Fractal Image Compression

Project Report

Viswanath Sankaranarayanan

4^{th} December, 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. 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 using Contractive Mapping to map range blocks to domains. The encoding step in fractal image compression has high computational complexity. In this project we implement an innovative method of partitioning an image, which reduces the computational complexity in the encoding step. We also present a comparison of this method against the quadtree method and against the JPEG standard.

1 Introduction

The encoding step in fractal image compression has high computational complexity. In this project we have implemented a method to reduce this computational complexity. We first present an introduction to fractal image compression. Then, we look into the complexity involved in the encoding step, describe the solution we have implemented, and present the results obtained. This section presents an introduction to fractal image compression.

The term *fractals* was coined by Mandelbrot [1] in 1975 to describe the irregular and fragmented patterns that appear in nature. Barnsley [2, 3] suggest that storing images as a collection of transformations will result in image compression. The image to be encoded is partitioned into non-overlapping *range blocks* $\mathcal{R}$. The encoder then finds a larger *domain block* $\mathcal{D}$ of the same image for every range block, such that a transformation of $\mathcal{D}$ is a good approximation of the range block $\mathcal{R}$. This transformation τ is a combination of an *affine transformation* [2, 3, 4, 11] γ and a *luminance transformation* λ. In matrix form τ can be expressed as

$$\tau \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & 0 \\ k_{21} & k_{22} & 0 \\ 0 & 0 & a \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \\ b \end{pmatrix}$$

Here, z denotes the pixel intensity at position (x, y). k_{ij} are the co-efficients of the affine transformation. Δx and Δy are the relative x and y positions of the range with respect to the domain. a and b are the contrast and brightness adjustments for the transformation. The encoder stores only these eight numbers. The decoder iteratively applies the transformations to any initial image and generates the reconstructed image. To ensure convergence at the decoder, the transformations τ have to be *contractive* [2, 4].

The first practical fractal compression system to appear in literature was by Jacquin [6]. He compressed an image by a block-based compression scheme. Most other compression systems to appear in literature are based on Jacquin's block-based compression system. One of the compression systems is a *quadtree-based* [4] fractal encoding scheme and we use this method to implement our improvement. Another method uses the same quadtree scheme,

but further breaks down the square region into irregular shapes before coding [5, 9].

2 Encoding Time

The objective of this project is to reduce the time complexity in the encoding step. We first see where the time complexity arises in the encoding step the time complexity arises, and how we can reduce it.

