

Fractal Image Coding-Based Image Compression Using Multithreaded Parallelization



Ranjita Asati, M. M. Raghuwanshi, and Kavita R. Singh

Abstract Fractal image coding-based image compression is characterized by its high compression ratio, high-resolution, and lower decompression time. In spite of these advantages, it is not being widely adopted because of its high computation time. Attempts made to reduce the computation duration in fractal image compression (FIC) fall into two categories like heuristics-based search time reduction and parallelism-based reduction. In this work, we have proposed a multithreading-based parallelism technique on the multi-core processors to minimize the compression duration. The compression duration of the suggested multithreading process is tested upon the images having different resolutions. It is observed that the proposed solution has reduced the compression time by almost 2.51 times as compared to sequential method.

Keywords Fractal image compression · Encoding · Fractal image coding · Data parallelism · Multithreading

1 Introduction

Image compression is the technique of encoding an image to lessen the image size which results into efficient storage and transmission. Compression schemes exploit the irrelevant and redundant information in the images [1]. Existing compression techniques are of two types: lossless and lossy. There is no information loss and the compressed image retains its quality in case of lossless methods. There is a bearable information loss in lossy compression methods to achieve higher compression ratio. Fractal image coding is a lossy compression scheme and aimed on reducing the redundancy in the self-similar images [2]. It achieves higher compression ratio

R. Asati (✉) · K. R. Singh

Department of Computer Technology, Yeshwantrao Chavan College of Engineering, Nagpur, India
e-mail: ranjita.asati@gmail.com

M. M. Raghuwanshi

Department of Data Science, G.H. Raisoni College of Engineering and Management, Pune, India

by losing some information in the compressed image. FIC permits an improbable volume of data to be kept in greatly encoded records. A fundamental feature of FIC is to convert images into fractal codes. This is identified as fractal scaling, ceasing original pixel structure. An image is converted forming a group of contractive similar transformations exploiting the self-affinity of the image. The parameters of these transformations are stored as compressed file [3]. FIC is able to provide higher compression ratio [4]. Higher the compression ratio, higher the computation complexity in the fractal coding schemes due to search of matching blocks [5].

Numerous efforts are made to decrease the intricacy of the fractal image compression (FIC). The attempts are divided into two categories; in first category, efforts are made to reduce the search time for matching blocks (algorithm-based approach), and the second category is focused on the use of modern computer architecture for speed up (hardware-based approach). The proposed work falls into second category that uses multithreading parallelism on multicore processors to decrease the compression time of an image. In this work, we make use multithreading-based parallelization on multicore processors to diminish the encoding duration. The search time for localization of approximate range block for each domain block can be reduced, and localization process can be done with least error due to this multithreading parallelization. Remainder of this paper is structured as follows: Sect. 2 put forward literature review on the fractal image coding techniques. Section 3 presents the suggested approach with its algorithm. Section 4 discourses the result part. In the end, some conclusions are presented in Sect. 5.

2 Related Work

FIC is a lossy compression technique developed by Barnsley [BH86] and Jacquin [2] to compress an image by encoding it as a transformation. Firstly, it divides the original image into domain blocks (they can be any size or shape). Then, a collection of possible range blocks is defined. For each domain block, the algorithm then examines for a suitable range region that, when applied with an proper affine transformation, very closely look like the domain block. Later, a Fractal Image Format (FIF) file is generated for the image. This file contains information on the choice of domain regions, and the list of affine coefficients (i.e., the entries of the transformation matrix) of all allied affine transformations. This process is very time-consuming, especially during the search for the proper range regions. But once the compression is done, the FIF file can be decompressed very rapidly. The literature survey on existing works is studied in two categories of heuristics-based search time reduction and parallelism-based reduction.

