$-th square pixel.

As a result, the gray level information for SA is now available for image processing and the experiment result can be displayed back on the conventional devices following the similar but opposite mapping method (See Figure 4).

4. Basic FIC on SA

The fundamental principle of fractal image compression consists of the representation of an image by an iterated function system (IFS) of which the fixed point is close to that image. This fixed point is named as ‘fractal’ [12]. Each IFS is
then coded as a contractive transformation with coefficients. Banach’s fixed point theorem guarantees that, within a complete metric space, the fixed point of such a transformation may be recovered by iterated implementation thereof to an arbitrary initial element of that space [16]. Therefore, the encoding process is to find an IFS whose fixed point is close to the given image. The usual approach is based on the Collage Theorem [17], which provides a bound on the distance between the image to be encoded and the fixed point of an IFS. A suitable transformation may therefore be constructed as a ‘collage’ from the image to itself with a sufficiently small ‘collage error’ (the distance between the collage and the image) guaranteeing that the fixed point of that transformation is close to the original image [18].

In the original approach, devised by Barnsley, this transformation was composed of the union of a number of affine mappings on the entire image [1]. Fractal image compression became a practical reality with the introduction by Jacquin of the partitioned IFS (PIFS) [19], which differs from an IFS in that each of the individual transformation operates on a subset of the image, rather than the entire image. Since the image support is tiled by ‘range blocks’, each of which is mapped from one of the ‘domain blocks’ as depicted in Figure 5, the combined mappings constitute a transformation on the image as a whole. The transformation minimizing the collage error within this framework is constructed by individually minimizing the collage error for each range block, which requires locating the domain block which may be made closest to it under an admissible block mapping. This transformation is then represented by specifying, for each range block, the identity of the matching domain block together with the block mapping parameters minimizing the collage error for that range block.

![Figure 5. Each range block is constructed by a transformed domain block](image)

For each range block $R$, the FIC encoder must find the best approximation

$$R \approx sD + oI,$$

where $D$ is a codebook block transformed from a domain block to the same size as $R$. The coefficients $s$ and $o$ are called scaling and offset. If we use the Euclidean norm, the (1) is equivalent to minimizing

$$E(D, R) = \min_{s, o} \| R - (sD + oI) \|.$$

The well known least squares method can be used to find the optimal coefficients directly as shown in [9,10].

In our early research on SA-Based FIC, we follow the same idea applied on square structure as described above. Firstly we separate the image into range blocks of seven hexagonal pixels and define the domain blocks of seven times more, i.e. 49 pixels (see Figure 6). Each pixel in the image can be the centre of a domain block. Then we include the neighboring 48 pixels around it based on Spiral counting to form a domain block unless any pixel of this domain block is out of the given image.

![Figure 6. Range and domain blocks in Spiral Architecture](image)

![Figure 7. Testing images: Building, Boat and House](image)
We use three figures as shown in Figure 7 to test our algorithms. Tables 1 and 2 summarize compression results on square structure (SQ) and SA. As the size of each range block on SA is of 7 pixels (compared with 16 pixels in square structure), the compression ratio is slightly lower but the quality of decompressed image has increased with higher PSNR output. Figures 8-10 show the original testing images represented on SA and the decompressed images.

<table>
<thead>
<tr>
<th>Image</th>
<th>Compression ratio</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building</td>
<td>3.37</td>
<td>23.40</td>
</tr>
<tr>
<td>Boat</td>
<td>3.37</td>
<td>26.56</td>
</tr>
<tr>
<td>House</td>
<td>3.37</td>
<td>22.41</td>
</tr>
</tbody>
</table>

Table 2. Summary for images on Spiral Architecture

<table>
<thead>
<tr>
<th>Image</th>
<th>Compression ratio</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building</td>
<td>2</td>
<td>25.43</td>
</tr>
<tr>
<td>Boat</td>
<td>2</td>
<td>29.73</td>
</tr>
<tr>
<td>House</td>
<td>2</td>
<td>26.20</td>
</tr>
</tbody>
</table>

Figure 8. Original and compressed ‘building’ in Spiral Architecture

Figure 9. Original and compressed ‘boat’ in Spiral Architecture

Figure 10. Original and compressed ‘house’ in Spiral Architecture

5. FIC on SA with Variable Sizes of Range Blocks

The experimental results in the previous section are hardly satisfying. The obvious difference with the FIC algorithms on the rectangular structure remains: firstly, the size of the range block on SA is 7 on SA but 16 on rectangular; secondly,
on SA the ratio between range block and domain block is no more the preferred 1:4 but 1:7; in addition, the compression ratio on SA is not flexible to make a complete comparison. We could have possibly seen the potential of SA in the improvement of FIC performance; however, all of the above points make it ambiguous. Therefore, some modifications are required in order to fully investigate the advantages of FIC on SA.

