

Improving Image Compression by Optimizing Fractal Encoding using DCT-based Classification

Milind Kulkarni¹, Aditya Naikwad², Aman Pal³, Aryan Naik⁴, Bhairavi Deshmukh⁵ and Tejas Bharambe⁶

¹Professor, Vishwakarma Institute of Technology, Pune, India

Email: milind.kulkarni@vit.edu

²⁻⁶UG Students, Department of Artificial Intelligence and Data Science, Vishwakarma Institute of Technology, Pune, India

Email: {arun.aditya211, aman.pal21, aryan.naik21, bhairavi.deshmukh21, tejas.bharambe21}@vit.edu

Abstract—This paper introduces two fractal image compression methods in the DCT domain. The first method eliminates domain block coordinate encoding, enhancing efficiency with small range blocks. The second reduces encoding time using domain pool reduction and preset contrast scaling. Both methods achieve superior performance and faster encoding with high visual quality.

Index Terms— DCT, quadtree partitioning, domain pool, Fractal Image compression.

I. INTRODUCTION

Fractal image compression is a preferred approach that has garnered a lot of attention due to its capability of achieving huge compression ratios without affecting quality of image. Unlike traditional methods that rely on the correlation between adjacent pixels, fractal coding exploits self-similarities within an image to achieve higher compression performance. This self-similarity property allows fractal coding to effectively compress natural images, textures, and other images that exhibit repetitive patterns.

Fractal image compression is founded on the concept of fractals, which are mathematical objects with varying sizes of self-similarity. This means that a fractal can be divided into smaller parts that are similar to the whole. Fractal coding makes use of this self-similarity by expressing an image as a collection of transformations that project the image's smaller components onto the larger image. These transformations are typically affine transformations, which include scaling, rotation, and translation

With fractal coding, compression ratios as high as 100:1 are possible, or more significantly, reducing file sizes while preserving image quality. It also produces reconstructed images with good visual quality, even at high compression ratios. Fractal-encoded images can be decoded at different resolutions, allowing for flexible viewing and zooming capabilities. It is relatively fast compared to other compression techniques. Fractal image compression has the potential to be applied in several ways, such as digital photography, medical imaging, satellite imagery, etc. Despite its advantages, fractal image compression faces certain challenges that hinder its widespread adoption.

- High computational complexity: Fractal encoding involves computationally expensive operations, particularly during self-similarity matching, which can significantly increase encoding time.
- Limited applicability: Fractal coding is most effective for natural images, textures, and images with self-similar patterns. It may not be as suitable for images with smooth regions or high contrast.

To overcome these challenges, this paper proposes a hybrid approach that leverages insights from DCT-based classification to optimize fractal encoding within the DCT domain. By synergizing these techniques, the goal is to enhance compression efficiency while maintaining or even improving visual quality.

The subsequent sections will delve into the details of the proposed methods, including image partitioning, domain pool(DP) reduction, DCT transformation, contrast scaling factor estimation, fractal encoding, and decoding processes. Experimental results and comparisons with existing algorithms will be presented to demonstrate the efficacy and superiority of the proposed approaches in achieving enhanced compression performance and perceptual quality.

II. LITERATURE REVIEW

Hurd et al [1] used methods for compressing videos that rely on fractal geometry and fractal transform. High compression ratios, quick decompression times, and error-insensitivity were all obtained using the compression technique. Abedellatif, Heba, et al [2] created a DP encoder using fractal coding together with picture segmentation algorithms. The outcome was compared with other methods like traditional fractal and it gave better performance than them.

Zhi-liang Zhu, Fu et al. [3] proposed a image coding technique based on fractals that used Discrete Cosine Transform (DCT), which improved the method of exploiting self-similarities. This approach uses the horizontal and vertical DCT coefficients having low frequency to categorize the blocks into three types: diagonal/slash-diagonal, vertical/horizontal and smooth. Phadatare et al. [4] developed a method that combines fractal compression with DCT techniques, resulting in improved PSNR and faster encoding for the reconstructed image. This approach also shortens the time and memory space necessary to save photos.

Gupta et al [5] provided a study on the fractal image compression (FIC) methods' computational difficulty. It looked at a number of variables that affect the encoder's computational cost. To demonstrate how the encoder comprehends delay, the automatic baseline fractal compression algorithm is looked at. The study demonstrates how different strategies balance CPU time, compression ratio, and decoder quality.

Saad, Abdul-Malik HY, et al [6] presented a full-search dynamic methodology for the FIC technique. This search is initiated from D which is the domain block (DB) nearest to the range block (RB) that must be encoded, according to proposed methodology. The search window is continually expanded until a good matching DB is found or the full DP is investigated. Lakshmi, GV Maha [7] developed an ANN-based FIC technique for MRI pictures. The IFIC technique, which was trained using the characteristics of an MRI image of a single brain, was used to compress other MRI images of brain. The BPA divides the RBs and DBs into 2 classes. This lowers the search space and accelerates the encoding process.