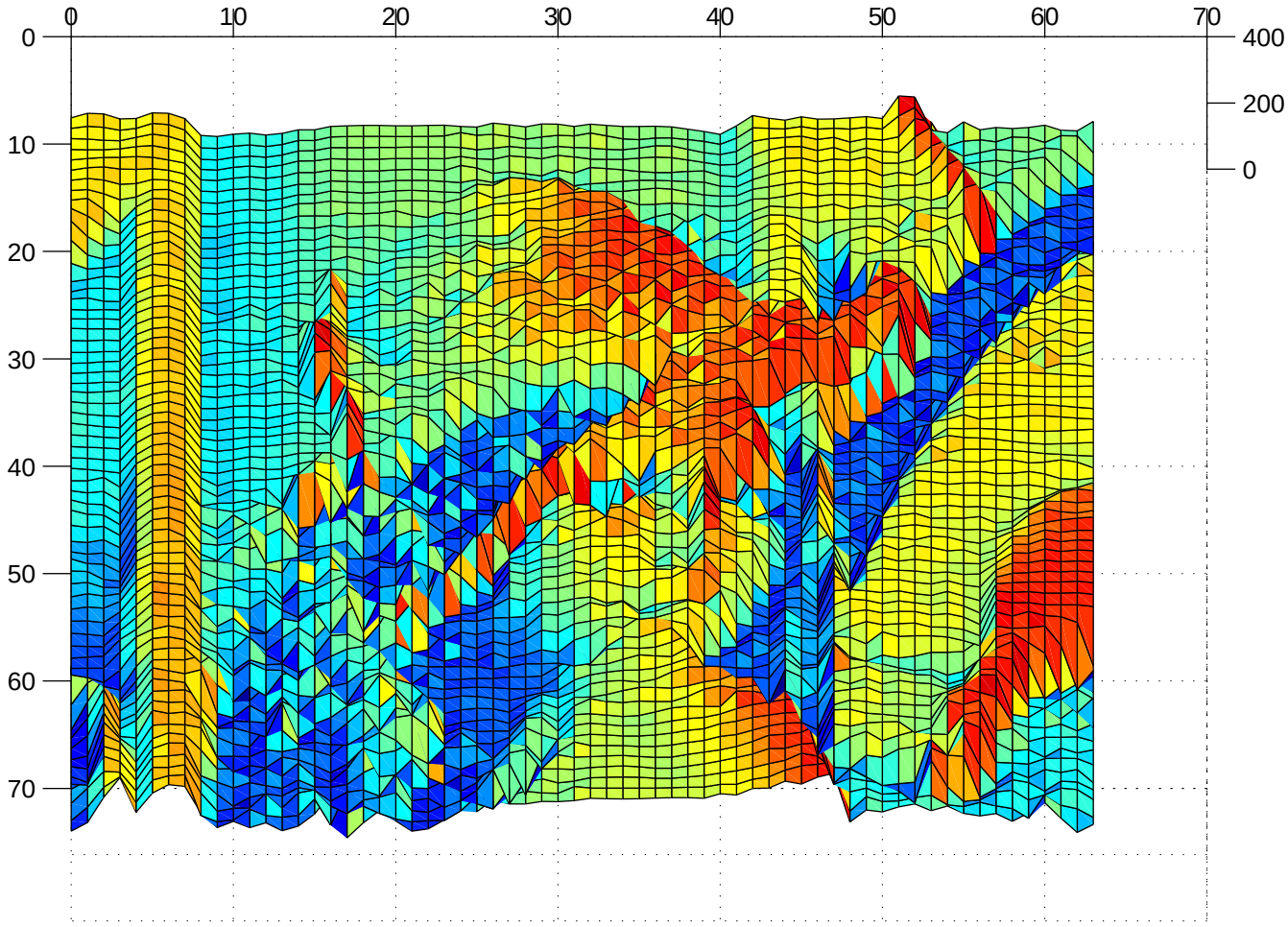
Consider an image that is 256x256 pixels in size. We partition it into 8x8 non-overlapping ranges $\mathcal{R}_1, \mathcal{R}_2 \ldots, \mathcal{R}_{1024}$. We also partition it into 16x16 overlapping domains $\mathcal{D} = \mathcal{D}_1, \mathcal{D}_2$, $\ldots, \mathcal{D}_{58,081}$. Now, for each $\mathcal{R}_i$, we search through all $\mathcal{D}$ to find a $\mathcal{D}_i$ which minimizes some threshold. In other words, we try to find a part of the image that looks similar to $\mathcal{R}_i$. There are eight ways to map one square to another. So for each range $\mathcal{R}_i$, 464,648 domains have to be compared. Various techniques have been proposed to overcome the time complexity. One such technique is the classification of domains based on some feature, such as the edges [6, 10] or bright spots [4, 8]. A domain once discarded removes from the pool all other similar domains for a particular range. Another technique is to classify the domains as *multidimensional keys*. This reduces the complexity from $\mathrm{O}(N)$ to $\mathrm{O}(\log N)$ [7].

The weakness of this method is that the size of the range $\mathcal{R}_i$ is fixed. This is overcome in the *quadtree* partition [4] method. Here, a square in the image is split up into four equal-sized sub-squares when it is not covered sufficiently by the domain. This process is repeated recursively, starting from the complete image and continued until the squares are small enough to be covered within some specified threshold. This range-domain match is the time

3 Variance Based Image Partitioning

In the quadtree partitioning, a range $\mathcal{R}_i$ which would have been split further is also compared with the domains. These *futile range* comparisons increase the encoding time unnecessarily. To prevent these futile domain searches, we have to partition the image before we begin the

encoding. We model the image as a contour of the grey-scale value versus position.



Model of Lena image

Now, for good detail and for good chance of a domain match, we have to partition the image such that, each range is a flat portion of the contour. We could use the variance metric for determining how flat the contour is. The variance metric is defined [4] as

$$V_i = \sum_{j=1}^{n} (r_j^i)^2 - A_i^2$$

where i is the range under consideration, n is the total number of pixels in the range, r_i^j is the grey level of the j^{th} pixel in the i^{th} range, and