2.1 *Heuristics-Based Search Time Reduction*

Ismail et al. [6] proposed a hybrid technique using variable range block and optimum domain block to decrease the compression duration in FIC. The solution is based on hexagonal structure of spiral architecture. The reduction in encoding time is achieved by reducing the number of matching operations and substituting its equivalent results with lower mean square error. Borkar et al. [7] reduce the encoding time in FIC using wavelet transform. In addition to reduction in time, use of wavelet also helps to enhance the visual appearance. Nadia et al. (2017) suggested a novel approach using fractal dimension to reduce the search process for each range block. This approach makes a small compromise in image quality to decrease the compression time. Wang et al. [8] presented a practice to accelerate FIC using Pearson's correlation coefficient. Sparse searching is implemented to find the matching domain region; due to this, encoding time is reduced at a cost of minor compromise in the quality of image. Hsu et al. [9] proposed a repetition-free FIC. A domain pool constitutes the mean image for both encoding and decoding process. This domain pool is able to avoid the iteration process. The method has lower encoding and decoding time, but it compromises the PSNR. Cao et al. [10] proposed a novel orthogonal sparse FIC to improve compression time. This method takes into account image texture feature. Most of these works achieved reduction in compression time by compromising the PSNR.

2.2 *Parallelism-Based Reduction*

Parallel computing is a computing where the tasks are fragmented into distinct parts that can be accomplished simultaneously. Every part is again fragmented into a series of instructions. These instructions are then carried out simultaneously on different processors. Parallel systems make simultaneous use of multiple computer resources that can employ a single computer with multiple processors, a computer network forming a parallel processing cluster or a combination of both. The heart of parallel processing are CPUs. Flynn's taxonomy is a peculiar arrangement of computer architectures dependent on the number of parallel instruction (single or multiple) and data streams (single or multiple) available in the architecture.

Min et al. [11] achieved speed up of FIC using a massively parallel implementation on a pyramid machine. Authors could diminish the computation intricacy from $O(n^4)$ to $O(n^2)$ for an $n \times n$ image using the parallel architecture. Erra et al. [12] proposed a parallel FIC using programmable graphics hardware. The solution used SIMD architecture to speed up the base line approach of fractal encoding. Palazzari et al. [13] exploited the massively parallel processing on SIMD machines to solve the problem of higher encoding time in FIC. Lee et al. [14] suggested a parallel quad tree-based FIC. It performs the fractal image coding based on quad tree partitioning. Hufnagl et al. [15] proposed a parallel version of FIC over massively parallel SIMD

arrays. The speed up is achieved in encoding time based on load balancing over 2D SIMD arrays. Bodo et al. [16] proposed a quad tree-based FIC scheme. The encoding time is reduced in this parallel architecture by maximizing the processor utilization. Haque et al. [17] carried out sequential as well as parallel FIC in Compute Unified Device Architecture (CUDA) platform. Medical images were used for experimentation achieving 3.5 times more speed up compared to CPU. Saad et al. (2016) proposed a parallel architecture using FPGA for implementing a full search FIC. A near optimal performance is achieved using low-cost FPGA hardware. Malik et al. [18] proposed FIC algorithm employing deep data pipelining for high-resolution images. By limiting the search in neighboring blocks, reduced further encoding time. The solution can compress large size images in less time.

Though these parallelism-based solution are able to achieve significant speed up, but they require expensive hardware's.

3 Proposed Solution

The proposed solution for achieving speed up in FIC based on the concept of adding multithreading feature to the original FIC encoding and decoding process so that it becomes more suitable for the data level parallelism without much inter thread communication. The serial version of FIC is modified to incorporate multithread parallelism. In the serial FIC, the initial image of dimension $M * M$ is split to m non-overlapping range blocks of size $r * r$ where $m = M/r^2$ and n overlapping domain blocks of size $2r * 2r$ where $n = (M - 2r + 1)^2$.