III. METHODOLOGY

A. Image partitioning

In this crucial step of FIC, the input image is divided into non-overlapping RBs of a predetermined size, denoted as b . These range blocks serve as the units for compression and reconstruction. To enhance the search process, a variety of DBs are constructed from the actual image. These domain blocks are essential for getting the finest match for each RB and are obtained by taking all square blocks of $2b \times 2b$ size having an integer step size. Three additional domain blocks have been added to the DP, which are generated from each original block by clockwise rotation at 90-degree intervals, resulting in a total of seven new domain blocks including their mirrored counterparts. This augmentation enriches the search space, increasing the likelihood of finding optimal matches during compression.

After the DP is constructed, we match each range block with the best-fitting domain block through an affine transformation that minimizes the Euclidean distance between them in the luminance dimension. The distance between the n -pixel-dense RB, R , and the decimated DB, D , can be represented as follows:

$$E(R,D) = \sum_{i=1}^n (Sd_i + O - r_i)^2 \quad (1)$$

To minimize the objective function, the optimal coefficients S and O , which indicate the degree of similarity between the range and domain blocks, must be determined. These coefficients are defined as follows:

$$S = \frac{\langle R - \bar{R} \cdot 1, D - \bar{D} \cdot 1 \rangle}{\|D - \bar{D} \cdot 1\|^2} \quad (2)$$

$$O = \bar{R} - s\bar{D} \quad (3)$$

where $\langle, \rangle$ denotes inner product, R denotes range block, D denotes domain block, and $\bar{D}$ and $\bar{R}$ are means of D and R respectively.

Following the initial partitioning of the image into RBs, the quadtree partitioning process commences as part of the fractal coding algorithm. Remaining RBs undergo subdivision into 4 smaller blocks during the matching process, retaining only those that can be mapped with an acceptable error. This subdivision effectively shrinking the RBs, triggering a new iteration of the matching process for these smaller blocks.

For instance, if the initial RBs are 16 x 16 pixels, the quadtree partitioning process will yield range blocks of sizes 8 x 8, 4 x 4, and 2 x 2 pixels successively, constituting a four-step method.

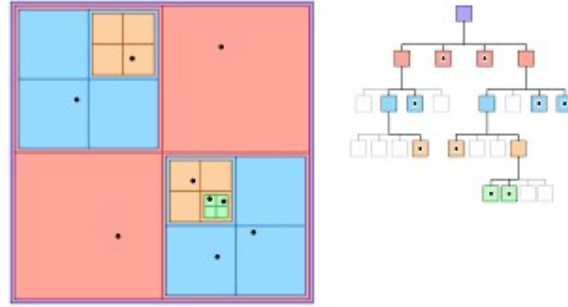


Fig1. An example of quadtree partitioning

At each step of the quadtree partitioning, the decision to encode a range block or further subdivide it for encoding in subsequent steps is based on a threshold for mapping error. As the quadtree partitioning progresses, the DP adjusts to the reduced size of the range blocks, incorporating smaller domain blocks to enable more precise matching.

This iterative process will continue until the smallest permissible range block size is reached, typically individual pixels. At this point, further subdivision would offer minimal benefit, and the remaining range blocks are encoded accordingly. The optimal transformations discovered at each step of the quadtree partitioning are retained for subsequent reconstruction.

B. Domain Pool Reduction

The initial approach involves restricting the DP to comprise exclusively of DBs having high energy. This restriction reduces the overall evaluation time needed to identify a suitable DB corresponding to any given RB. One important metric in this case is a block's entropy. Take $n \times n$ block N from the image, where the brightness level of each pixel fluctuates within $\{L_1, L_2, \dots, L_K\}$.

g_{11}	g_{12}	$\dots$	g_{1n}
g_{21}	g_{22}	$\dots$	g_{2n}
$\vdots$			
g_{n1}	g_{n2}		g_{nn}

Fig2. 'g_{ij}' indicates the pixel's brightness level at location(i,j)

Let q_i be the total number of observations of L_i across the pixels. The probability of L_i is calculated using the following formula:

$$p_i = q_i / \sum_{j=1}^K q_j = \frac{q_i}{n^2}$$

Subsequently, the entropy of the block is calculated using these probabilities. It is defined as follows:

$$\text{Entropy} = - \sum_{i=1}^K p_i \ln(p_i)$$

where p_i is the probability of occurrence of luminance level L_i in the block. After that, domain blocks are removed from the domain pool if their entropy is below a certain threshold.