$$A_i = \sum_{j=1}^{n} r_j^i$$

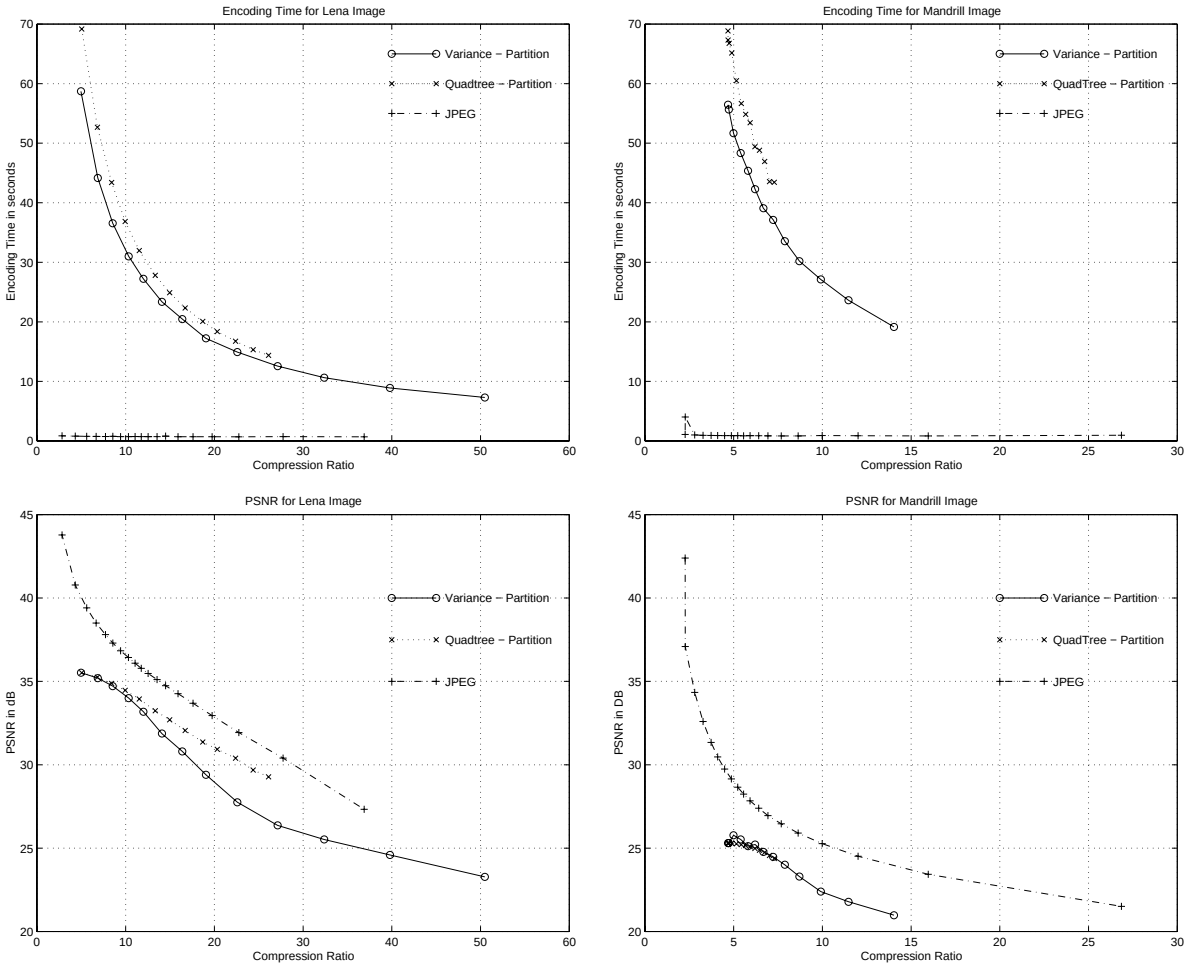
This computation is quite involved and would consume considerable time. Instead, we assume that if the peak difference between the pixels is above some threshold, we have

to split the range further. This partitioning is done before we start the domain matches and thus only useful ranges are compared with the domains. This method gives significant improvement over the quadtree partition method, with comparable image quality. The disadvantage of this method it that, some of the larger squares, which would have been mapped originally, would be skipped now.

The 'C' code for the quadtree partition method already exists [4]. We implemented the variance partition method, using this quadtree code.

4 Results

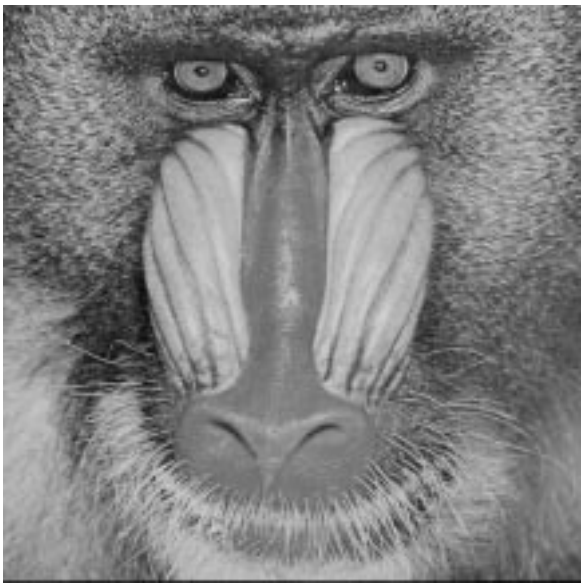
In this section, we present the results that we obtained with our method.