Fractal coding follows Iterated Function System. In first step image is partitioned into non-overlapping blocks called range blocks and overlapping sub-blocks called domain blocks. Each range block is mapped with one of the available domain blocks. To locate most suitable matching domain block, a range block has to be compared with each domain block to record the minimum distortion. Eight affine transformations do not change pixel values; they simply shuffle pixels within a range block, in a deterministic way. For every range block, search is done on the domain pool for a block which can be mapped linearly to the range block with smallest error. As a result, an approximate domain block and a suitable contractive affine transformation is selected according to the minimization function below

$$d(R_i, w_{ik}(D_k)) = \min d(R_i, w_{ij}(D_j)) \quad (1)$$

where w_{ik} is the contractive affine transformation from D_k to R_i . This is done in such a way that mean square error distance (represented as $d(R_i, w_{ij}(D_j))$ from range block R_i and the transformed domain block $w_{ij}(D_j)$) is minimized.

Each thread processes a range block in the range block pool. The thread does transformation on domain blocks, finds the most suitable transformed blocks based on least mean square error between the transformed domain block and range block.

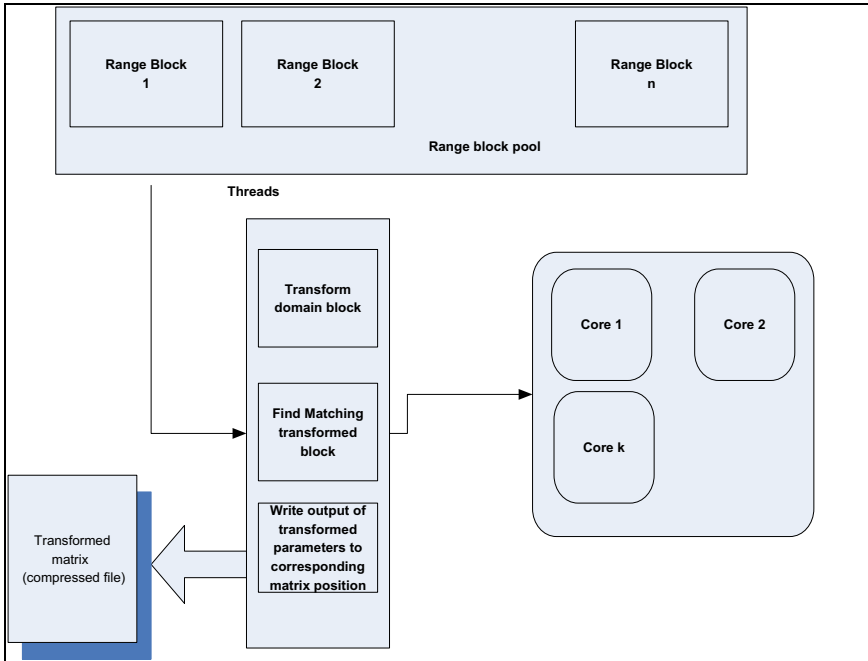


Fig. 1 Proposed architecture

The threads are scheduled for execution on the any cores of the multi core processor by the OS. Since there is no inter thread communication, the usual pitfalls of multithread communication like resource contention, racing, deadlocks etc. are totally avoided. Each thread generates the transformed parameters for it range block which is written to the matrix and returned as a compressed file. In the proposed work, multiple threads are assigned to lookup for domain block for each range block. Each read takes the range block as input. For the range block it iterates through the domain block pool and select the domain block which on transformation can be approximated to a range block with minimum distortion. Each searching for domain block is done in parallel, it reduces the encoding time. Each thread will execute in parallel on different cores. Since processing steps for each range block use same instructions but different data, a data level parallelism can be achieved without much inter thread cooperation. Only when returning the final compressed file in terms of transformation parameters, a wait must be done to collect the results from all threads. The parallelism architecture of the proposed solution is given in Fig. 1. Both the encoding and decoding process are parallelized to achieve speed up. The steps in the parallelized encoding process are given below.

