ACCELERATING FRACTAL IMAGE COMPRESSION BY ADAPTIVE DOMAIN POOL SCHEME.

D. Venkatasekhar, P. Aruna

Department of computer science and Engg, Annamalai University, Annamalai nagar, Tamil nadu, India.

ARTICLE INFO

Corresponding Author:
D. Venkatasekhar,
Department of computer science and Engg, Annamalai University, Annamalai nagar, Tamil nadu, India.

KeyWords: Fractals, Domain pool, Stepsize

ABSTRACT

An adaptive domain pool scheme for fractal image compression is proposed in this paper. This scheme adaptively selects domain pool for each range block based on the range's location, so there is a more effective and smaller domain library for each range. This technique results in the decrease of the computational load and the bits needed to store the domain location. The experimental results on standard images show that the proposed method yields a far better rate-distortion performance and has a far lower computational load. And finally we also perform the comparison of this method against the block partition (domain pool scheme) method.

©2011, AjCSIT, All Right Reserved.

INTRODUCTION

The demand for images, computer animations has increased drastically over the years. This has resulted in image compression becoming an important issue in reduce the cost of data storage and transmission. JPEG is currently the accepted industry standard for still image compression, but alternative method 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 domain. Most data contains some amount of redundancy, which can sometimes be removed for storage and replaced for recovery, but this redundancy does not lead to high compression ratios.

In the basic fractal image compression, an image is partitioned into a set of nonoverlapping blocks of size r*r, called range blocks. And all the possible overlapped blocks of size 2r*2r called domain blocks from the domain pool. The domains, shrunk by pixel averaging to match the range size, are used as the codebook to approximate each range block with a transformation. Moreover, the codebook is enlarged by including all rotations and reflections of each domain. The partition of the image, the scaling and offset values for each range and the best match domain index need to be stored in order to reconstruct the image. The problem of the basic method is the heavy computation of searching the best-match domain block for each range block. For an n*n image, the number of range blocks are (n^n/r^r), and the number of domain blocks are (n-2r+1)*(n-2r+1). The computation of best match between a range block and a domain block is O (r^2). If r is constant, the computation complexity of entire search is O (n^4). Due to the huge computational load, the encoding phase is time consuming. Many techniques have been proposed in order to speed-up fractal image encoding. Most of these methods reduce the number of domains to decrease the computational load according to some rules. Despite the advances made the loss of the fidelity still remain the main drawbacks of these speed-up techniques.

In this paper, we implement an innovative method of partitioning an image, which reduce the computational complexity in the encoding stage, which adaptively selects domain pool for each range block based on the range's location. This method yields a far better rate-distortion performance and has a far lower computational load. This method reduces the number of domains to decrease the computational load according to some rules. The range block is selected in a local area centered at domain region. The domains are selected in a local area centered at each range. This method drastically reduces the encoding time and increase computational speed. The compression rate and the PSNR of the reconstructed image are also increased because of more effective and smaller domain library.

Fig 1 Block diagram for fractal image compression

- Input Image
- Partitioning Image
- Range Blocks
- Suitable Domain Search
- Domain Blocks
- Domain Pool
- Fractal Encoder
- Fractal Coded Image
- Fractal Decoder
- Reconstructed Image
 Venkatasekhar et. al/ Accelerating Fractal Image Compression By Adaptive Domain Pool Scheme

PARTITIONING IMAGE
An image is partition into set of nonoverlapping block (range block) and overlapping block (domain block). In order to decrease the coding time, many methods are proposed by changing the searching means while block matching. Namely, the range block needn’t be compared to the transformations of all domain blocks. However, as the size of the blocks is fixed, the compression ratio, the quality of the reconstructed image and the coding time are contradictory.

In the general methods of selecting the domain, the subsquares of the image whose upper-left corners are positioned on a lattice are selected as the domains, the lattice spacing decide the size of the domain pool. Figure 2 describes this scheme. This is a 48*48 image, where the range is 4*4, the domain is 8*8, the lattice spacing equal to 4. The positions of the black squares are the upper-left corners of the domains. The domainblocks are well distributed over the whole image.

Fig 2 General domain pool selecting scheme

In the proposed method, the domains are selected in a local area centered at each range, not over the whole image. This local area is called domain region. The distance between two nearby domain blocks in the horizontal or vertical direction is called search stepsize. The domain region and the search stepsize decide the size of the domains. Figure 3 shows the domain pool selecting scheme of the proposed method, where the range is 4*4, the domain is 8*8, and domain region is a square with side 12, search stepsize equal to 2. The positions of black squares in the domain region are the upper-left corners of the domains. The most time consuming part of the encoding procedure is usually the search for finding the best matching domain blocks. During encoding a large pool of image subsets, called domains, has to be searched repeatedly many times, which by far dominates all other computations in the encoding process.

Fig 3 Adaptive domain pool selecting scheme

FRACTAL ENCODER AND DECODER
In general method the rotation and the reflection of each domain blocks are stored in a separate book called codebook, this helps in order to reconstruct the image. In this method the scaling and offset values and the match-domain’s index are stored for each range block in order to reconstruct the image. Huffman coding is used to store the scaling and offset for each range block.

EXPERIMENTAL RESULTS
The experimental result for an image compression is, consider a 512*512 Lena is selected as the test image, the following domain pools are compared in the experiments.

D1: the domains are selected as the subsquares of the image whose upper-left corners are positioned on a lattice, and the lattice has spacing equal to 64, 32, 16, 8, and 4 respectively. For the 512*512 image, it’s 4, 6, 8, 10, 12 bits respectively to be required to store the domain index. Accordingly, the numbers of operations to find the bestmatch domain are16, 64, 256, 1024, and 4096 respectively for each range.

Fig 4 Reconstructed image D1
(PSNR: 28.60 dB; bits: 4; computation: 16)

D2: the domains are selected as the subsquares of each range’s domain region. Search stepsize are set as 1. The number of domain-range comparison is set as 64, 256, 1024, and 4096 respectively for each range by changing the domain region.

Fig 5 Reconstructed image D2
(PSNR: 31.68 dB; bits: 4; computation: 16)

D3: the domains are selected as the subsquares of each range’s domain region. Search stepsize are set as 2. The number of domain-range comparison is set as 64, 256,
The PSNR of the reconstructed image, the number of matching operations for each range, and the bits needed to store the domain index for various domain pool selection schemes. About 3dB is improved for using domain pool D2 and D3 compared with using domain pool D1.

The comparison between the PSNR verses bits shows that at least 7 bits are needed for D1 to store the domain index, but only 4 bits for D3 in order to gain the same PSNR (31.97dB). So the total bits needed for each range block reduce from 18 to 15. About 3 bits are saved for each range block for the proposed scheme. The compression rate increases by 20%. In the same way, the comparison between the PSNR verses operations, shows that about 160 comparisons are needed for D1, but only 16 for D3 in order to gain the same PSNR (31.97dB). So the computational load decreases by 90%.

**CONCLUSION**

An adaptive domain pool scheme is proposed in this paper. Experimental results on standard images show that the proposed method yields a far better rate-distortion performance and needs a far lower computational load.

**REFERENCES**