C. DCT Transformation

The Discrete Cosine Transform (DCT) has demonstrated remarkable capability in compacting information, making it a preferred choice for image compression applications. Notably, popular compression standards like JPEG rely on the DCT for efficient encoding.

The DCT for an $L \times L$ image block, denoted as $f(i, j)$ for $i, j = 0, 1, \dots, L-1$, is defined below

$$F(m, n) = \frac{2}{L} C_m C_n \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} f(i, j) \cos\left(\frac{(2i+1)m\pi}{2L}\right) \cdot \cos\left(\frac{(2j+1)n\pi}{2L}\right)$$

Where $m, n = 0, 1, \dots, L-1$ and

$$C_k = \begin{cases} 1/\sqrt{2}, & k = 0 \\ 1, & k \neq 0 \end{cases}$$

When the Discrete Cosine Transform (DCT) is performed on an $L \times L$ image block, it generates $L \times L$ coefficients. However, applying the DCT directly to large image blocks can be computationally demanding and time-consuming. To address this, the DCT is typically applied to smaller blocks of the image.

The DCT is applied to each range block and its corresponding domain block, resulting in DCT coefficients $Fr(m, n)$ and $Fd(m, n)$, respectively. Initially, we store the DC coefficient ($Fr(1, 1)$) obtained from the DCT transformation of the RB. The appropriate scaling parameter value is then found, taking into consideration that $Fr(1, 1) = Fd(1, 1) = 0$. This scaling parameter is saved for subsequent use in the encoding process, selecting the value that best satisfies this condition.

D. Fractal Encoding

During fractal encoding, each range block is individually analyzed to determine the contrast scaling factor (S) that minimizes the disparity between the RB and its corresponding decimated DB in the domain of DCT.

Initially, the best-found value of S and the coefficient of DC ($Fd(1, 1)$) obtained from the DCT transformation of the range block are saved, provided that the mapping error falls below a predetermined threshold.

However, if the mapping error exceeds the threshold, indicating that the range block cannot be accurately mapped to the DB, the RB is subdivided to create four smaller blocks. Also, quad-tree algorithm is then employed to iteratively repeat the encoding process until the desired level of accuracy is achieved.

IV. RESULTS AND DISCUSSIONS

The effectiveness of the proposed technique was verified using numerous thresholds for each step and a multitude of experiments utilizing the common 512x 512 8bit/pixel. In this case, the range block sizes in the subsequent phase are 16×16 , 8×8 , 4×4 , and 2×2 , respectively. Two bits were conserved in the scaling coefficient. Additionally, each block's dc coefficient is quantized with k bits, one of which is used to indicate the coefficient's sign. For levels 1, 2, 3, and 4, k is 8, 7, 7, and 7 bits, respectively. PSNR is calculated using the formula as follows:

$$PSNR = 10 \log_{10} \left(\frac{255^2}{\frac{1}{N} \sum_{i=1}^N (f_i - f_i^*)^2} \right)$$

All of the experiments were run on a Windows XP-powered personal computer equipped with a 1.7 GHz CPU, 256 MB of RAM, and Matlab 7. The results in the spatial domain are compared with the nosearch algorithm in Table 1. According to the table, the DCT domain's nosearch algorithm outperforms the spatial domain's nosearch method in terms of coding. Table 2 presents a comparison between the suggested algorithm and the outcome of the nosearch algorithm in the wavelet domain. While all nosearch approaches are incredibly quick, the suggested method offers a better compression ratio while maintaining the same level of image quality.

TABLE I. COMPARISON OF NOSEARCH ALGORITHM IN SPATIAL DOMAIN AND DCT DOMAIN

	Spatial domain			DCT domain		
	Cr	t(s)	PSNR (db)	Cr	t(s)	PSNR (db)
Lena	36.4	17	29.44	41.15	18	29.44
Baboon	4.46	150	23.65	5.87	157	23.17
Peppers	8.61	25	32.11	9	26	32.12

TABLE II. COMPARISON OF NOSEARCH ALGORITHM IN WAVELET AND DCT DOMAIN

	Wavelet Domain		DCT Domain	
	Cr	PSNR (dB)	Cr	PSNR (dB)
Lena	6	29.98	20.04	32.05
Peppers	6.5	30.11	11.55	31