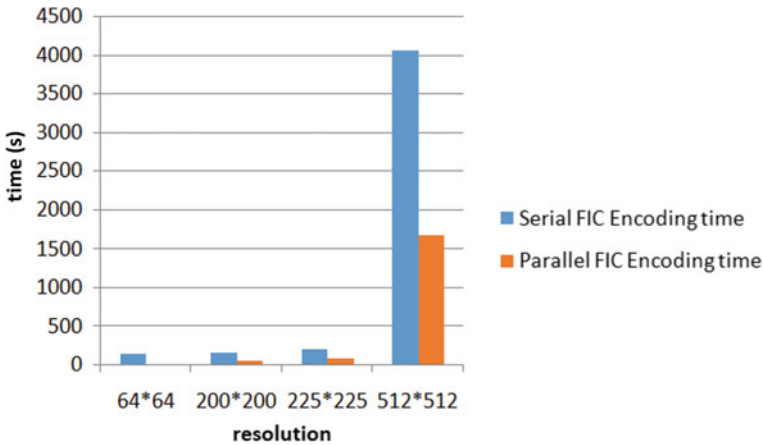


Fig. 2 Serial and parallel FIC encoding time

Algorithm: Parallel FIC Encoding

Step 1. Input image is divided into non-overlapping range regions and overlapping domain regions.

Step 2. For every range region start a thread.

In each thread for the domain region, find the approximate range region by selecting the range block with lowest MSE distance of range region and transformed domain regions.

Step 3. Wait for all thread to complete.

Step 4. Return all the transformations.

The decoding process is parallelized by splitting the compressed file into portions and doing decoding process on each portion. Due to parallel reconstruction, the time of reconstruction is reduced. The steps in decoding algorithm are given below.

Algorithm: Parallel FIC Decoding

Step 1. Generate range blocks from the encoding result file.

Step 2. Chose a starting image, which should be of same dimension as that of the initial image.


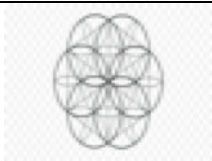



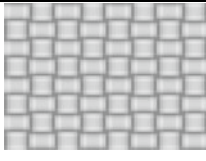

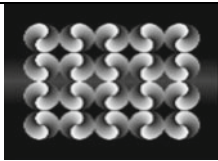

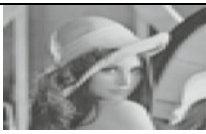
Step 3. Split range block matrix to K partition and start K thread.

Step 4. In each thread do following.

Apply the transformation parameters derived from the transformed block for each range block.

Substitute the range block pixels with the pixels which are obtained from the transformed block.

Table 1 Image used for experimentation

Image		
		
		
		
		

Both transformations and mappings are enforced on the initial image repetitively until the image is restored.

Step 5. Wait for all thread to complete.

Step 6. Return the reconstructed image.

Though only concern in FIC is encoding time, in this work, decoding process is also parallelized just to increase the resource utilization of multiple cores (Table 1).

4 Results

The performance of the proposed solution was conducted in 10th generation Intel core i5 10300H, 2.5 GHZ, 4 cores, 8 threads, 8 GB RAM. Performance was tested with 40 images in different resolutions of 64 * 64, 200 * 200, 225 * 225, and 512

Table 2 Comparison of encoding and decoding time

Size	Serial FIC	Serial FIC	Parallel FIC	Parallel FIC
	Encoding time (s)	Decoding time (s)	Encoding time (s)	Decoding time (s)
64 * 64	36.49	0.94	7.59	0.94
200 * 200	155.3	3.24	29.63	1.88
225 * 225	228.3	4.36	93.40	2.77
512 * 512	4058.1	18.84	1824.4	11.00

* 512. The dataset image for performance testing is obtained by taking 10 different images in resolution of 512 * 512 and rescaling them to dimensions of 64 * 64, 200 * 200, and 225 * 225. By this way, dataset of 40 images was created with different scales. By this way of testing the FIC against these recalled images, the robustness of the proposed FIC algorithm against different scales can be tested. These images were then used for FIC testing. Since there were no standard dataset for testing fractal image compression, we used the same set of images used for testing block complexity-based FIC proposed in [19].

The proposed solution was compared with serial version of FIC implementation with both encoding and decoding process realized as a single process. The encoding and decoding time are measured for different images, and the result is summarized in Table 2. The results for encoding time are as below in Fig. 2.

