Fractal Image Compression

Literature Survey

Viswanath Sankaranarayanan

19^{th} October, 1998

Abstract

The demand for images, video sequences and computer animations has increased drastically over the years. This has resulted in image and video compression becoming an important issue in reducing the cost of data storage and transmission time. JPEG is currently the accepted industry standard for still image compression,but alternative methods are also being explored. Fractal Image Compression is one of them. This scheme works by partitioning an image into blocks and making use of Contractive Mapping in which the range blocks are covered by domains. In this project we proposes to implement a method of partitioning an image, which would reduce the encoding time.

Contents

1 Introduction

The encoding step in fractal image compression involves very large time complexity. In this project we proposes to implement a method to reduce the encoding time. First an introduction to fractals and fractal image compression is presented. Then the complexity involved in the encoding time is looked into and finally a plan to implement the improvement is given.

1.1 Fractals

The term *Fractals* was coined by Mandelbrot[6] in 1975. A general definition of a fractal is a set F that has any of the following properties[2]:

- F has detail at every level.
- F is exactly, approximately or statistically *self-similar*.
- The *Hausdorff Besicovitch*[6, 2] dimension of F is greater than its topological dimension.
- There is a simple algorithmic description of F.

The defining characteristic of a fractal is that it has a fractional dimension[7].

1.2 Self-similarity

Subsets of fractals when magnified, appear similar or identical to the original fractal and to other subsets. This property called *self-similarity*[2, 7] makes fractals independent of scale and scaling. Thus there is no characteristic size associated with a fractal.

1.3 Affine Transformations

An *affine transformation*[5, 3] maps a plane to itself. The general form of an *Affine Transformation* is

$$w_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix}$$

Affine transformations can skew, stretch, rotate, scale and translate an input image.

1.4 Attractor

Repeatedly applying an affine transformation repetitively on any image results in an image called the *attractor*[2, 7]. The attractor is unchanged for any further application of the affine transformation. Each affine transformation has a characteristic attractor associated with it.

1.5 Iterated Function Systems

An *iterated function system*[5, 1] is a collection of affine transformations that map a plane to itself. This collection defines a map W given by

$$W(.) = \bigcup w_i(.)$$

1.6 Contractive Mapping Fixed-Point Theorem

The *contractive mapping fixed-point theorem*[1, 3] states that if we are given a contractive map W, then there is an attractor denoted by x_w with the following properties:

- Applying the affine transformation to the attractor yields the attractor. This attractor x_w is called the *fixed point* of W, i.e.

$$W(x_w) = x_w = w_1(x_w) \cup w_2(x_w) \cup \ldots \cup w_n(x_w)$$

- Given any input image, if we run the affine transformations sufficiently large number of times, the resulting image is the attractor, i.e.

$$x_w = S_\infty = \lim_{n \to \infty} W^{\circ n}(S_0)$$

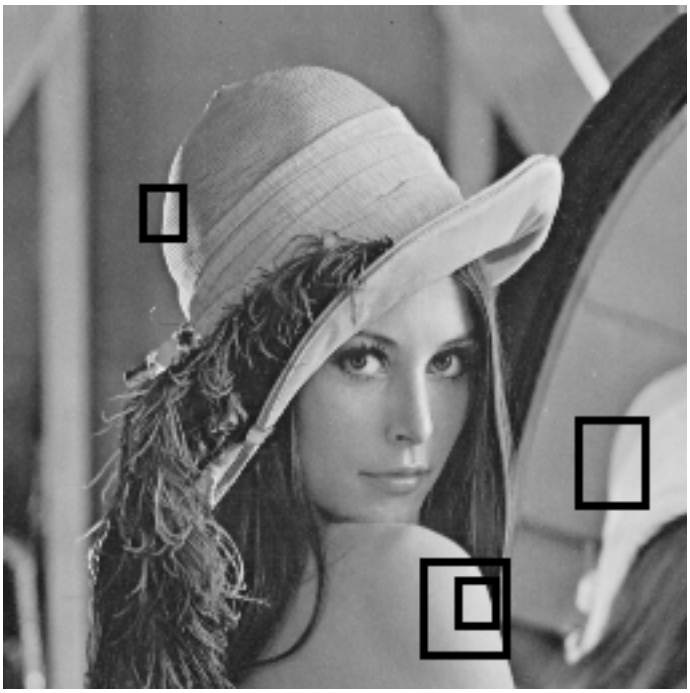
- x_w is unique.

2 Fractal Image Compression

M.Barnsley[5] suggested that storing images as a collection of transformations would result in image compression. In this section we see how the theory of fractals can be used to compress images.

2.1 Self-Similarity in Images

A typical image does not contain the type of self-similarity found in fractals. But, it contains a different sort of self-similarity. The figure on the next page shows regions of Lenna that are self-similar at different scales. A portion of her shoulder overlaps a smaller region that is almost identical, and a portion of the reflection of the hat in the mirror is similar to a smaller part of her hat.



The difference here is that the entire image is not self-similar, but parts of the image is self-similar with properly transformed parts of itself. Studies[6, 5, 2] suggest that most naturally occurring images contain this type of self-similarity. It is this restricted redundancy that fractal image compression schemes attempt to eliminate.