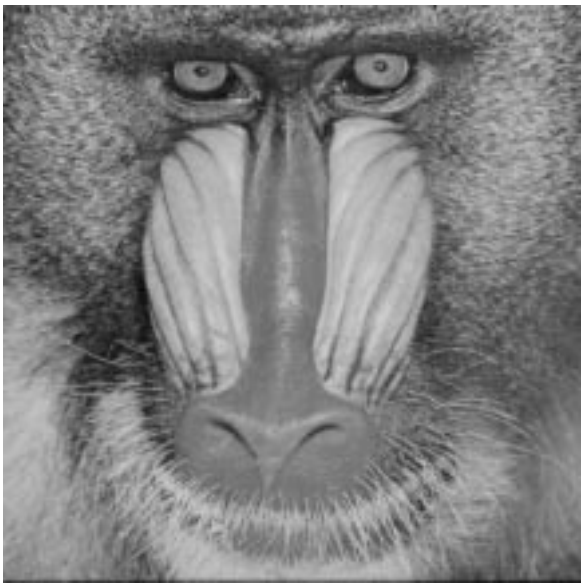
Comparison of the Encoding Time and Compression Ratio for the different schemes

We find that our method gives considerable saving in encoding time over the quadtree method for all images and compression ratios. The image quality is slightly degraded for some images at higher compression ratios. The 'Mandrill' image coded with different schemes is shown below (Compression ratio 6:1).