From this result, it is evident that as the resolution of the image increases, the encoding time increases, but the increment is lower in the proposed solution compared to Serial FIC. The average encoding time in Serial FIC is 1145 s, but in the proposed solution, the average encoding time is 455 s. Thus, the proposed solution has an average speed up of 2.51 compared to Serial FIC. The average decoding time in Serial FIC is 7.14 s, but in the proposed solution, it is only 3.67. The proposed solution has a 1.95 times speed up in decoding time compared to Serial FIC. The speed up in encoding process in the proposed solution is due to parallelization in the steps of calculation of transformations and matching to every transformation to find best match with least error.

The reduction in the decoding time is due to parallelization in the step of replacing the pixels of range block from the transformed block of each range block (Fig. 3).

The box-whisker plot of encoding time for different images in four different scales of 64 * 64, 200 * 200, 225 * 225, and 512 * 512 is given in Fig. 4.

Though earlier many solutions have been proposed for FIC, they did not fully exploit the processing capability of the underlying hardware. Due to this, hardware was underutilized and it took long time for compression. Parallelization was the solution to increase the hardware utilization and reduce the time for compression. There were two options for parallelization, process level and thread level. In case of FIC, thread level parallelization was suited more than process level as the processing overhead of inter-process communication would be high for FIC, and it would ruin the advantages of parallelization. Due to this, thread-level parallelization was adopted.

The results proved that compression time is reduced by increasing the utilization of underlying hardware using threads.