In [11], the range blocks for FIC on SA were redefined. The size of range blocks does not have to be a multiple of 7 pixels and each domain block is 4 times (rather than 7 times) as big as range blocks.

As stated earlier, when an image is projected onto SA, every hexagonal pixel is identified by its unique spiral address, and image translation and partitioning can be performed based on the two spiral operations.

Figure 11 shows an example of image partitioning using spiral multiplication. The original image has 16807\textsubscript{10} hexagon pixels. The multiplier used in spiral multiplication is 55555\textsubscript{7}. The separated sub-image areas are shown using different illuminations and labeled by different area numbers (see Figure 11b). Finally, the fragments of each sub-image are collected together to produce a complete partitioned sub-image. The addends used in spiral addition for fragment collection are also shown on each sub-image (see Figure 11c).

The uniform image partitioning on SA applied here reveals the one-to-four relationship between the original image and its four sub-images. Therefore, for any given block in the SA space, it would be possible to find another corresponding area of four-time larger size by implementing the opposite operation of this uniform image partitioning. If we consider the smaller block as a range block, the larger block then would be one of the candidate domain blocks for this range.

In Fisher’s basic FIC algorithm on SQ, the union of all range blocks in the same size covers the complete image space and there is no overlap between range blocks. The FIC on SA follows the fundamental idea with some enhancement for the flexibility in compression ratio.

![Figure 12. Two range blocks defined within a sub-image of 49 pixels](image-url)

Figure 11. Uniform image partitioning on SA. (a) Original image, (b) Four labeled sub-images after spiral multiplication, (c) Four complete sub-images after fragment collecting with spiral addition.

In Fisher’s basic FIC algorithm on SQ, the union of all range blocks in the same size covers the complete image space and there is no overlap between range blocks. The FIC on SA follows the fundamental idea with some enhancement for the flexibility in compression ratio.

![Figure 12. Two range blocks defined within a sub-image of 49 pixels](image-url)

The procedure begins with an equal-size image partitioning on SA. The number of pixels in each sub-image is a power of seven, i.e., 7, 49 or 343. A sub-image can then be defined as a range block or it can be further separated into parts of similar size to obtain smaller range blocks. In the example shown in Figure 12, the image is firstly partitioned into sub-images of 49 pixels and within each sub-image two range blocks are defined, namely the first range block has 25 pixels and another has 24 pixels so that the 49 pixels sub-image space is completely covered and the two range blocks have no overlap. Following the systematical scheme, range blocks can be defined quickly based on spiral counting and the size of range block is adjustable so that flexible compression ratios are available for further analysis.

Following the Fisher’s FIC algorithm on SQ, the candidate domain block is four times larger than the given range block. Each pixel in a given range block should find four corresponding pixels in the candidate domain block. In addition, those sets of four pixels do not overlie with each other and the union of them will completely cover the candidate domain block. It also requires that the shape of the partition needs to be as similar as possible to the given range block. For example, it is often a square-to-square mapping between a range block and its domain block in most of FIC approaches on SQ.

The task on SA can be fulfilled by the opposite operation of spiral multiplication. Let us consider a sub-image in Figure 8c as a range block and it is easy to discover that one suitable domain block would be the image shown in Figure 8a. Then the 1 to 4 relationship between pixels in a range block and its domain block can be identified by the opposite operation of spiral multiplication, namely spiral division. The number 55555\textsubscript{7}, which was previously used as multiplier, is used now to be a divisor. This spiral division is implemented over every range block pixel to find out the spiral address of the first corresponding pixel in the domain block. The other three corresponding pixels are traced by spiral addition with addends 1, 2 and 6 respectively.
With the assistance of this definition scheme based on spiral operations, for a given range block in spite of the size and shape, a set of corresponding domain blocks could be found. As long as the pixels in the range block are continuous and non-overlapped, the domain block pixels cover a constant space without overlap. This guarantees that the ratio between range block and domain block used in our proposed FIC on SA follows the same 1:4 ratio as Fisher’s basic FIC algorithm. Once the range and domain blocks are defined, for each range block, the fractal transformation is constructed by searching for all the elements in the domain pool to find the most similar block following Fisher’s basic FIC.

The same three images as shown in Figure 7 were used to test our improved algorithms. For every testing image, the performance of FIC on SQ is compared with that on SA and the results are shown in the figures from Figures 13 to 15. In these figures, the curve representing FIC performance on SA is always above that for SQ. It is easy to see that there are higher PSNR values over different compression ratios for all tested images.