2.2 Partitioned Iterated Function Systems

To facilitate compression we extend the Iterated Function System(Section 1.5) to allow us to partition an image into pieces which are each transformed separately. A modified affine transformation for *Partitioned iterated function systems* is

$$w_i \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & s_i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \\ o_i \end{pmatrix}$$

Here s_i and o_i are the contrast and brightness adjustments for the transformations.

2.3 Ranges and Domains

The problem of fractal image compression is to find the best domain that will map to a range. A *Domain* is a region where the transformation maps from, and a *range* is the region where it maps to.

2.4 A Simple Encoder and Decoder

2.4.1 The Encoder

Consider an image that is $256 * 256$ in size and each pixel is one of 256 grey levels. The image is partitioned into $8 * 8$ pixel non-overlapping sub-square ranges, and $16 * 16$ overlapping sub-square domains. For each of the Ranges R_i we now search through all of the Domains **D** which best covers this range. We write out the position of the range, the best domain and the transformation. This process is repeated till all the ranges have been covered.

2.4.2 The Decoder

The decoding step is very simple. We start with any image and apply the stored affine transformations repeatedly till the image no longer changes. This is the decoded image.

Jacquin[1] originally encoded images with fewer grey levels using a method similar to this example. A similar method has also been used to encode contours[2].

2.5 Quadtree Partitioned Encoding and Decoding

A weakness of the previous method is that the size of the range is fixed. This results in areas with fine detail not being covered well and vice-versa. A generalization of the fixed size R_i is the use of *quadtree*[2] partitioning

of the image. In this scheme a square in the image is broken up into four equal-sized sub-squares when it is not covered well. This process repeats recursively starting from the whole image and continuing until the squares are small enough to be covered. This method yields better results than the simple encoder[2].

3 Encoding Time

3.1 Simple Encoder

For the example considered above, there are 1024 ranges and 58,081 domains. There are eight ways that each domain can be mapped to a range and this yields 464, 648 square that have to be compared for each range. The time complexity of this search is linear in the number of domains. Various techniques have been proposed to overcome this complexity. One such technique is the classification of domains based on some feature such as edges[1] or bright spots. A domain once discarded removes from the pool of domains all other similar domains for the current range. Another technique is to classify the domains as *multi-dimensional* keys and this reduces the complexity from $O(N)$ to $O(\log(N))$[4].

3.2 Quadtree encoder

The Quadtree encoder is *adaptive* in nature and therefore the number of computations necessary depend on the input image detail. Nevertheless the complexity is still quite large. This is because the partitioning of the image is based on how well the candidate range matches a domain.

3.3 Improvement

It is reasonable to base the partitioning of the image, on the variance, rather than the match of a candidate range with the domain. Doing this would miss some of the larger squares, but would result in significant reduction in encoding time[2]. This is because none of the previously discarded ranges would have to be compared to the domains. In this project we proposes to implement this improvement to the quadtree partitioned fractal compression scheme.

4 Implementation Plan

4.1 Coding

'C' code for quadtree partition based fractal compression already exists[2]. This code will have to be studied. The changes for the proposed improvement will have to be implemented and tested.

4.2 Performance Study

A reduction in the encoding time is bound to affect the compression ratio. A comparison of the performance of the new method with the old method, with respect to encoding time and compression ratio is also proposed.

4.3 Pitfalls to avoid

- Images encoded by fractal methods have no characteristic size. A $256 * 256$ image can be encoded and restored at $512 * 512$. This has the effect of making the compression ratio 4 times larger. This will be avoided and images will be decoded at the original size only.

- PSNR is not a measure of perceived image quality, and will not be used to compare the images.
- The performance tests will be carried out for a diverse set of images so that 'freak' effects do not distort the results.

References

[1] Arnaud E. Jacquin, "Image Coding Based on a Fractal theory of Iterated Contractive Image Transformations," *IEEE Trans. on Image Processing*, vol. 1, no. 1, pp. 18-30, Jan. 1992.

[2] Yuval Fisher, *Fractal Image Compression - Theory and Application*, Springer-Verlag, ISBN 0-387-94211-4, 1994.

[3] Yuval Fisher," Fractal Image Compression" ,*Course Notes*, vol. 12, ACM SIGGRAPH, 1992.

[4] Dietmar Saupe, "Breaking the Time Complexity of Fractal Image Compression," *Technical Report*, vol. 53, Institut für Informatik, Universität Freiburg, 1994.

[5] Michael F. Barnsley and Lyman P. Hurd, *Fractal Image Compression*, AK Peters Ltd.,ISBN 1-56881-000-8, 1993.

[6] Benoit B. Mandelbrot, *The Fractal Geometry of Nature*, W.H. Freeman and Company, ISBN 0-7167-1186-9, 1983.

[7] Maaruf Ali and Trevor G. Clarkson, "Fractal Image Compression," *Proc. 1st Seminar on Information Technology and it Applications (ITA '91)*, Sept. 29, 1991, Leicester.