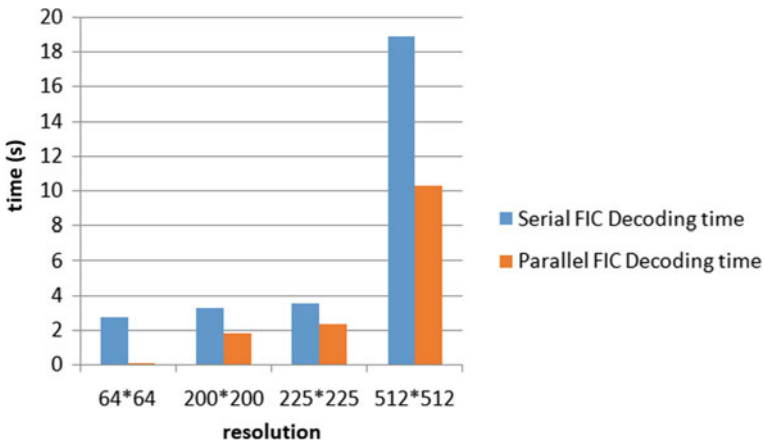


Fig. 3 Serial and parallel FIC decoding time

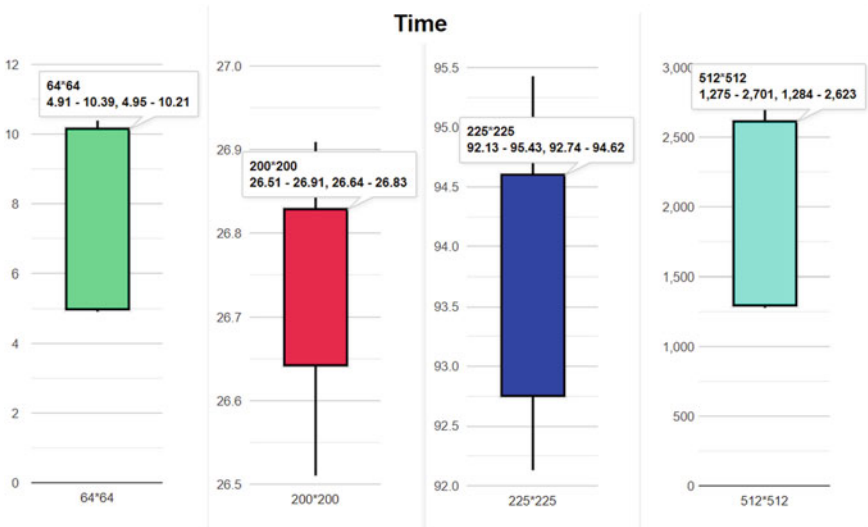


Fig. 4 Box-whisker plot of encoding time

5 Conclusions

In this work, a multi thread-based parallelization using multi-core processors is implemented to speed up the execution of FIC. FIC is generally resource and time-consuming algorithm, and the computation complexity is exponentially increasing with increase in size of the image. Though many algorithms have been proposed for FIC, they have two issues of underutilization of hardware and large time for encoding. In this work, thread-level parallelism is introduced to FIC to solve the two issues of underutilization and larger encoding time. The performance of the proposed solution was tested for different images at different scales, and results show an average 2.51-time reduction in encoding time. The reduction in encoding time was achieved at the cost of increasing the hardware utilization through thread-level parallelism. A limitation in this approach is that for self-similar blocks, the computation effort is repeated. This limitation can be solved by further parallelization by splitting the image to high-level blocks and clustering them based on entropy or structural similarity and applying parallel FIC for a representative block in each cluster and approximating other blocks in cluster based on it. This will further reduce the encoding time.

References

1. Hussain A, Al-Fayadh A, Radi N (2018) Image compression techniques: a survey in lossless and lossy algorithms
2. Jacquin A (1989) A fractal theory of iterated Markov operators with applications to digital image coding
3. Asati R, Raghuwanshi MM (2020) Fractal image compression: a review. *Int J Future Gener Commun Network* 13(1s):66–75
4. Wohlberg B, de Jager G (1999) A review of the fractal image coding literature. *IEEE Trans Image Process* 8
5. Fisher Y (1995) *Fractal image compression: theory and application*. Springer, New York
6. Ismail M, Reddy BTB (2016) Spiral architecture based hybrid fractal image compression. In: *International conference on electrical, electronics, communication, computer and optimization techniques (ICEECCOT)*
7. Borkar E, Gokhale A (2017) Wavelet based fast fractal image compression. In: *International conference on innovations in information embedded and communication systems (ICIIECS)*
8. Wang JJ, Chen P, Xi B et al (2017) Fast sparse fractal image compression. *PLOS ONE* 12(9)
9. Hsu C-C (2018) Iteration-free fractal mating coding for mutual image compression. In: *International symposium on computer, consumer and control (IS3C)*
10. Cao J, Zhang A, Shi L (2019) Orthogonal sparse fractal coding algorithm based on image texture feature. *IET Image Process* 13(11):1872–1879
11. Min X, Hanson T, Merigot A (1994) A massively parallel implementation of fractal image compression. In: *IEEE international conference on image processing*
12. Erra U (2005) Toward real time fractal image compression using graphics hardware. *Adv Vis Comput Proc Lect Notes Comput Sci* 3804:723–728
13. Palazzari P, Coli M, Guglielmo L (1999) Massively parallel processing approach to fractal image compression with near-optimal coefficient quantization. *J Syst Archit* 45:765–779

14. Lee S, Omachi S, Aso H (2000) A parallel architecture for quadtree-based fractal image coding. In: Proceedings of 2000 international conference on parallel processing, pp 15–22
15. Hufnagl C, Uhl A (2000) Algorithms for fractal image compression on massively parallel SIMD arrays. *Real-Time Imag* 6:267–281
16. Bodo ZP (2004) Maximal processor utilization in parallel quadtree-based fractal image compression on MIMD Architectures. *Informatica XLIX*(2)
17. Haque ME, Al Kaisan A, Saniat MR (2014) GPU accelerated fractal image compression for medical imaging in parallel computing platform
18. Abdul-Malik HYS, Abdullah MZ (2018) High-speed fractal image compression featuring deep data pipelining strategy. *IEEE Access* 6
19. AlSaidi NMG, Ali A (2017) Towards enhancing of fractal image compression performance via block complexity. In: Annual conference on new trends in information & communications technology applications-(NTICT'2017) 7–9 Mar 2017