The improved performance is introduced by the enhanced image structure, i.e. SA. First of all, SA has higher degrees of symmetry that results in a considerable saving in storage [11]. Meanwhile the beautiful circular symmetry guarantees that the search of the best domain block is smoother and closer to the given range block compared with that on SQ can only follow either horizontal or vertical direction. In addition, in our preliminary research on image compression on SA, it has been found that for a fixed size block on SA always introduces less average variance than on SQ [12]. This less variance can reduce the error when computing the best similarity in FIC and eventually increases the fidelity in both encoding and decoding.

![Figure 13. PSNR versus Compression Ratio for Building](image1.png)

![Figure 14. PSNR versus Compression Ratio for Boat](image2.png)

![Figure 15. PSNR versus Compression Ratio for House](image3.png)
6. FIC on a Novel SA Presentation

Up to Section 5, we have already fully revealed the potential of SA in improving the performance of FIC. However, an inconvenient fact is that the relatively matured FIC algorithms developed based on the rectangular structure will need great modification to be applicable to SA. This has results in difficulties in our research. Therefore, a new SA presentation rather than VSA is proposed, namely, Virtual Hexagonal Structure (VHS).

The idea of uniformly separating an image into four sub-images is adopted in this section for the definition of range and domain blocks, however, on a virtual hexagonal structure (VHS). There are the following differences between our previous approach and the work to be presented in this section. First of all, we will not explicitly use the time consuming Spiral Addition and Multiplication for range and domain blocks definition. Secondly, the shape of images to be compressed is no longer hexagon-like, and can be a rectangle. The third but not the least difference is that the conversion between image represented on square structure and the hexagonal structure is fast and easy. The construction of the virtual hexagonal structure and the operation for image separation is shown as follows.

To construct virtual hexagonal pixels, each square pixel is first separated into 7×7 smaller pixels, called sub-pixels. To be simple, the light intensity for each of these sub-pixels is set to be the same as that of the pixel from which the sub-pixels are separated. Each virtual hexagonal pixel is formed by 56 sub-pixels. To be simple, the light intensity of each constructed hexagonal pixel is computed as the average of the intensities of the 56 sub-pixels forming the hexagonal pixel when necessary. Figure 16 shows the arrangement of each hexagonal pixel with its 56 sub-pixels and also a collection of seven hexagonal pixels constructed with spiral addresses from 0 to 6.

The idea for image separation on the virtual structure is to define rows and columns as images represented on the square structure. Therefore, each hexagonal pixel on SA has also a Cartesian coordinator like the one used for square image structure. The definition of rows and columns is demonstrated in Figure 17 and Figure 18. Then, each sub-image is a collection of hexagonal pixels from every second row and column. Figure 19 shows the results of image separation.

After the virtual image represented using hexagonal pixels is obtained, FIC procedures become obvious by simply following Fisher’s basic FIC algorithm on a square structure. The size of range block is scalable and can be $2 \times 2$, $3 \times 3$, $4 \times 4$, $6 \times 6$, $8 \times 8$ and so forth which make the various compression ratios available. Then, the performance of the proposed codec can be compared with those obtained from the original square image using the same scales.

When testing our algorithms using the three images shown in Figure 7, similar results to those shown in Figures 13 to 15 in the previous section were obtained and shown in Figures 20-22. It was found that the FIC performance on VHS was always above that for square structure. It is easy to see that the proposed FIC on VHS always performs better than that on square because there are higher PSNR values for different compression ratios for all test images.
Figure 17. Columns on a hexagonal structure

Figure 18. Rows on a hexagonal structure

Figure 19. Four similar sub-images separated from a 512×384 a car image

Figure 20. PSNR versus Compression Ratio for Building.
Figure 21. PSNR versus Compression Ratio for Boat.

Figure 22. PSNR versus Compression Ratio for House.

7. Conclusions and Future Work

In this paper, we have reviewed the research work for FIC on a hexagonal image structure. The investigation carried out and the performance analyses show the advantages of the FIC algorithms on SA. By inheriting the basic FIC method directly from square structure (SQ) to SA, it gives more accurate compression results (higher PSNR at the same compression ratios as on SQ) for all testing images. The FIC algorithms also have flexible range block sizes so that the compression ratio is adjustable.

There are many ways to improve FIC on SA for future work. In order to improve the quality for image display and representation on VHS, a better image interpolation method must be developed to convert images represented on SQ and SA. An adaptive FIC method that uses different sizes for different range blocks (and domain blocks) is another research direction to further enhance the compression performance. For real-time web-based applications, it will be interested to consider progressive FIC encoding and decoding [20], which can more efficiently use network bandwidth and increase image transmission speed. Parallel processing is another potential approach to increase the computation speed and this can be performed through the uniform image separation on SA using a spiral multiplication.

References