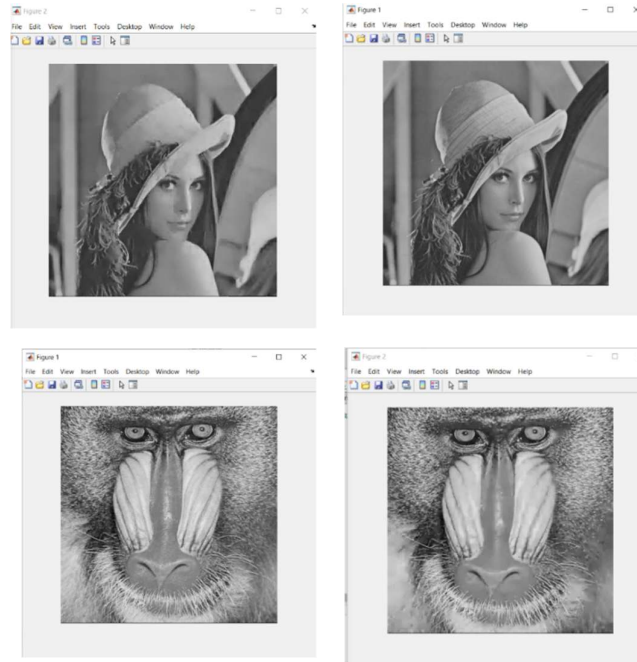


Fig3. Figures on left show original images while figures on right show decoded images

V. CONCLUSIONS

- The proposed method offers faster processing times compared to existing methods.
- It consistently delivers higher-quality reconstructed images.
- Its versatility allows for compression of a broader range of image types.
- The method signifies a promising advancement in fractal image coding techniques.
- Its implementation has the potential to greatly enhance the overall performance of fractal image coding systems.

VI. FUTURE SCOPE

- The proposed method can be further improved by developing a lossless compression algorithm that can be used to compress fractal codes.
- JPEG compression can be also combined to improve compression performance further.
- The method can be also applied to the wavelet domain.

REFERENCES

- [1] Hurd, L. P., M. A. Gustavus, and M. F. Barnsley. "Fractal video compression." COMPTON Spring 1992. IEEE Computer Society, 1992.
- [2] Abedellatif, Heba, et al. "Non-Exhaustive Search Method for Fractal Image Compression." 2021 International Conference on Electronic Engineering (ICEEM). IEEE, 2021.
- [3] Fu, Chong, and Zhi-liang Zhu. "A DCT-based fractal image compression method." 2009 International Workshop on Chaos-Fractals Theories and Applications. IEEE, 2009.
- [4] Phadatare, Pallavi, and Pratibha Chavan. "Implementation of DCT and Fractal Compression Technique." 2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE, 2021.
- [5] Gupta, Richa, Deepti Mehrotra, and Rajesh Kumar Tyagi. "Computational complexity of fractal image compression algorithm." IET Image Processing 14.17 (2020): 4425-4434.
- [6] Saad, Abdul-Malik HY, et al. "An Improved Full-search Fractal Image Compression Method with Dynamic Search Approach." 2020 10th IEEE International Conference on Control System, Computing and Engineering (ICCSCE). IEEE, 2020.
- [7] Lakshmi, GV Maha. "Implementation of image compression using Fractal Image Compression and neural networks for MRI images." 2016 International Conference on Information Science (ICIS). IEEE, 2016.
- [8] M.F. Barnsley, and S. Demko, "Iterated function systems and the global construction of fractals," Proc. Roy. Soc. Lond., vol. A399, pp. 243-275, 1985.
- [9] M.F. Barnsley, and L.P. Hurd, "Fractal Image Compression," Massachusetts: A.K. Peters, Wellesley, 1993.
- [10] A. Jacquin, "Image coding based on a fractal theory of iterated contractive image transformations," IEEE Transaction on Image Processing, vol. 1, pp. 18-30, 1992.
- [11] M. Barnsley, and A.D. Sloan, "A better way to compress images," BYTE Magazine, pp. 215-223, 1998.
- [12] Y. Fisher, "Fractal Image Encoding and Analysis," NATO ASI Series F 159. New York: Springer Verlag, 1998.
- [13] C.K. Lee, and W.K. Lee, "Fast fractal image block coding based on local variances," IEEE Transaction on Image Processing, vol. 7, no. 6, pp. 888-891, 1998.
- [14] Yao Zhao, Baozong Yuan, "A new affine transformation: its theory and application to image coding," IEEE Transactions on Circuits and Systems for video Technology, 1998, 8(3), pp. 437-442.
- [15] T.K. Truong, J.H. Jeng, I.S. Reed, et al, "A fast encoding algorithm for fractal image compression using the DCT inner product," IEEE Transactions on Image Processing, 1999, 9(4), pp. 529-535.
- [16] Yong Ho Moon, Hyung Soon Kim, Jae Ho Kim, "A fast fractal decoding algorithm based on the selection of an initial image," IEEE Transaction on image Processing, 2000, 9(5), pp. 941-945.