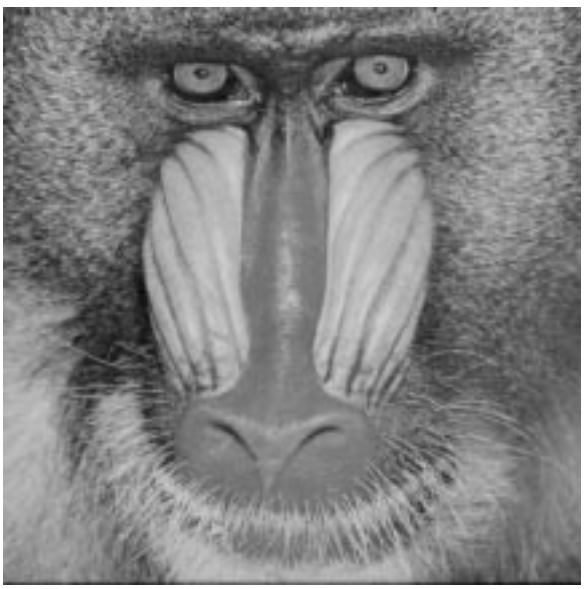
Original Image



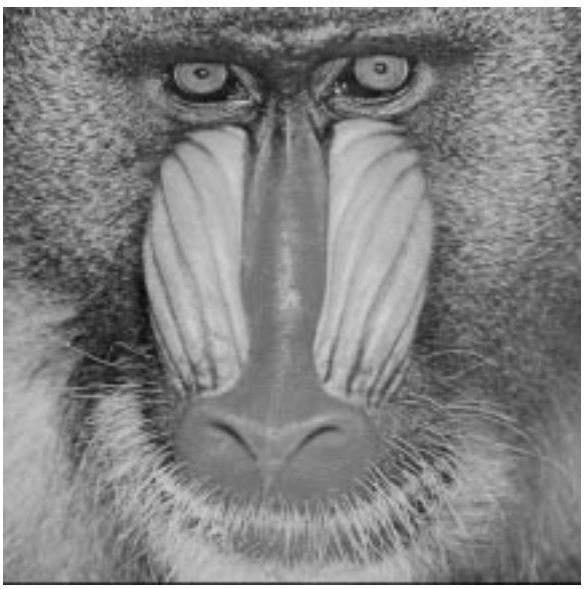
QuadTree Partitioned



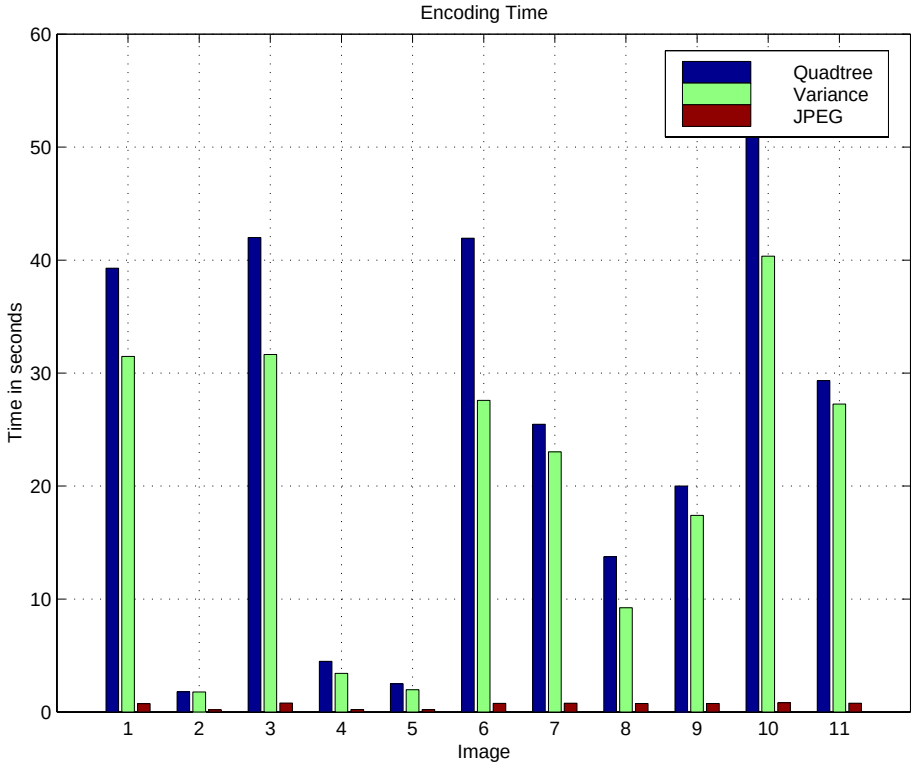
Variance Partitioned



JPEG



We finally give results for the encoding time for 11 standard images. There figures show that our method is faster than the quadtree method for all of these images.



Encoding Time on Standard Images

5 Conclusion

Our method gives a considerable saving in the encoding time over the quadtree method, with comparable image quality. The key point to note from the results is that JPEG is better than both these methods. Fractal image compression, using a quadtree partition is far from replacing JPEG as the standard for still image compression. There are some properties of fractals that make them ideal for other applications.

- Fractals have no characteristic size, and an encoded image can be decoded at any resolution.
- Realistic detail is artificially generated at all scales.

- Fractals perform very well for highly sampled, natural images.
- Synthetic images and half tones are not compressed by this method.
- The encoding step is time consuming, but the decoding step is very fast.

All these properties make fractals a good choice for transmitting images over the Internet. The resolution independence property is desired, as target browsers have different resolutions. Images are encoded only once, but decoded many times. A sizeable portion of the images over the Internet are natural images. The decoding step works is the natural way that images load up in browsers. Netscape 4.02 has a plug in for the *Fractal Image Format*. The 1992 version of Microsoft Encarta had all its images compressed in this format.

The quadtree partition is not the optimal way of compressing images [5]. Other methods like the triangular partition [4, 8] and irregular partition [5, 10] yield better image quality. Our variance partition improvement can also be applied to these methods to reduce the encoding time.

References

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

[2] M. F. Barnsley and L. Y. Hurd, *Fractal Image Compression*, AK Peters Ltd., ISBN 1-56881-000-8, 1993.

[3] M. F. Barnsley, *Fractals Everywhere*, Academic Press Professional, ISBN 0-12-079061-0, 1993.

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

[5] L. Thomas and F. Deravi, "Region-Based Fractal Image Compression Using Heuristic Search," *IEEE Trans. on Image Processing*, vol. 4, no. 6, pp. 832-838, Jun. 1995.

[6] A. 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.

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

[8] F. Davoine, M. Antonini, J. M. Chassey and M. Barlaud, "Fractal Image Compression Based on Delaunay Triangulation and Vector Quantization," *IEEE Trans. on Image Processing*, vol. 5, no. 2, pp. 338-346, Feb. 1996.

[9] Y. Wang, Y. Jin and Q. Peng, "Merged quadtree fractal image compression," Optical Engineering, vol. 37, no. 8, pp. 2284-2288, Aug. 1998.

[10] H. Lin and A. N. Venetsanopoulos, "Fast fractal image compression using pyramids," *Optical Engineering* vol. 36, no. 6, pp. 1720-1730, Jun. 1998.

[11] D. Saupe and R. Hamzaoui, "A review of the fractal compression literature," Computer Graphics, vol. 28, no. 4, pp. 268-276, 1994.